Debian GNU/Linux
y la Línea de Comandos

debian

Antonio Carrillo Ledesma
Karla Ivonne González Rosas

Debian GNU/Linux y la Línea de Comandos

Antonio Carrillo Ledesma y Karla Ivonne González Rosas
Facultad de Ciencias, UNAM
http://academicos.fciencias.unam.mx/antoniocarrillo

La última versión de este trabajo se puede descargar de la página:
https://sites.google.com/ciencias.unam.mx/acl/en-desarrollo
http://132.248.181.216/acl/EnDesarrollo.html

2025, Versión 1.0α[1]

Índice

1 Introducción

Actualmente tenemos 3 grandes sistemas operativos en el mercado[1]:

- Windows
- Unix
- GNU[2]/Linux

De los cuales, sus dignos representantes son: Windows, macOS, iOS, Android, Chrome OS y GNU/Linux con todas sus diferentes distribuciones[3]. Y sin temor a equivocarnos aseguramos que Android es la distribución de GNU/Linux más popular e iOS es el más popular de los UNIX.

¿Qué Sistema Operativo Usar? ¿Apple o Microsoft? ¿Windows o Linux? ¿Android o iOS? Son preguntas frecuentes que todos nos hemos hecho alguna vez y es que elegir un sistema operativo, una computadora o un dispositivo móvil no es tan simple. Al menos no lo era años atrás. En la actualidad las diferencias entre sistemas operativos de escritorio son cada vez

[1] Cuotas de mercado de diferentes sistemas operativos:

https://gs.statcounter.com/os-market-share/desktop/worldwide
https://netmarketshare.com

[2] GNU -es un acrónimo recursivo de «GNU no es UNIX»- es un sistema operativo de Software libre, es decir, respeta la libertad de los usuarios. El sistema operativo GNU consiste en paquetes de GNU además de Software libre publicado por terceras partes con distintas licencias que conforman una distribución.

[3] Una distribución de Linux es un sistema operativo compuesto por el Kernel de Linux, herramientas GNU, Software adicional y un administrador de paquetes. También puede incluir un servidor de pantalla y un entorno de escritorio que se utilizarán como sistema operativo de escritorio normal. El término es distribución de Linux (o distribución en forma abreviada) porque una entidad como Debian o Ubuntu 'distribuye' el Kernel de Linux junto con todo el Software y las utilidades consideradas por cada entidad como necesarias (como administrador de red, administrador de paquetes, entornos de escritorio, etc.) para que pueda ser utilizado como sistema operativo. Sus distribuciones también asumen la responsabilidad de proporcionar actualizaciones para mantener el Kernel y otras utilidades.

Entonces, Linux es el Kernel, mientras que la distribución de Linux es el sistema operativo. Esta es la razón por la que también se les conoce como sistemas operativos basados en Linux (hay otros Kernels como son FreeBSD, NetBSD y Hurd).

menos, hasta el punto que prácticamente cualquier servicio Online es compatible con Windows, Mac, GNU/Linux y las principales firmas de Software crean aplicaciones para las tres plataformas principales, salvo excepciones.

Poco tendremos que decir del sistema operativo de Apple, Mac o iOS, ya que son los sistemas operativos más bonitos y que mejores resultados han dado a todos los usuarios que los han probado. Mac es un sistema pensado para los profesionales de los sectores que necesitan de un equipo de cómputo que sea capaz de proveer programas específicos para: desarrolladores, programadores, diseñadores, periodistas, fotógrafos, músicos, DJ's y muchos más empleos que se benefician de este sistema operativo.

Después tenemos a Windows, un sistema operativo versátil pensado principalmente para un uso doméstico, aunque eso no quita que muchas empresas utilicen Windows en sus equipos de cómputo ya que es un sistema operativo que puede dar muy buenos resultados en este aspecto.

Sin embargo, llegamos a Linux, el gran desconocido por muchos. Un sistema operativo mucho más versátil que Windows y que puede ser igual o más profesional que Mac. Sin embargo, la ventaja que tienen estos dos sistemas operativos, es que vienen ya preparados y configurados para el tipo de mercado al que van dirigidos, pero GNU/Linux no. Esto es una ventaja y una desventaja al mismo tiempo, ya que si tenemos práctica, podemos hacer que el sistema operativo se adapte a nuestras necesidades sin problemas, además es Software libre (véase apéndice 14.2).

1.1 GNU/Linux

Nos permite trabajar de manera más segura -y gratuita- que los demás sistemas operativos, este se ve y es muy parecido a cualquier otro sistema UNIX -como Mac OS X-, de hecho la compatibilidad con UNIX ha sido una importante meta del diseño del proyecto Linux. No obstante, Linux es mucho más joven que la mayor parte de los sistemas UNIX.

En sus inicios, el desarrollo de Linux giraba en gran medida alrededor del núcleo del sistema operativo central: el ejecutivo privilegiado que administra todos los recursos del sistema e interactúa directamente con el Hardware. Desde luego, se requiere mucho más que este núcleo para producir un sistema operativo completo.

Resulta útil hacer la distinción entre el núcleo (Kernel) de Linux y el sistema Linux: el núcleo en Linux es una identidad de Software totalmente original desarrollada desde cero por la comunidad Linux (suele encontrarse en

el directorio */boot/* en el sistema de archivos); el sistema Linux, tal como lo conocemos hoy, incluye una multitud de componentes, algunos escritos desde cero, otros tomados en préstamo de varios proyectos o creados, en colaboración con equipos como el proyecto GNU de la Free Software Foundation -Fundación por el Software libre es una organización creada en 1985 con el propósito de difundir este movimiento-.

Ante la duda de que distribución de GNU/Linux usar, lo mejor es analizar ciertos parámetros o características de las distribuciones. Los puntos más relevantes en los que te deberías fijar para elegir la mejor son:

- Robustez y estabilidad: si buscas un sistema operativo para usarlo en producción, seguro que no quieres perder tiempo con fallos o problemas. Por eso, es importante elegir las distribuciones más robustas y estables que funcionen como relojes suizos. Algunos buenos ejemplos son Debian, Ubuntu, Arch, openSUSE y Fedora.

- Seguridad: no podía faltar la seguridad, es un tema prioritario. Muchas distribuciones de Linux respetan tu privacidad mucho más que otros sistemas operativos, ya que no reportan datos del usuario, o al menos dan la elección de no hacerlo. Aunque GNU/Linux es un sistema seguro de base, no hay que confiarse, los ciberdelincuentes cada vez están más atentos a este sistema y cada vez hay más Malware que lo afecta. Por eso, si vas a elegir una distribución para una empresa o servidor, este debería ser un criterio prioritario. Algunas como Debian, SUSE, RHEL, CentOS, etc., podrían ser buenos casos para servidores. Y también tienes proyectos más específicos centrados en la seguridad como Whonix, QubeOS, TAILS, etc.

- Compatibilidad y soporte: el Kernel Linux tiene soporte para multitud de arquitecturas diferentes, como x86, ARM, RISC-V, etc. Sin embargo, no todas las distribuciones ofrecen ese soporte de forma oficial. Por eso, si vas a usar la distribución en una arquitectura diferente, es importante que te informes de si cuentan o no con dicho soporte. Por otro lado está el tema de los controladores y la compatibilidad de Software. En ese caso, Debian, Ubuntu y las distribuciones basadas en ésta son las «reinas», debido a que hay infinidad de paquetes y controladores para ella (es una de las más populares).

- Paquetería: a pesar de que los paquetes estándar deberían ser los RPM, como se especifica en LBS, lo cierto es que distribuciones tan populares como Debian, Ubuntu han conseguido que DEB sea el predominante. Con la llegada de los paquetes universales se han resuelto algunos problemas, pero si deseas contar con la mayor cantidad de Software, ya sean apps o videojuegos, la mejor opción es DEB con Debian y Ubuntu.

- Usabilidad: esto no depende de la distribución en sí, sino del entorno de escritorio, y de otras partes como el gestor de paquetes, si tiene o no utilidades que faciliten la administración como las incluidas en Debian, Linux Mint, o YaST 2 en openSUSE/SUSE, etc. Aunque, por lo general, las distribuciones actuales suelen ser bastante fáciles y amigables, con algunas excepciones...

- Ligero vs pesado: muchas distribuciones modernas suelen ser más pesadas, es decir, que demandan más recursos de Hardware o solo soportan ya 64-bit. En cambio, existen algunos entornos de escritorio ligeros como KDE Plasma (que ha «adelgazado» mucho últimamente y ha dejado de ser el escritorio pesado que era), LXDE, LXQt, Xfce, etc., así como distribuciones más livianas pensadas para ordenadores más antiguos o con pocos recursos. Todas estas opciones se encuentran en Debian.

- Otros aspectos: otro factor a tener en cuenta es tus preferencias o gustos por ciertos sistemas. Por ejemplo:

  ◦ SELinux (Fedora, CentOS, RHEL,...) vs AppArmor (Ubuntu, SUSE, openSUSE, Debian...)

  ◦ systemd (la mayoría) vs SysV init (Devuan, Void, Gentoo, Knoppix,...)

  ◦ FHS (la mayoría) vs otros como el de GoboLinux.

  ◦ etc.

1.2 ¿Qué tan Seguro es GNU/Linux?

No es ningún secreto que el sistema operativo que elijas es un determinante clave de su seguridad (no sólo en internet, también la privacidad de tus datos en el equipo que usas). Después de todo, el sistema operativo es el

Software más crítico que se ejecuta en nuestra computadora o dispositivo inteligente: administra su memoria y procesos, así como todo su Software y Hardware. El consenso general entre los expertos es que Linux es un sistema operativo altamente seguro, posiblemente el sistema operativo más seguro por diseño. Examinaremos aquí algunos factores clave que contribuyen a la sólida seguridad de Linux y veremos el nivel de protección contra vulnerabilidades y ataques que Linux ofrece a los administradores y usuarios.

Seguro por Diseño cuando se trata de seguridad, los usuarios de Linux tienen una clara ventaja sobre sus contrapartes que usan Windows o MacOs. A diferencia de los sistemas operativos propietarios, Linux, en muchos sentidos, tiene seguridad integrada en su diseño central. El sistema operativo de código abierto cada vez más popular es de alta flexibilidad, configurable y diverso. También implementa un modelo estricto de privilegios de usuario y ofrece una selección de defensas de seguridad de Kernel integradas para protegerse contra vulnerabilidades y ataques. La transparencia del código fuente de Linux significa que las vulnerabilidades en él, que son inevitables hasta cierto punto en cualquier sistema operativo, casi siempre son de corta duración. Echemos un vistazo más de cerca a cada uno de estos factores y cómo contribuye a la seguridad anunciada de Linux.

La Ventaja de la Seguridad de Código Abierto el código fuente de Linux se somete a una revisión exhaustiva y constante por parte de los miembros de la vibrante comunidad global de código abierto y, como resultado de este escrutinio, las vulnerabilidades de seguridad de Linux generalmente se identifican y eliminan muy rápidamente. Por el contrario, los proveedores propietarios como Microsoft y Apple emplean un método conocido como "seguridad por oscuridad", donde el código fuente se oculta a los extraños en un intento de ocultar las vulnerabilidades de los actores de amenazas. Sin embargo, este enfoque generalmente es ineficaz para prevenir las vulnerabilidades modernas y, en realidad, socava la seguridad del código fuente "oculto" al evitar que personas ajenas identifiquen y reporten fallas antes de que sean descubiertas por actores malintencionados. Seamos realistas: cuando se trata de descubrir errores de seguridad, un pequeño equipo de desarrolladores propietarios no es rival para la comunidad mundial de usuarios-desarrolladores de Linux que están profundamente involucrados en su trabajo tanto para su propio beneficio como para el beneficio de la comunidad.

Un Modelo Superior de Privilegios de Usuario a diferencia de Windows, donde "todo el mundo es administrador", Linux restringe en gran medida el acceso a la raíz a través de un modelo estricto de privilegios de usuario. En Linux, el superusuario posee todos los privilegios, y a los usuarios comunes solo se les otorgan suficientes permisos para realizar tareas comunes. Debido a que los usuarios de Linux tienen pocos derechos de acceso automático y requieren permisos adicionales para abrir archivos adjuntos, acceder a archivos o ajustar las opciones del Kernel, es más difícil propagar Malware y Rootkits en un sistema Linux. Por lo tanto, estas restricciones inherentes sirven como una defensa clave contra los ataques y el compromiso del sistema.

Defensas de Seguridad de Kernel Incorporadas el Kernel de Linux cuenta con una variedad de defensas de seguridad integradas que incluyen Firewalls que utilizan filtros de paquetes en el Kernel, el mecanismo de verificación de Firmware UEFI Secure Boot, la opción de configuración Linux Kernel Lockdown y los sistemas de mejora de seguridad SELinux o AppArmor Mandatory Access Control (MAC). . Al habilitar estas funciones y configurarlas para brindar el más alto nivel de seguridad en una práctica conocida como autoprotección del Kernel de Linux, los administradores pueden agregar una capa adicional de seguridad a sus sistemas.

Seguridad a Través de la Diversidad existe un alto nivel de diversidad posible dentro de los entornos de Linux como resultado de las muchas distribuciones de Linux disponibles y las diferentes arquitecturas de sistema y componentes que presentan. Esta diversidad no solo ayuda a satisfacer los requisitos individuales de los usuarios, sino que también ayuda a protegerse contra los ataques al dificultar que los actores maliciosos elaboren de manera eficiente Exploits que puedan usarse contra una amplia gama de sistemas Linux/Linux. Por el contrario, la "monocultura" homogénea de Windows convierte a Windows en un objetivo de ataque relativamente fácil y eficiente (algo parecido también les pasa a las Mac).

Además de la diversidad de diseño que se ve en Linux, ciertas distribuciones seguras de Linux se diferencian en formas que abordan específicamente las preocupaciones de seguridad y privacidad avanzadas compartidas entre los Pentesters, los ingenieros inversos y los investigadores de seguridad.

Altamente Flexible y Configurable hay muchas más opciones de configuración y control disponibles para los administradores de Linux que para los usuarios de Windows y MacOs, muchas de las cuales se pueden usar para mejorar la seguridad. Por ejemplo, los administradores de sistemas de Linux tienen la capacidad de usar SELinux o AppArmor para bloquear su sistema con políticas de seguridad que ofrecen controles de acceso granulares, proporcionando una capa adicional crítica de seguridad en todo el sistema. Los administradores también pueden usar la opción de configuración Linux Kernel Lockdown para fortalecer la división entre los procesos de la zona de usuario y el código del Kernel, y pueden fortalecer el archivo sysctl.conf, el principal punto de configuración de parámetros del Kernel para un sistema Linux, para darle a su sistema una base más segura.

Linux: Un Objetivo Cada Vez Más Popular Entre los Ciberdelincuentes Linux alimenta la mayoría de los dispositivos y supercomputadoras de alto valor del mundo y la base de usuarios del sistema operativo está creciendo constantemente, y los ciberdelincuentes han tomado nota de estas tendencias. Los autores y operadores de Malware apuntan cada vez más a los sistemas Linux en sus campañas maliciosas. Por ejemplo, en los últimos años han estado plagados de cepas emergentes de Malware para Linux: Cloud Snooper, EvilGnome, HiddenWasp, QNAPCrypt, GonnaCry, FBOT y Tycoon se encuentran entre las más notorias. Dicho esto, Linux sigue siendo un objetivo relativamente pequeño, con el 96 % del nuevo Malware dirigido a Windows en 2022. Además, el reciente aumento de los ataques de Malware de Linux no es un reflejo de la seguridad de Linux. La mayoría de los ataques a los sistemas Linux se pueden atribuir a configuraciones incorrectas y una administración deficiente, lo que destaca una falla generalizada entre los administradores de sistemas Linux para priorizar la seguridad.

Afortunadamente, a medida que el Malware de Linux continúa siendo cada vez más frecuente y problemático, Linux cuenta con protección integrada contra ataques de Malware a través de su estricto modelo de privilegios de usuario y diversidad de diseño, y hay una selección de excelentes herramientas, Kits de herramientas y utilidades de análisis de Malware e ingeniería inversa que incluyen REMnux, Chkrootkit, Rkhunter, Lynis y Linux Malware Detect (LMD) disponibles para ayudar a los administradores a detectar y analizar Malware en sus sistemas.

Para Tomar en Cuenta la seguridad del sistema operativo que implementa es un determinante clave de su seguridad en internet, pero de ninguna manera es una protección segura contra Malware, Rootkits y otros ataques. La seguridad efectiva depende de la defensa en profundidad, y otros factores, incluida la implementación de las mejores prácticas de seguridad y el comportamiento inteligente internet, juegan un papel central en su postura de seguridad digital. Dicho esto, elegir un sistema operativo seguro es de suma importancia, ya que el sistema operativo es la pieza de Software más crítica que se ejecuta en nuestros dispositivos computacionales, y Linux es una excelente opción ya que tiene el potencial de ser altamente seguro, posiblemente más que su contraparte propietaria, debido a su código de fuente abierta, modelo estricto de privilegios de usuario, diversidad y base de usuarios relativamente pequeña.

Sin embargo, Linux no es una "bala de plata" cuando se trata de seguridad digital: el sistema operativo debe configurarse de manera adecuada y segura, y los administradores de sistemas deben practicar una administración responsable y segura para evitar ataques. Además, es fundamental tener en cuenta que la seguridad tiene que ver con las compensaciones, tanto entre seguridad y facilidad de uso como entre seguridad y facilidad de uso. Los administradores deben configurar sus sistemas para que sean tan seguros como sea práctico dentro de su entorno. En lo que respecta a la conveniencia, Linux tiene una pequeña curva de aprendizaje, pero ofrece importantes ventajas de seguridad sobre Windows o MacOs.

1.3 Debian GNU/Linux

Ian Murdock[†] fundó oficialmente el proyecto Debian[4] GNU/Linux el 16 de agosto de 1993. Hasta ese momento, el concepto de una distribución de Linux era nuevo. Ian pretendió que Debian GNU/Linux fuera una distribución realizada de forma abierta, siguiendo el espíritu de Linux y GNU. La creación de Debian GNU/Linux fue patrocinada por el proyecto GNU de la FSF durante un año (noviembre de 1994 a noviembre de 1995).

Debian GNU/Linux estaba pensada para ser desarrollada cuidadosa y conscientemente y ser mantenida y soportada con un cuidado similar. Lo que comenzó con un pequeño y muy unido grupo de Hackers de Software libre, fue creciendo gradualmente hasta convertirse en una gran comunidad

[4]El nombre de Debian surge de la unión del nombre Ian y de su esposa Debra.

de desarrolladores y usuarios bien organizada.

Debian GNU/Linux es un sistema operativo libre, que actualmente usa el Kernel GNU Hurd, FreeBSD y Linux como núcleo de todo el Software de la distribución Debian GNU/Linux. La distribución está disponible para varias arquitecturas:

- 64-bit PC (amd64)
- 64-bit ARM (AArch64)
- EABI ARM (armel)
- Hard Float ABI ARM (armhf)
- 32-bit PC (i386)
- MIPS (little-endian)
- 64-bit MIPS (little-endian)
- POWER Processors
- IBM System z

El núcleo de la distribución Debian GNU/Linux es la sección principal. Constituye con mucho el grueso de los paquetes, y está compuesto sólo por Software libre, conforme con lo que la DFSG (Debian Free Software Guidelines) entiende por Software libre. Esta distribución se puede descargar de la red.

La distribución Debian GNU/Linux es recopilada por el Proyecto Debian, un grupo de miles desarrolladores voluntarios repartidos por todo el mundo y colaborando a través de Internet. No sólo se ocupan de adaptar y empaquetar los programas de la distribución, también de la infraestructura Web, el sistema de control de errores, la "internacionalización"(adaptación a países e idiomas), las listas de correo de Debian GNU/Linux de desarrollo y mantenimiento, y en un sentido amplio, a toda la infraestructura que hace la distribución Debian GNU/Linux posible.

Los desarrolladores de Debian GNU/Linux empaquetan el Software que obtienen de los autores originales ("upstream"), asegurándose de que funciona correctamente con el resto de los programas Debian GNU/Linux. Para ello, hay un conjunto de reglas que todo paquete debe cumplir, el Manual

de Política Debian (Debian Policy Manual). La mayor parte del esfuerzo de empaquetar un determinado programa generalmente consiste en hacerlo compatible con estas normas. Entre los diferentes entornos de escritorio que pueden ser seleccionados al momento de instalar son:

- GNOME
- KDE Plasma
- LXDE
- LXQt
- MATE
- Xfce

Los desarrolladores también gestionan los errores en los programas, intentan solucionarlos (informando de problemas y soluciones a los autores originales), siguen el desarrollo de nuevos programas y construyen todo el Software intermedio necesario para que el sistema Debian GNU/Linux funcione.

Los fallos y los problemas de seguridad se discuten abiertamente, y diariamente se ponen a disposición de los usuarios actualizaciones para las distribuciones estables, para solucionar problemas importantes de forma que los sistemas permanezcan tan seguros y libres de errores como sea posible.

Debian GNU/Linux es único por muchos motivos. Es destacable su dedicación al Software libre, su naturaleza sin ánimo de lucro y su modelo abierto de desarrollo (donde la mayor parte de las discusiones se hacen en listas de correo públicas). El Proyecto Debian está comprometido con el Software libre, como refleja el Contrato Social Debian

El proyecto Debian es un grupo mundial de voluntarios que se esfuerzan por producir una distribución de sistema operativo que esté compuesta enteramente de Software libre. El producto principal del proyecto a la fecha es la distribución de Software Debian GNU/Linux, la cual incluye a Linux como núcleo del sistema operativo, así como miles de aplicaciones pre-empaquetadas.

Debian GNU/Linux es la única distribución que está abierta a las contribuciones de cada desarrollador y usuario que deseen participar con su trabajo. Y es la única distribución relevante de Linux que no es una entidad

comercial. Es el único gran proyecto con una constitución, contrato social, y documento de directrices que organizan el proyecto.

Debian GNU/Linux ha adoptado un gran conjunto de directrices y procedimientos para el empaquetamiento y la distribución de Software para poder alcanzar y mantener altos estándares de calidad. Se producen herramientas, sistemas automáticos y documentación de cada uno de los aspectos claves de Debian GNU/Linux de una forma abierta y visible para poder sostener estos estándares.

Lo que más distingue a Debian GNU/Linux de otras distribuciones GNU/Linux es su sistema de gestión de paquetes. Estas herramientas otorgan al administrador de un sistema Debian GNU/Linux total control sobre los paquetes instalados, incluyendo la capacidad de instalar un sólo paquete o actualizar el sistema por completo. También es posible proteger paquetes individualmente de forma que no se actualicen. Incluso puede indicar al sistema de gestión de paquetes qué programas ha compilado usted mismo y qué dependencias cumplen.

Para proteger su sistema contra "caballos de Troya" y otros programas malévolos, los servidores de Debian GNU/Linux verifican que los paquetes provienen de sus auténticos encargados. Los desarrolladores de Debian GNU/Linux también ponen gran cuidado en configurarlos de forma segura. Se publican parches muy rápidamente si se descubren problemas de seguridad en los paquetes ya distribuidos. Con el sencillo sistema de actualización de Debian GNU/Linux, se puede descargar e instalar parches de seguridad automáticamente a través de Internet.

Razones para Preferir Debian GNU/Linux Es un sistema operativo estable, que proporciona conscientemente Software maduro y proporciona correcciones oportunas. Debian GNU/Linux unifica la administración a través de muchas arquitecturas y nos permite aplicar nuestros conocimientos en todas partes, con independencia de los procesadores que alimentan a sus sistemas. Si algo necesita ser arreglado o mejorado, existen voluntarios a nivel mundial que se dedican a mejorar y dar solución a múltiples errores, otras razones son:

- El mejor sistema de empaquetamiento de Software, esto se logra través del programa base para manejo de paquetes DPKG, el sólido sistema de empaquetamiento de Debian GNU/Linux, se encarga de cuidar que

su sistema no colapse debido a conflictos de Software gracias a su sólida capacidad de empaquetamiento.

- Debian GNU/Linux viene con decenas de miles de paquetes de Software diferentes. Cada bit de éstos es Software libre. Si se tiene Software propietario que corre bajo GNU/Linux o GNU/kFreeBSD, pueden ser usados.

- Debian GNU/Linux sobrepasa a todas las otras distribuciones en lo bien integrados que están sus paquetes. Como todo el Software lo empaqueta un grupo coherente, no sólo puede encontrar todos los paquetes en un mismo sitio sino que puede estar seguro de que se han eliminado todos los problemas al respecto de complejas dependencias. Aunque el formato deb tiene algunas ventajas sobre el rpm, es la integración entre paquetes lo que hace a un sistema Debian GNU/Linux más robusto.

- Todo el Software en la distribución principal es conforme al criterio de las Directrices de Software Libre de Debian (DFSG). Esto significa que usted puede usar libremente este código para estudiarlo o para incorporarlo a un nuevo proyecto de Software libre. También hay una buena cantidad de herramientas y código apropiado para el uso en proyectos propietarios.

- Actualizarse a una nueva versión de Debian GNU/Linux es muy fácil gracias a su sistema de empaquetamiento. Sólo tiene que ejecutar; apt update ; apt dist-upgrade (o aptitude update; aptitude dist-upgrade, según la versión) y puede actualizarse desde un CD en cuestión de minutos o configurar apt para que utilice alguno de los cientos de espejos de Debian GNU/Linux y actualizarse desde la red.

- Actualmente Debian GNU/Linux soporta un impresionante número de arquitecturas CPU: alpha, amd64, armel, hppa, i386, ia64, mips, mipsel, powerpc, s390, sparc, etc. También corre con los Kernels GNU Hurd y FreeBSD además de Linux, y con la utilidad debootstrap es difícil que encuentre un dispositivo que no pueda correr Debian GNU/Linux.

- El sistema de seguimiento de errores de Debian GNU/Linux es público. No se intenta esconder la realidad de que el Software no siempre trabaja de la manera que los usuarios desean. Aconsejamos a los usuarios que

envíen informes de errores y serán notificados de cuándo y cómo el error ha sido solucionado. Este sistema permite que Debian GNU/Linux responda a los problemas rápida y honestamente.

- Existen muchos casos de máquinas que trabajan durante más de un año seguido sin reiniciarse. De la misma forma, hay equipos que tan sólo son reiniciados debido a un fallo en el suministro de corriente o a una actualización del Hardware.

- Otros sistemas operativos pueden ser rápidos en una o dos áreas, pero, estando basado en GNU/Linux o GNU/kFreeBSD, Debian GNU/Linux es ligero y humilde. El Software para Windows se ejecuta bajo GNU/ Linux usando un emulador a veces más rápido que en su ambiente ori-ginal.

- Los controladores para la mayoría del Hardware están escritos por usuarios de GNU/Linux / GNU/kFreeBSD, no por el fabricante. Mientras que ésto puede significar retrasos antes de que el nuevo Hardware sea soportado y la no existencia de soporte para algún Hardware, permite que continúe el soporte mucho después de que el fabricante haya detenido su producción o haya quebrado. La experiencia ha demostrado que los controladores de fuentes abiertas son usualmente mejores que los controladores propietarios.

- Debian GNU/Linux y la comunidad del Software libre son muy sensibles a asegurarse de que los arreglos de problemas de seguridad entren en la distribución rápidamente. Normalmente, los paquetes arreglados se hacen disponibles a los pocos días. La disponibilidad del código fuente permite que la seguridad en Debian GNU/Linux se evalúe de forma abierta, lo que evita que se implementen modelos de seguridad pobres. Además, la mayoría de los proyectos de Software libre tienen sistemas de revisión por terceras partes, que, como primera medida, evitan que se introduzcan en el sistema problemas de seguridad potenciales.

- Muchos desconocen que cualquier cosa enviada por la red puede ser leída por cualquier máquina entre usted y el receptor. Debian GNU/ Linux tiene paquetes del famoso Software GPG (y PGP) que permite enviar correo entre usuarios preservando su privacidad. Además, ssh

permite crear conexiones seguras a otras máquinas que tengan ssh instalado.

Desventajas Si usted es nuevo en Unix/Linux. Obviamente, si usted está dispuesto a saltar a la parte más profunda e invertir mucho tiempo, nadie lo va a detener. Pero usted puede iniciarse con otra distribución. Siempre se puede volver a Debian GNU/Linux cuando se tiene suficiente experiencia adquirida con Linux.

- Debian GNU/Linux es un sistema complejo con posibilidades prácticamente ili-mitadas. Sin embargo, para aprovechar el poder, tiene que invertir tiempo.

- Es cierto que en Debian GNU/Linux no se dispone de algunos paquetes de Software populares. Sin embargo, existen programas para reemplazar la ma-yoría de ellos, diseñados para imitar las mejores características de los programas propietarios, con el valor añadido de ser Software libre.

- La falta de programas de oficina como Word o Excel debería dejar de ser un problema, porque Debian GNU/Linux incluye varias suites de programas de oficina compuestos por entero de Software libre: LibreOffice, Calligra, y programas de oficina GNOME. También dispone de varias suites de programas de oficina propietarias: Applixware (Anyware), Hancom Office, Axene y otros.

- Para aquellos interesados en bases de datos, Debian GNU/Linux se distribuye con dos programas de bases de datos populares: MySQL, MariaDB, PostgreSQL, Sqlite, MongoDB, etc. También existen versiones para GNU/ Linux deSAP DB, Informix, IBM DB2 y otras.

- Están apareciendo gran cantidad de otros paquetes propietarios, al tiempo que más compañías descubren la potencia de GNU/Linux / GNU/kFreeBSD y su gran mercado oculto, con una base de usuarios en rápido crecimiento (Ya que no se pueden usar cifras de ventas para estimar sus usuarios).

- No todo el Hardware está soportado, particularmente, Hardware realmente nuevo, viejo o raro. También el Hardware que depende de

Software de controlador complejo, que el fabricante sólo distribuye para plataformas Windows (por ejemplo los WinModems o las tarjetas WiFi de equipos portátiles). Aun así, en la mayoría de los casos, hay disponible Hardware equivalente que trabaja con Debian GNU/Linux. Algunos dispositivos no están soportados debido a que el vendedor decidió no dejar las especificaciones disponibles. Esta también es un área en la que se está trabajando.

1.4 Software Propietario y Libre

Con el constante aumento de la comercialización de las computadoras y su relativo bajo costo, las computadoras se han convertido en un objeto omnipresente, ya que estas se encuentran en las actividades cotidianas de millones de usuarios, en formas tan diversas como teléfonos celulares, tabletas, computadoras portátiles y de escritorio, etc.

Las computadoras por sí solas tienen poca utilidad, pero su uso en conjunción con el Software adecuado forman un dúo que nos ha permitido tener los avances de los que actualmente disfrutamos. El Software -sistema operativo y los programas de aplicaciones- son los que realmente generan las soluciones al interactuar uno o más paquetes informáticos con los datos del usuario. También, es común que al comprar una computadora, en el costo total, se integre el del sistema operativo, aplicaciones ofimáticas y de antivirus, sean estos usados por el usuario o no; y en la mayoría de los casos no es posible solicitar que no sean incluidos en el costo de la computadora.

Por otro lado, el Software comercial suele quedar obsoleto muy rápido, ya que constantemente se le agregan nuevas funcionalidades al mismo y estas en general son vendidas como versiones independientes de la adquirida originalmente. Esto obliga al usuario -si quiere hacer uso de ellas- a comprar las nuevas versiones del Software para satisfacer sus crecientes necesidades informáticas y la obsolescencia programada. Por lo anterior y dada la creciente complejidad de los paquetes de cómputo y el alto costo de desarrollo de aplicaciones innovadoras, en muchos casos, el costo total del Software que comúnmente los usuarios instalan -y que no necesariamente usan las capacidades avanzadas del programa, por las cuales el Software tiene un alto costo comercial- en su computadora, suele ser más caro que el propio equipo en el que se ejecutan.

1.4.1 Software Propietario

En entornos comerciales, es posible por parte de la empresa, adquirir y mantener actualizado el Software necesario para sus actividades comerciales, pues el costo del mismo se traslada al consumidor final del bien o servicio que la empresa proporcione. En entornos educativos, de instituciones sin fines lucrativos e incluso, en sector gubernamental, no se cuenta con los recursos necesarios para adquirir y mantener actualizado el Software requerido para todas y cada una de las aplicaciones usadas en las computadoras, ya que en general, las licencias de uso del Software propietario son asignadas en forma individual a cada computadora y no es fácilmente transferible a otra computadora.

Dado que existe una gran demanda de programas de cómputo tanto de uso común como especializado por nuestras crecientes necesidades informáticas, y por la gran cantidad de recursos económicos involucrados, existe una gran cantidad de empresas que tratan de satisfacer dichas necesidades, para generar y comercializar, además de proveer la adecuada documentación y opciones de capacitación que permita a las empresas contratar recursos humanos capacitados.

Por otro lado, generalmente se deja la investigación y desarrollo de productos computacionales nuevos o innovadores a grandes empresas o Universidades -que cuenten con la infraestructura y el capital humano- con la capacidad de analizar, diseñar y programar las herramientas que requieran para sus procesos de investigación, enseñanza o desarrollo.

Existen hoy en día, una gran cantidad de paquetes y sistemas operativos comerciales de Software propietario que mediante un pago oneroso, permiten a los usuarios de los mismos ser productivos en todas y cada una de las ramas comerciales que involucra nuestra vida globalizada, pero el licenciamiento del uso de los programas comerciales es en extremo restrictivo en su uso y más en su distribución.

1.4.2 Software Libre

El Software libre son programas de cómputo -sistema operativo, paquetes de uso común y especializados-, desarrollados por usuarios y para usuarios que, entre otras cosas, comparten el código fuente y el programa ejecutable otorgando de esa forma la libertad para estudiar, adaptar y redistribuir a quien así lo requiera, el programa y todos sus derivados.

El Software libre es desarrollado por una creciente y pujante comunidad de programadores, usuarios y empresas desarrolladoras de Software y Hardware que tratan de poner la mayor cantidad de programas a disposición de todos los interesados, de forma que permiten al usuario promedio sacar el mayor provecho del equipo de cómputo que use, sin importar el sistema operativo subyacente (pero es mejor que todo el Software de un equipo incluyendo el sistema operativo sea Software libre).

¿Qué es el Software Libre? La definición exacta y sus diversas variantes se verán en la siguiente sección, pero podemos entender su esencia a través de los documentos de la fundación para el Software libre. El Software libre concierne a la libertad de los usuarios para ejecutar, copiar, distribuir, cambiar y mejorar el Software:

0. La libertad de usar el programa, con cualquier propósito.
1. La libertad de estudiar cómo funciona el programa y modificarlo, adaptándolo a tus necesidades.
2. La libertad de distribuir copias del programa, con lo cual puedes ayudar a tu prójimo.
3. La libertad de mejorar el programa y hacer públicas esas mejoras a los demás, de modo que toda la comunidad se beneficie.

La lista de proyectos de este tipo es realmente impresionante (véase [21], [20] y [18]). Algunos han conseguido un uso y alta calidad, por ejemplo el compilador GCC (véase [23]), el Kernel de Linux (véase [24]) y el sistema operativo Debian GNU/Linux y Android. Mientras que otros proyectos han caído en el olvido, pero de la gran mayoría se tiene copia del código fuente que permitiría a quienes estén interesados en dicho proyecto poder reusarlo y en su caso ampliarlo.

La característica más importante que aparece típicamente en un proyecto de este tipo, es que un conjunto de personas separadas a gran distancia, sean capaces, a través de la Web, de los E-mail y de foros de aunar sus esfuerzos para crear, mejorar y distribuir un producto, de forma que todos ellos se benefician unos de otros. Evidentemente, la mayor parte del peso recae en los desarrolladores, pero también es necesaria una difusión para que los usuarios documenten, encuentren errores, hagan foros de discusión, etc.

Si bien, el Software Libre no es más seguro (en el sentido de invulnerable) que el propietario, la diferencia estriba en que el código fuente en el Software libre está disponible para todos y cualquiera puede aportar una solución, y por lo general al poco tiempo de detectarse una vulnerabilidad (a veces en cuestión de horas) se puede disponer de una solución para la misma. Además, al tener acceso al código fuente se puede detectar fácilmente si alguien introdujo código malicioso a una determinada aplicación.

¿Por qué se Interesan los Autores, Alumnos y Profesores Universitarios en el Software Libre? La ventaja principal es porque bajo el Software libre subyace la idea de compartir conocimiento y favorecer la existencia de nuevas ideas[5]; y ¿qué es investigar y enseñar?, sino crear conocimiento y procurar que los alumnos aprendan e incluso vayan más allá de lo aprendido. Se comparte la idea, que el espíritu del Software libre es similar al que debería reinar en las instituciones educativas:

- Porque así no se condiciona a los estudiantes a usar siempre lo mismo.
- No se fomenta la piratería en los estudiantes y se evita pagar licencias que no son necesarias al existir alternativas gratuitas.
- Es mucho más seguro ya que el Software libre es público y se puede ver qué hace exactamente sin recelos.
- Se ofrece libertad de elección a los estudiantes y profesores al no limitarlos a usar una solución determinada, ampliando sus opciones y permitiendo un mayor aprendizaje.

Concretando estas ideas, profesores e investigadores necesitan herramientas para la investigación y docencia y estas deben tener una calidad mínima y ser fácilmente distribuibles entre los alumnos. En muchos casos las compañías desarrolladoras y distribuidoras de programas de cómputo no han sabido ofrecer sus productos con la flexibilidad adecuada para las labores docentes o, en otros casos, los productos desarrollados no tienen la calidad esperada.

[5] ¿Por qué el Software creado con dinero de los impuestos no se publica como Software Libre?
¡El código pagado por los ciudadanos debería estar disponible para los ciudadanos y el mismo gobierno!

El Software libre es aún joven, pese a las decenas de miles de proyectos actuales (véase [18] y [19]) -en los que se trabaja constantemente en mejorar la parte computacional de los algoritmos involucrados en el proyecto, haciendo y puliendo interfaces gráficas, generando ayuda en línea así como la documentación necesaria para que usuarios noveles y avanzados usen la mayor cantidad de opciones programadas- existen muchas otras necesidades profesionales y de investigación que requieren el desarrollo innovador de programas de cómputo para automatizarlas y hacerlas eficientes. Esto queda plasmado en las decenas de proyectos que a diario son registrados en las páginas especializadas en busca de difusión y apoyo para su proyecto (véase [18] y [19]).

En los últimos años, muchos proyectos han pasado de ser simples programas en línea de comandos a complejas aplicaciones multiplataforma -se ejecutan en distintos sistemas operativos como son Windows, Linux, Unix, Mac OS, Android- con ambientes gráficos multimedia que en muchos casos han superado a sus contrapartes comerciales -por ejemplo los navegadores Web-. Para muestra de este maravilloso avance, tomemos el proyecto del sistema operativo Android, que actualmente se ejecuta en millones de equipos -como celulares, tabletas, electrodomésticos, etc.- y en los cuales se pueden descargar miles de aplicaciones y está soportado por una gran cantidad de usuarios y empresas comerciales como Google, IBM y últimamente Microsoft -que años atrás era acérrima enemiga del Software libre-.

El Software libre ha logrado desplazar a muchos de sus competidores por sus múltiples bondades y bajo costo de desarrollo -es el caso de Windows Phone que fue reemplazado por Android de Google-, al reusar miles de aplicaciones ya existentes que usan Software libre y permitir desarrollar otro tanto de aplicaciones bajo una plataforma que se ejecuta en los más diversos procesadores. Además, el uso de Software libre y su posibilidad de ampliarlo y/o especializarlo según sea necesario, ha permitido crear de forma cada vez más rápida y confiable; para poner a disposición de un gran público programas de uso común, así como especializado que satisfagan las nuevas necesidades de los usuarios.

1.5 La Línea de Comandos

Todo creador comienza su viaje con un conjunto básico de herramientas de buena calidad. Un carpintero puede necesitar reglas, calibres, un par de sierras, unas buenas garlopas, escoplos finos, taladros, mazos y tornillos de

banco. Estas herramientas se elegirán con mimo, estarán hechas paras durar, realizarán trabajos específicos sin mucho solapamiento con otras herramientas y, quizá lo más importante, se sentirá que las manos del carpintero en ciernes son el lugar al que pertenecen.

Comienza entonces el proceso de aprendizaje y adaptación. Cada herramienta tiene su propia personalidad y sus peculiaridades, y necesita su propio manejo especial. Cada una debe afilarse de una manera única o sostenerse de un modo concreto. Con el tiempo, cada una se desgastará en función de su uso, hasta que la empuñadura parezca un molde de las manos del carpintero y la superficie de corte se alinee a la perfección con el ángulo en el que sostiene la herramienta. En este punto, las herramientas se convierten en canales desde el cerebro del carpintero hasta el producto acabado; se han convertido en extensiones de sus manos. Con el tiempo, el carpintero añadirá herramientas nuevas, como engalletadoras, sierras de inglete guiadas por láser, plantillas de cola de pato... todas ellas formas maravillosas de tecnología. Pero puede estar seguro de que el carpintero será más feliz con esas herramientas originales en la mano, sintiendo cómo canta la garlopa al deslizarse por la madera.

Las herramientas amplifican el talento. Cuanto mejores sean sus herramientas y mejor sepa cómo usarlas, más productivo podrá ser. Empiece con un conjunto básico de herramientas aplicables a nivel general. A medida que adquiera experiencia y vaya encontrándose requisitos especiales, añada más al conjunto básico. Al igual que el carpintero, cuente con añadir herramientas a su kit con regularidad. Esté siempre atento a la aparición de formas mejores de hacer las cosas. Si se encuentra en una situación en la que siente que sus herramientas actuales no sirven, tome nota de buscar algo diferente o más potente que le hubiese ayudado.

Deje que la necesidad impulse sus adquisiciones. Muchos programadores nuevos cometen el error de adoptar una sola herramienta potente, como un entorno de desarrollo integrado (IDE, por sus siglas en inglés) y nunca abandonan su acogedora interfaz. Eso es un verdadero error. Tiene que sentirse cómodo más allá de los límites impuestos por un IDE. La única manera de hacerlo es mantener las herramientas básicas afiladas y listas para usar.

En este texto, hablaremos acerca de la investigación de su propia caja de herramientas básica. Como ocurre con cualquier buena charla sobre herramientas, empezaremos echando un vistazo a nuestra materia prima, aquello a lo que vamos a dar forma. Para garantizar que nunca se pierde nada de nuestro valioso trabajo, deberíamos utilizar un sistema de "Control de

versiones", incluso para cosas personales como recetas o notas. Y, puesto que Murphy era en realidad un optimista, al fin y al cabo, no se puede ser un gran programador a menos que desarrolle una gran habilidad en la "Depuración". Necesitará algo de pegamento para unir toda la magia. Por último, vale más tinta pálida que memoria brillante. Mantenga un registro de sus pensamientos y su historial, en un cuadernos de bitácora.

Además, todo carpintero necesita una buena mesa de trabajo, sólida y fiable, para sujetar las piezas en las que trabaja a una altura conveniente mientras les da forma. La mesa de trabajo se convierte en el centro del taller, y el carpintero vuelve a ella una y otra vez mientras la pieza toma forma.

Para un programador que manipula archivos de texto, esa mesa de trabajo es el intérprete de comandos. Desde el Prompt del indicador de comandos, puede invocar su repertorio completo de herramientas, usando Pipes para combinarlos de maneras que sus desarrolladores originales jamás soñaron. Desde el intérprete de comandos, puede iniciar aplicaciones, depuradores, navegadores, editores y servicios. Puede buscar archivos, consultar el estado del sistema y filtrar salidas. Y, al programar el intérprete de comandos, puede crear comandos en macros complejas para actividades que realiza a menudo.

Para los programadores que han crecido con interfaces GUI y entornos de desarrollo integrados, esto podría parecer una posición extrema. Al fin y al cabo, ¿no se puede hacer todo igual de bien apuntando y haciendo clic?

La respuesta sencilla es "no". Las interfaces GUI son maravillosas y pueden ser más rápidas y convenientes para algunas operaciones sencillas. Mover archivos, leer y escribir correos electrónicos y crear y desarrollar un proyecto son cosas que podría interesarle hacer en un entorno gráfico, pero, si hace todo su trabajo usando GUI, está perdiéndose todas las capacidades de su entorno. No podrá automatizar tareas comunes ni aprovechar el máximo potencial de las herramientas a su disposición. Y no podrá combinar sus herramientas para crear macros personalizadas. Un beneficio de las GUI es WYSIWYG (what you see is what you get, lo que ves es lo que obtienes). La desventaja es WYSIAYG (what you see is all you get, lo que ves es todo lo que obtienes).

Los entornos GUI suelen estar limitados a las capacidades que sus diseñadores planearon. Si necesita ir más allá del modelo proporcionado por el diseñador, por lo general no tendrá mucha suerte, y la mayoría de las veces sí necesita ir más allá del modelo. Los programadores pragmáticos no

solo escribimos código, desarrollamos modelos de objetos, escribimos documentación o automatizamos el proceso de construcción; hacemos todas esas cosas. El alcance de una herramienta cualquiera suele verse limitado a las tareas que se espera que realice esa herramienta. Por ejemplo, supongamos que necesita integrar un preprocesador de código (para implementar diseño por contrato o progmas de multiprocesamiento, o algo de ese estilo) en su IDE. A menos que el diseñador del IDE proporcionase de forma explícita enganches para esa capacidad, no podrá hacerlo.

Familiarícese con el intérprete de comandos y verá cómo se dispara su productividad. Si no ha dedicado mucho tiempo a explorar las capacidades del intérprete de comandos en los sistemas que utiliza, esto podría parecer abrumador. No obstante, invierta algo de tiempo en familiarizarse con su intérprete de comandos y pronto todo empezará a tener sentido. Juguetee con el intérprete de comandos y le sorprenderá cuánto le ayuda a ser más productivo.

Dedique tiempo a aprender a utilizar estas herramientas y, en algún momento, se sorprenderá al descubrir sus dedos moviéndose sobre el teclado, manipulando texto sin un pensamiento consciente. Las herramientas se habrán convertido en una extensión de sus manos.

La Línea de Comandos El sistema Linux básico es un entorno estándar para aplicaciones y programación de los usuarios -generalmente en modo de consola no gráfico-, pero no obliga a adoptar mecanismos estándar para controlar las funcionalidades disponibles como un todo (como Windows o Mac). A medida que Linux ha madurado, se ha hecho necesaria otra capa de funcionalidad encima del sistema Linux. Una distribución de GNU/Linux incluye todos los componentes estándar del sistema Linux -modo consola y gráfico-, más un conjunto de herramientas administrativas que simplifican la instalación inicial y desins-talación de paquetes del sistema.

GNU/Linux puede funcionar tanto en entorno gráfico como en modo consola (línea de comandos o *Shell*). La consola es común en distribuciones para servidores, mientras que la interfaz gráfica está orientada al usuario final del hogar como el empresarial. Así mismo, también existen los entornos de escritorio (GNOME, KDE, LXQt, LXDE, Xfce, Unity, MATE, Cinnamon, Pantheon, Deepin, Budgie, PIXEL, Enlightenment, Trinity, Moksha, Ukui, etc.), que son un conjunto de programas conformado por ventanas, íconos y muchas aplicaciones que facilitan el uso de la computadora.

La mayoría de los usuarios de ordenadores de hoy sólo están familiarizados con la interfaz gráfica de usuario o GUI (del inglés Graphical User Interface) y en los últimos años se han puesto muy de moda las WUI (del inglés Web User Interface) muy similar a las GUI, pero a las que se accede a través de un servidor Web.

Los vendedores y los expertos les han enseñado a los usuarios noveles que la interfaz de línea de comandos o CLI (del inglés Command Line Interface) y su complemento las TUI (del inglés Text User Interface) que juntas son interfaces de texto poco amigables, con menús y ayuda por pantalla es una cosa espantosa del pasado. Es una pena, porque una buena interfaz de línea de comandos es una maravillosa y práctica forma de comunicarse con el ordenador, muy parecida a lo que el lenguaje escrito es para los seres humanos. Se ha dicho que "las interfaces gráficas de usuario hacen fáciles las tareas fáciles, mientras que las interfaces de línea de comandos hacen posibles las tareas difíciles" y eso es muy cierto aún hoy.

Cabe mencionar que Unix/Linux son los únicos sistemas operativos que quedan cuya interfaz gráfica (un montón de código llamado X Windows System[6]) está separado del sistema operativo[7], es decir, se puede ejecutar Unix/Linux en puro modo de línea de comandos si quieres, sin ventanas, iconos, ratones, etc., y seguir siendo Unix/Linux y capaz de hacer todo lo que se supone que hace Unix/Linux. Pero los demás sistemas operativos tienen sus GUI enmarañadas con las funciones del sistema operativo en tal grado que han de ejecutarse en modo GUI o no se ejecutarán, en estos casos no es posible pensar en las GUI como algo distinto del sistema operativo; ahora forman parte inalienable de los sistemas operativos a los que pertenecen -y son, con mucho, la parte mayor, más cara y difícil de crear-.

Dado que Linux fue desarrollado desde la familia de sistemas operativos Unix, comparte la misma rica herencia de herramientas de línea de comandos que Unix. Unix saltó a la fama a principios de los años ochenta -aunque fue desarrollado una década antes-, antes de que se extendiera la adopción

[6]Se trata de un potente sistema de ventanas basado en una arquitectura cliente/servidor. Una de sus ventajas de está arquitectura es que puede ser implementada tanto de manera distribuida (es decir, aplicaciones y servidor gráfico ejecutándose en máquinas diferentes) como local (todo el subsistema gráfico ejecutándose en el mismo equipo de cómputo).

[7]Existen distribuciones de GNU/Linux completas contenidas en solo unas decenas de megabytes. En ellas se usan las últimas versiones del Kernel de Linux y cuentan con los paquetes necesarios para hacer tareas especializadas.

de las interfaces gráficas de usuario, y por eso, se desarrolló una amplia interfaz de línea de comandos en su lugar. De hecho, una de las razones más potentes para que los primeros que utilizaron Linux lo eligieron sobre, digamos, Windows NT, era la poderosa interfaz de línea de comandos que hacía las "tareas difíciles posibles" y eso es lo que trataremos de mostrar a lo largo de este trabajo.

¿Por qué Aprender a Programar en la Línea de Comandos? hay muchas razones por las que deberías aprender sobre la línea de comandos de Linux/Unix. Algunas de estas son:

- Más control sobre tu máquina: tienes mucho poder y control con la línea de comando. Puede ejecutar comandos para cambiar permisos, ver archivos ocultos, interactuar con bases de datos, iniciar servidores y más.
- Es más rápido: puede completar tareas mucho más rápidamente con los comandos básicos de su caja de herramientas que con una interfaz gráfica de usuario (GUI). Solo tenga en cuenta que puede ser más lento mientras aprende la CLI.
- Automatice muchas tareas: puede acelerar su trabajo utilizando un solo comando para crear 10.000 archivos, cada uno con un nombre único. Con una GUI, este proceso es laborioso.
- Disponible en todas partes: las instrucciones que emita se ejecutarán automáticamente de manera similar en computadoras Linux y Mac. Y con algunos ajustes, también funcionarán en Windows.
- Requisito básico: necesitas utilizar la línea de comandos si desea mejorar sus conocimientos en cualquier campo tecnológico relacionado con la codificación, incluido el desarrollo, el análisis de datos, la ingeniería de desarrollo, la administración de sistemas, la seguridad, la ingeniería de aprendizaje automático y otros.

¿Qué es una Shell? un Shell es una interfaz de computadora para un sistema operativo. El Shell expone los servicios del sistema operativo a los usuarios u otros programas. El Shell toma sus comandos y se los da al sistema operativo para que pueda ejecutarlos.

Se llama Shell porque es la capa exterior que rodea el sistema operativo, ¡como la concha alrededor de una ostra!

¿Qué es la Terminal? una terminal es un programa que ejecuta un Shell. Aquí es donde ejecutamos la mayoría de nuestros comandos que le indican al sistema operativo qué hacer.

La terminal se instala de las siguientes maneras en diferentes sistemas operativos:

- Usuarios de alguna distribución de Linux: el Shell Bash está instalado de forma predeterminada
- Usuarios de Mac: la terminal se instala de forma predeterminada y puede ejecutar comandos similares de Linux en Unix
- Usuarios de Windows: descargue el Subsistema de Windows para Linux (WSL) o use git bash y ejecute todos los comandos de Linux desde allí.

Ventajas y Desventajas de Usar la Línea de Comandos Algunas de las ventajas y desventajas al usar la línea de comandos o terminal son las que citaremos a continuación:

Facilidad de Uso Aprender a usar la línea de comandos tiene una curva de aprendizaje más alta que usar interfaz gráfica. Usar la terminal requiere:

- Tener la capacidad y paciencia de memorizar comandos.
- Tener curiosidad y ganas de aprender.
- Ser paciente para leer la documentación de las páginas man.
- A ser posible tener nociones básicas de programación.
- Tener la suficiente paciencia para irse familiarizando con el uso de terminal.

En contraposición las aplicaciones con interfaz gráfica tienen una curva de aprendizaje mucho menor que las que se ejecutan en la línea de comandos. Esto es así por los siguientes motivos:

- Proporcionan un entorno visual agradable. Si una cosa es bonita por fuera atrae a los usuarios.
- Acostumbran a ser intuitivas. Las interfaces gráficas tienen iconos, colores e imágenes que normalmente hacen que la curva de aprendizaje de los usuarios sea mucho menor. En este caso es importante recordar que la mayoría de seres humanos tienden a memorizar mejor los conceptos gráficos que los que están en formato texto.

Por las razones citadas en este apartado los usuarios noveles prefieren usar interfaces gráficas antes que la línea de comandos. No obstante es importante remarcar lo siguiente. Es verdad que tardaremos más tiempo en aprender los comandos para usar la línea de comandos, pero una vez aprendidos podremos abrir una terminal en cualquier ordenador y trabajar de forma habitual. En cambio las GUI acostumbran a variar mucho entre distintas versiones de programas.

Automatización de Tareas mediante Scripts Por norma general las interfaces gráficas no permiten la automatización de tareas repetitivas. Un claro ejemplo de lo que acabo de comentar es el redimensionado de fotografías. Si usaramos un programa con interfaz gráfica y quisiéramos redimensionar 1000 fotografías de un directorio a un 50% es altamente probable que tengamos que repetir 1000 veces la misma operación.

En cambio si usamos la línea de comandos lo podríamos hacer en cuestión de segundos ejecutando un comando similar al siguiente:

```
$ for file in *.png; do convert $file -resize 50% resized-$file;
done
```

Notemos que, cuando se ejecutan comandos en GNU/Linux, ya sea uno a la vez en la línea de comandos o desde un Script de Bash, los comandos se ejecutan en secuencia (usando solo un core de nuestro procesador). El primer comando se ejecuta, seguido por el segundo, seguido por el tercero. Es cierto, el tiempo entre los comandos es tan minúsculo, que el ojo humano no se daría cuenta. Pero para algunos casos, puede que no sea el medio más eficiente para ejecutar comandos.

Con GNU Parallel[8] podemos hacer uso de todos los cores de nuestro procesador, por ejemplo, para cambiar el formato de los .jpg a .png podríamos hacer lo siguiente:

```
$ find .  -name "*.jpg" | parallel -I% –max-args 1 convert %
%.png
```

o

```
$ ls -l *.jpg | parallel convert '{}' '{.}.png'
```

Basándome en los últimos ejemplos recomendaría el uso de la línea de comandos a la totalidad de usuarios que tengan en mente la productividad personal. Tengamos en cuenta que toda tarea/operación realizada en la línea de comandos se podrá automatizar mediante Scripts. Cuantas más operaciones consigamos automatizar más tiempo ahorraremos. Por lo tanto una de las ventajas que proporciona la línea de comandos es la automatización de tareas.

Consumo de Recursos y Rendimiento Las aplicaciones que se ejecutan en la línea de comandos consumen menos recursos y tienen un rendimiento superior. Esto es así porque las aplicaciones ejecutadas en la terminal:

- No requieren de un Driver gráfico avanzado.
- No usan iconos ni imágenes.
- No tienen que cargar ningún tipo de fuente.
- Tienen un consumo de memoria menor.
- Tienen un consumo de CPU menor.
- Requieren una capacidad de almacenamiento en disco duro menor. Las aplicaciones que se ejecutan en la terminal prácticamente no ocupan espacio en nuestro disco duro.

[8] GNU Parallel se puede instalar en casi cualquier distribución de GNU/Linux. Dado que GNU Parallel se encuentra en el repositorio estándar, se instala mediante:

```
# apt install parallel
```

- Interactúan de forma mucho más directa con el sistema operativo.
- etc.

Otra de las ventajas de la línea de comandos será que el consumo de recursos será mucho menor que si usamos una interfaz gráfica. Consecuentemente el rendimiento y la velocidad con que se ejecutarán las tareas/operaciones será mejor en la terminal.

Velocidad de Ejecución y Uso de los Programas Con la línea de comandos pueden satisfacer el 100% de sus necesidades mediante el uso del teclado. En cambio con las interfaces gráficas tendrán que ir moviendo el ratón e ir seleccionado en las opciones pertinentes. Por lo tanto ejecutar acciones en la terminal será más rápido que con una interfaz gráfica. Esto es así porque en la mayoría de ocasiones es mucho más rápido usar el teclado que ir moviendo el ratón e ir clicando encima de los iconos y opciones pertinentes. No obstante para ejecutar tareas de forma rápida en la terminal tendremos que ser capaces de:

- Recordar comandos.
- Recordar los atajos de teclado para poder interactuar con las aplicaciones que ejecutamos en la terminal.
- Acceder de forma rápida a los comandos almacenados en el historial de nuestra Shell.
- Conocer trucos y los atajos de teclado para ejecutar de forma rápida comandos que se han ejecutado en el pasado.
- Tener conocimientos básicos de programación de Scripts. Los Scripts permiten ejecutar un conjunto de comandos para conseguir un fin de forma rápida y sencilla.

Por lo tanto, si tienen paciencia y dominan los 5 puntos que acabo de citar les aseguro que podrán realizar las tareas de forma sensiblemente más rápida en la terminal. No obstante requiere de un tiempo de aprendizaje y de persistencia.

Evitamos Distracciones Mientras usamos un Programa Las aplicaciones que se ejecutan en la terminal facilitan concentrarte en lo que estás haciendo. Las interfaces de terminal acostumbran a:

- Mostrar únicamente las opciones que son necesarias para proseguir al siguiente paso.
- Tener un número reducido de colores e información en pantalla. La interfaz de uso acostumbra a ser limpia.

En cambio las aplicaciones que se ejecutan mediante una interfaz gráfica acostumbran a ser todo lo contrario a lo que acabo de citar. Por lo tanto las aplicaciones que se ejecutan en la línea de comandos evitan distracciones mientras las estamos usando. También nos deberían permitir ser más productivos.

Interacción con el Sistema Operativo La línea de comandos permite interactuar de forma más directa y precisa con nuestro sistema operativo y saber en todo momento lo que está pasando. Por ejemplo si al ejecutar un programa en la terminal se produce un error nos saldrá un mensaje de error que nos dará alguna pista de lo que está pasando. En cambio si el programa se ejecuta a través de una interfaz gráfica no podremos ver el error de forma directa.

Además cuando lanzamos una aplicación en la terminal te da a entender que estás ejecutando un fichero binario junto a una serie de opciones que nosotros podemos modificar. Esta serie de opciones nos dan un mejor control de las acciones que realizamos en todo momento.

Nota de Usar la Terminal en GNU/Linux En ningún momento voy a decir que la línea de comandos es mejor que una interfaz gráfica o viceversa. Simplemente me limité a citar sus ventajas e inconvenientes y después que cada usuario elija lo que crea conveniente. No obstante, de antemano también digo que la terminal no es para todos los usuarios.

Hoy en día la realidad es que la gran mayoría de usuarios prefieren la interfaz gráfica porque de entrada es más amigable. No obstante si eres un usuario que requiere de muchas tareas repetitivas en uno o más equipos de cómputo, un programador o un administrador de sistemas te deberías plantear iniciarte poco a poco en el uso de la terminal.

1.6 Agradecimientos

Este texto es una recopilación de múltiples fuentes, nuestra aportación -si es que podemos llamarla así- es plasmarlo en este documento, en el que tratamos de dar coherencia a nuestra visión de los temas desarrollados.

En la realización de este texto se han revisado -en la mayoría de los casos indicamos la referencia, pero pudimos omitir varias de ellas, por lo cual pedimos una disculpa- múltiples páginas Web, artículos técnicos, libros, entre otros materiales bibliográficos, los más representativos y de libre acceso los ponemos a su disposición en la siguiente liga:

Herramientas
http://132.248.181.216/Herramientas/

Además, la documentación y los diferentes ejemplos que se presentan en este trabajo, se encuentran disponibles en dicha liga, para que puedan ser copiados desde el navegador y ser usados. En aras de que el interesado pueda correr dichos ejemplos y afianzar sus conocimientos, además de que puedan ser usados en diferentes ámbitos a los presentados aquí.

Este proyecto fue posible gracias al apoyo recibido por la Facultad de Ciencias de la Universidad Nacional Autónoma de México (UNAM) y al tiempo robado a nuestras actividades académicas, principalmente durante el período de confinamiento de los años 2020 a 2022. Agradecemos la revisión y aportaciones del Dr. Jorge Luis Ortega Arjona (Facultad de Ciencias, UNAM), Dr. Ed Scott Wilson García (Instituto Mexicano del Petróleo, México) y Antonio Díaz Díaz (Madrid, España).

2 Sistemas Operativos

Actualmente tenemos 3 grandes sistemas operativos en el mercado[9]:

- Windows
- Unix
- GNU[10]/Linux

De los cuales, sus dignos representantes son: Windows, macOS, iOS, Android, Chrome OS y GNU/Linux con todas sus diferentes distribuciones[11]. Y sin temor a equivocarnos aseguramos que Android es la distribución de GNU/Linux más popular e iOS es el más popular de los UNIX.

¿Qué es un Sistema Operativo? El conjunto de programas informáticos que permiten la administración eficaz de los recursos de una computadora es conocido como sistema operativo o Software de sistema. Estos programas comienzan a trabajar apenas se enciende el equipo, ya que gestionan el Hardware desde los niveles más básicos y permiten además la interacción

[9] Cuotas de mercado de diferentes sistemas operativos:

https://gs.statcounter.com/os-market-share/desktop/worldwide
https://netmarketshare.com

[10] GNU -es un acrónimo recursivo de «GNU no es UNIX»- es un sistema operativo de Software libre, es decir, respeta la libertad de los usuarios. El sistema operativo GNU consiste en paquetes de GNU además de Software libre publicado por terceras partes con distintas licencias que conforman una distribución.

[11] Una distribución de Linux es un sistema operativo compuesto por el Kernel de Linux, herramientas GNU, Software adicional y un administrador de paquetes. También puede incluir un servidor de pantalla y un entorno de escritorio que se utilizarán como sistema operativo de escritorio normal. El término es distribución de Linux (o distribución en forma abreviada) porque una entidad como Debian o Ubuntu 'distribuye' el Kernel de Linux junto con todo el Software y las utilidades consideradas por cada entidad como necesarias (como administrador de red, administrador de paquetes, entornos de escritorio, etc.) para que pueda ser utilizado como sistema operativo. Sus distribuciones también asumen la responsabilidad de proporcionar actualizaciones para mantener el Kernel y otras utilidades.

Entonces, Linux es el Kernel, mientras que la distribución de Linux es el sistema operativo. Esta es la razón por la que también se les conoce como sistemas operativos basados en Linux (hay otros Kernels como son FreeBSD, NetBSD y Hurd).

con el usuario. Cabe destacar que los sistemas operativos no funcionan sólo en las computadoras. Por el contrario, este tipo de sistemas se encuentran en la mayoría de los dispositivos electrónicos que utilizan microprocesadores: el Software de sistema posibilita que el dispositivo cumpla con sus funciones -por ejemplo, un teléfono móvil o un reproductor de DVD-.

El sistema operativo cumple con cinco funciones básicas:

- Proporciona la interfaz del usuario -gráfica o de texto-
- La administración de recursos
- La administración de archivos
- La administración de tareas
- El servicio de soporte y utilidades

En cuanto a la interfaz del usuario, el sistema se encarga de que el usuario pueda ejecutar programas, acceder a archivos y realizar otras tareas con la computadora. La administración de recursos permite el control del Hardware, incluyendo los periféricos y la red. El Software de sistema también se encarga de la gestión de archivos, al controlar la creación, la eliminación y el acceso a los mismos, así también, de la administración de las tareas informáticas que ejecutan los usuarios finales. Por último, podemos mencionar que el servicio de soporte se encarga de actualizar las versiones, mejorar la seguridad del sistema, agregar nuevas utilidades, controlar los nuevos periféricos que se agregan a la computadora y corregir los errores del Software.

Tipos de Sistemas Operativos en Función de la Administración de las Tareas Podemos distinguir dos clases de sistemas operativos en función de cómo administran sus tareas, pueden ser:

Sistemas Operativos Monotarea: son sistemas operativos que únicamente cuentan con la capacidad para realizar una tarea al mismo tiempo. Son los sistemas más antiguos, que también llevan aparejados un CPU de menor capacidad. En estos casos, si el equipo está imprimiendo, no atenderá a las nuevas órdenes, ni será capaz de iniciar un nuevo proceso hasta que el anterior haya finalizado.

Sistemas Operativos Multitarea: son los sistemas operativos más modernos, con capacidad para el procesamiento de varias tareas al mismo tiempo. Cuentan con la capacidad para ejecutar varios procesos en uno o más procesadores, por lo que existe la posibilidad de que sean utilizados por varios usuarios al mismo tiempo, y podrían aceptar múltiples conexiones a través de sesiones remotas.

Tipos de Sistemas Operativos en Función de la Administración de los Usuarios También es posible realizar una división de los sistemas operativos en función de la forma en la que se administran los usuarios, como vemos a continuación:

Sistema de Administración Monousuario: sólo pueden gestionar un usuario al mismo tiempo. Así, a pesar de que varios usuarios pueden tener acceso al sistema, solo un usuario puede acceder para realizar y ejecutar operaciones y programas.

Sistemas de Administración Multiusuario: se refiere a todos aquellos sistemas operativos que permiten el empleo de sus procesamientos y servicios al mismo tiempo. Así, el sistema operativo cuenta con la capacidad de satisfacer las necesidades de varios usuarios al mismo tiempo, siendo capaz de gestionar y compartir sus recursos en función del número de usuarios que estén conectados a la vez.

¿Qué Sistema Operativo Usar? ¿Mac o Microsoft? ¿Windows o Linux? ¿Android o iOS? Son preguntas frecuentes que todos nos hemos hecho alguna vez, y es que elegir un sistema operativo, una computadora o un dispositivo móvil no es tan simple. O al menos no lo era años atrás. En la actualidad las diferencias entre sistemas operativos de escritorio son cada vez menos, hasta el punto que prácticamente cualquier servicio Online es compatible con Windows, Mac y GNU/Linux y las principales firmas de Software crean aplicaciones para las tres plataformas principales, salvo excepciones. Lo mismo empieza a ocurrir con el Hardware.

Poco tendremos que decir del sistema operativo de Apple, Mac o iOS (ambos son derivados de Darwin BSD que es un sistema operativo tipo UNIX), ya que son los sistemas operativos más bonitos y que mejores resultados han dado a todos los usuarios que los han probado. Mac es un sistema pensado

para los profesionales de los sectores que necesitan de un equipo de cómputo que sea capaz de todo, como los desarrolladores, programadores, diseñadores, periodistas, fotógrafos, músicos, DJ's y muchos más empleos que se benefician de este sistema operativo.

Después tenemos a Windows, un sistema operativo versátil pensado sobre todo para un uso doméstico, aunque eso no quita que muchas empresas utilicen Windows en sus equipos de cómputo ya que es un sistema operativo que puede dar muy buenos resultados en este aspecto.

Sin embargo, llegamos a Linux, el gran desconocido por muchos. Un sistema operativo mucho más versátil que Windows y que puede ser igual o más profesional que Mac. Sin embargo, la ventaja que tienen estos dos sistemas operativos, es que vienen ya preparados y configurados para el tipo de mercado al que van dirigidos, pero GNU/Linux no.

Esto es una ventaja y una desventaja al mismo tiempo, ya que si tenemos práctica, podemos hacer que el sistema operativo se adapte a nuestras necesidades sin problemas, pero si no tienes práctica, puede que sea demasiado lo que tienes que configurar.

Cuota de Mercado para los Sistemas Operativos Febrero y Agosto son los meses en los que miles de compañías analizan el tráfico que les llega de usuarios a sus páginas Web y desde que plataformas llegan, según un informe de International Data Corporation (https://www.idc.com), Statcounter (https://www.statcounter.com) y The Linux Foundation (https://www.linux-fundation.org) en el último año tenemos:

- En el segmento de los sistemas operativos de escritorio basados en Linux ha subido su cuota de mercado llegando al 4%, esto no parecerá mucho, pero si nos fijamos bien, vemos que Mac tiene un 15% -basado en Unix-, Chrome OS -usa el Kernel de Linux- tiene 2% y Windows el resto.

- En el segmento de teléfonos inteligentes (SmartPhones) y tabletas basadas en Android -usa el Kernel de Linux- tiene 86 % , iOS tiene 13.9 % -basado en Unix- y menos del 1% el resto de los sistemas operativos.

- En el segmento de servidores se estima que más del 60% de los servidores a nivel mundial usan Linux, 1% usan Unix y el resto Windows. Pero en los principales servidores del mundo (un millón) 96 % usan Linux.

- El 90% de toda la infraestructura Cloud corre usando Linux. Es de destacar que en el servicio de servidores Microsoft Azure, el sistema predominante es Linux.
- En el segmento de supercomputadoras, Linux tiene la cuota más importante del mercado; es utilizado en los Top 500 sistemas de supercómputo de alto desempeño del mundo[12].

Hay que decir, que hoy en día y tal y como están las cosas, no existe un sistema operativo que sea definitivo. Así que la pregunta de si GNU/Linux[13] es mejor que Windows o Mac no tiene sentido, ya que cada sistema operativo tiene sus pros y sus contras.

Pero la disyuntiva sigue ahí. ¿Debemos usar Windows en nuestro equipo de cómputo?, ¿nos conviene pasarnos a Linux?. Hay razones a favor y en contra para todos los gustos.

Número de Líneas del Código Fuente de un Sistema Operativo
Pese a que existen múltiples variantes de cada sistema operativo, se han dado a conocer los números de líneas de código fuente que componen la vertiente más usada de algunos sistemas operativos:

- Microsoft Windows 3.1 (Abril de 1992): 3 millones de líneas ($200 USD en 1992)
- Microsoft Windows 95 (Agosto de 1995): 15 millones de líneas (Home $109.95 y Pro$ 209.95 USD en 1995)
- Microsoft Windows NT 4.0 (Julio 1996): 12 millones de líneas (5 usuarios $809, 10 usuarios $1,129 USD en 1996)

[12] Existe el Ranking de las 500 supercomputadoras más poderosas del mundo (esta se actualiza cada seis meses en junio y noviembre) y puede ser consultada en:
https://top500.org
La cuota de supercomputadoras con GNU/Linux ha sido de: 2012 (94%), 2013 (95%), 2014 (97%), 2015 (97.2%), 2016 (99.6%), 2017 (99.6%), 2018 (100%), 2019 (100%), 2020 (100%).

[13] Los resultados de GNU/Linux son muy satisfactorios para los desarrolladores y partícipes de la comunidad Linux, pero todavía hace falta mucho por hacer para que tenga una cuota significativa en el escritorio y esto sólo será posible si los distribuidores de equipo generan un esquema más agresivo para vender máquinas con Linux preinstalado.

- Microsoft Windows 2000 (Febrero 2000): 29 millones de líneas (Pro $319 USD en 2000)
- Microsoft Windows XP (Octubre 2001): 45 millones de líneas (Home $175 US y Pro $ 255 USD en 2001)
- Microsoft Windows Vista (Enero 2007): 50 millones de líneas (Home Premium $239.99 USD en 2007)
- Microsoft Windows 7 (Octubre 2009): 40 millones de líneas (Home Premium $199.99 USD en 2009)
- Microsoft Windows 8 (Octubre 2012): 60 millones de líneas (Home $119.99 US y Pro $ 199.99 USD en 2013)
- Microsoft Windows 10 (Julio 2015): 60 millones de líneas sin Cortana y 65 millones de líneas con Cortana ($139.99 USD en 2019)
- Sun Solaris (Octubre de 1998) 7.5 millones de líneas
- Red Hat Linux 6.2 (Marzo de 2000): 17 millones de líneas
- Red Hat Linux 7.1 (Abril de 2001): 30 millones líneas
- Red Hat Linux 8.0 (Septiembre de 2002): 50 millones de líneas
- Fedora Core 4 (Mayo de 2005): 76 millones de líneas
- Fedora 9 (Mayo del 2008): 205 millones de líneas
- Debian GNU/Linux 3.0 "Woody" (Julio de 2002): 105,000,000 líneas
- Debian GNU/Linux 3.1 "Sarge" (Junio de 2005); 229,500,000 líneas
- Debian GNU/Linux 7 "Wheezy" (Mayo 2013): 419 millones de líneas
- Debian GNU/Linux 10 "Buster" (Julio 2019): 1,077,110,982 líneas
- Debian GNU/Linux 11 "Bullseye" (Agosto 2021): 1,152,960,944 líneas
- Debian GNU/Linux 12 "bookworm" (Junio 2023): 1,341,564,204 líneas
- Kernel Linux 0.01 (Septiembre 1991): 8,413 líneas

- Kernel Linux 1.0 (Marzo 1994): 176,250 líneas
- Kernel Linux 2.6 (Diciembre 2003): 5,475,685 líneas
- Kernel Linux 4.12 (Julio 2017): 24 millones líneas
- Kernel Linux 5.8 (Agosto 2020): 28 millones líneas
- Kernel Linux 5.14 (Julio 2021): 29.7 millones de líneas

El Kernel o Núcleo Es un componente fundamental de cualquier sistema operativo. Es el encargado de que el Software y el Hardware de cualquier equipo de cómputo puedan trabajar juntos en un mismo sistema, para lo cual administra la memoria de los programas y procesos ejecutados, el tiempo de procesador que utilizan los programas, o se encarga de permitir el acceso y el correcto funcionamiento de periféricos y otros elementos físicos del equipo.

Kernel de Linux el núcleo del sistema operativo Linux/Unix (llamado Kernel) es un programa escrito casi en su totalidad en lenguaje C, con excepción de una parte del manejo de interrupciones, expresada en el lenguaje ensamblador del procesador en el que opera, el Kernel reside permanentemente en memoria y alguna parte de él está ejecutándose en todo momento. Pero es muy común confundir al Kernel de Linux con una distribución como Debian y Ubuntu, Linux solo es un núcleo (hay otros como son FreeBSD, NetBSD y Hurd).

Durante mucho tiempo el núcleo Linux solo funcionaba en la serie de máquinas x86 de Intel, desde el 386 en adelante. Sin embargo, hoy día esto ya no es cierto. El núcleo Linux se ha adaptado a una larga y creciente lista de arquitecturas. Siguiendo esos pasos, la distribución Debian GNU/Linux se ha adaptado a estas plataformas. En general este proceso tiene un comienzo difícil (hay que conseguir que la *libc* y el enlazador dinámico funcionen sin trabas), luego sigue un trabajo relativamente largo y rutinario, de conseguir recompilar todos los paquetes bajo las nuevas arquitecturas.

Debian GNU/Linux es un sistema operativo, no un núcleo (en realidad es más que un SO, ya que incluye miles de aplicaciones). Para probar esta afirmación, aun cuando la mayor parte de adaptaciones se hacen sobre núcleos Linux, también existen adaptaciones basadas en los núcleos FreeBSD, NetBSD y Hurd.

Linux es multiprogramado, dispone de memoria virtual, gestión de memoria, conectividad en red y permite bibliotecas compartidas. Linux es multiplataforma y es portable a cualquier arquitectura siempre y cuando está disponga de una versión de GCC compatible.

La parte de un sistema operativo que se ejecuta sin privilegios o en espacio de usuario es la biblioteca del lenguaje C, que provee el entorno de tiempo de ejecución, y una serie de programas o herramientas que permiten la administración y uso del núcleo y proveer servicios al resto de programas en espacio de usuario, formando junto con el núcleo el sistema operativo.

En un sistema con núcleo monolítico como Linux la biblioteca de lenguaje C (*libc*) consiste en una abstracción de acceso al núcleo. Algunas bibliotecas como la biblioteca de GNU proveen funcionalidad adicional para facilitar la vida del programador y usuario o mejorar el rendimiento de los programas. En un sistema con micronúcleo la biblioteca de lenguaje C puede gestionar sistemas de archivos o controladores además del acceso al núcleo del sistema.

A los sistemas operativos que llevan Linux se les llama de forma genérica distribuciones Linux. Estas consisten en una recopilación de software que incluye el núcleo Linux y el resto de programas necesarios para completar un sistema operativo. Las distribuciones más comunes son de hecho distribuciones GNU/Linux o distribuciones Android. El hecho de que compartan núcleo no significa que sean compatibles entre sí. Una aplicación hecha para GNU/Linux no es compatible con Android sin la labor adicional necesaria para que sea multiplataforma.

Las distribuciones GNU/Linux usan Linux como núcleo junto con el entorno de tiempo de ejecución del Proyecto GNU y una serie de programas y herramientas del mismo que garantizan un sistema funcional mínimo. La mayoría de distribuciones GNU/Linux incluye software adicional como entornos gráficos o navegadores Web así como los programas necesarios para permitirse instalar a sí mismas. Los programas de instalación son aportados por el desarrollador de la distribución. Se les conoce como gestores de paquetes. Los creadores de una distribución también se pueden encargar de añadir configuraciones iniciales de los distintos programas incluidos en la distribución.

Las distribuciones Android incluyen el núcleo Linux junto con el entorno de ejecución y herramientas del proyecto AOSP de Google. Cada fabricante de teléfonos dispone de su propia distribución de Android a la cual modifica, elimina o añade programas extra: interfaces gráficas, tiendas de aplicaciones y clientes de correo electrónico son algunos ejemplos de programas suscepti-

bles de ser añadidos, modificados o eliminados. Además de las distribuciones de los fabricantes de teléfonos existen grupos de programadores independientes que también desarrollan distribuciones de Android. LineageOS y Replicant son dos ejemplos de distribuciones Android independientes.

Los usuarios de Linux/Unix estamos acostumbrados a hablar y oír hablar sobre su Kernel[14], el cual puede actualizarse y manipularse en cualquier distribución. Sin embargo, en un sistema operativo tan centrado en el usuario y la sencillez como Windows, su Kernel es un gran desconocido.

Kernel de Windows en la década de los noventa Microsoft estaba basando sus sistemas operativos en los Kernel Windows 9x, donde el código básico tenía muchas similitudes con MS-DOS. De hecho necesitaba recurrir a él para poder operar. Paralelamente, Microsoft también estaba desarrollando otra versión de su sistema dirigido a los servidores llamada Windows NT.

Ambas versiones de Windows fueron desarrollándose por separado. Windows NT era más bien una jugada a largo plazo, una tecnología para ir desarrollando para los Windows del mañana, y en el año 2000 dieron un nuevo paso en esa dirección. A la versión 5.0 de NT la llamaron Windows 2000, y se convirtió en un interesante participante en el sector empresarial.

Tras ver la buena acogida que tuvo, Microsoft decidió llevar NT al resto de usuarios para que ambas ramificaciones convergien. Lo hicieron en octubre del 2001 con la versión 5.1 de Windows NT, que llegó al mercado con el nombre de Windows XP. Por lo tanto, esta versión marcó un antes y un después no sólo por su gran impacto en el mercado, sino porque era el principio de la aventura del Kernel Windows NT en el mundo de los usuarios comunes.

Desde ese día, todas las versiones de Windows han estado basadas en este Kernel con más de 20 años de edad. La versión 5.1.2600 fue Windows XP, la 6.0.6002 fue Windows Vista, y la 6.1.7601 Windows 7. Antes hubo otros

[14] En el caso de los sistemas derivados de Unix y Linux el Kernel lo podemos encontrar en el directorio /boot/, este directorio incluye todos los ejecutables y archivos que son necesarios en el proceso de arranque del sistema y deben ser utilizados antes que el Kernel empiece a dar las órdenes de ejecución de los diferentes módulos del sistema, aquí también es donde reside el gestor de arranque.

En algunas distribuciones al usar un gestor de volúmenes lógico (Logical Volume Manager, LVM) se genera un esquema de particiones con el directorio boot en una partición aparte.

Windows Server 2008 y 2003, y después llegaron las versiones de NT 6.2.9200 llamada Windows 8, la 6.3.9600 o Windows 8, la NT 10.0, también conocida como Windows 10 y finalmente Windows 11.

La principal característica del Kernel de Windows NT es que es bastante modular, y está basada en dos capas principales, la de usuario y la de Kernel. El sistema utiliza cada una para diferentes tipos de programa. Por ejemplo, las aplicaciones se ejecutan en el modo usuario, y los componentes principales del sistema operativo en el modo Kernel. Mientras, la mayoría de los Drivers suelen usar el modo Kernel, aunque con excepciones.

Es por eso que se refieren a él como Kernel híbrido, pero sobre todo también porque permite tener subsistemas en el espacio del usuario que se comunicaban con el Kernel a través de un mecanismo de intercomunicación de procesos IPC (Interprocess Communication).

Cuando ejecutas una aplicación, está accede al modo usuario, donde Windows crea un proceso específico para la aplicación. Cada aplicación tiene su dirección virtual privada, ninguna puede alterar los datos que pertenecen a otra y tampoco acceder al espacio virtual del propio sistema operativo. Es por lo tanto el modo que menos privilegios otorga, incluso el acceso al Hardware está limitado, y para pedir los servicios del sistema las aplicaciones tienen que recurrir a la interfaz de programación de aplicaciones API (Application Programming Interface) de Windows.

El modo núcleo en cambio es ese en el que el código que se ejecuta en él tiene acceso directo a todo el Hardware y toda la memoria del equipo. Aquí todo el código comparte un mismo espacio virtual, y puede incluso acceder a los espacios de dirección de todos los procesos del modo usuario. Esto es peligroso, ya que si un Driver en el modo Kernel modifica lo que no debe, podría afectar al funcionamiento de todo el sistema operativo.

Este modo núcleo está formado por servicios Executive, como el controlador de Caché, el gestor de comunicación, gestor de E/S, las llamadas de procedimientos locales, o los gestores de energía y memoria entre otros. Estos a su vez están formados por varios módulos que realizan tareas específicas, controladores de núcleo, un núcleo y una capa de abstracción del Hardware HAL (Hardware Abstraction Layer).

Diferencias entre los Kernel de Linux y Windows La principal diferencia entre el Kernel de los sistemas operativos Windows y el de Linux está en su filosofía. El desarrollado por el equipo de Linus Torvalds es de

código abierto y cualquiera puede usarlo y modificarlo, algo que le sirve para estar presente en múltiples sistemas operativos o distribuciones GNU/Linux. El Kernel de Microsoft[15] en cambio es bastante más cerrado, y está hecho por y para el sistema operativo Windows.

En esencia, en Linux adoptaron los principios de modularidad de Unix y decidieron abrir el código y las discusiones técnicas. Gracias a ello, Linux ha creado una comunidad meritocrática de desarrolladores, una en la que todos pueden colaborar y en la que cada cambio que se sugiere se debate con dureza para desechar las peores ideas y quedarse con las mejores. También se halaga a quienes consiguen mejorar las funcionalidades más veteranas.

Mientras, en Windows no funciona así, los responsables del Kernel no ven con buenos ojos que se hagan propuestas que se desvíen del plan de trabajo, y asegura que hay pocos incentivos para mejorar las funcionalidades existentes que no sean prioritarias.

Esto hace, a ojos de ese antiguo desarrollador, que al dársele mayor importancia a cumplir planes que a aceptar cambios que mejoren la calidad del producto, o al no tener tantos programadores sin experiencia, el Kernel de Windows NT siempre esté un paso por detrás en estabilidad y funcionalidades.

A nivel técnico existen similitudes entre ambos. Los dos núcleos controlan el Software del sistema de bajo nivel y las interacciones con el Hardware del ordenador a través de la capa de abstracción de Hardware (HAL). El HAL es un elemento del sistema que funciona como interfaz entre Software y Hardware, y como las API, permite que las aplicaciones sean independientes del Hardware.

Los dos están escritos principalmente en C, y son capaces de manejar

[15] Para conocer la información del Kernel de Windows usando la línea de comandos podemos utilizar el siguiente comando en un cmd shell:

```
systeminfo | findstr /B /C:"OS Name" /C:"OS Version"
```

Y en powershell:

```
Get-CimInstance Win32_OperatingSystem | Select-Object Caption, CS-
DVersion, ServicePackMajorVersion, BuildNumber | FL
```

o

```
[System.Environment]::OSVersion.Version
```

el almacenamiento en Caché, los controladores de dispositivos, la memoria virtual, los sistemas de archivos, los protocolos de red y las llamadas de sistema. En esencia sus funcionalidades son las mismas, aunque la manera de llevarlas a cabo es diferente.

Así como el Kernel de Windows tiene dos modos, y por lo tanto se le considera híbrido, la gran diferencia es que el de Linux sólo tiene una capa, o sea que es un núcleo monolítico. Eso sí, pese a ser más sencillo en este aspecto, para funcionar correctamente tiene su núcleo dividido en tres subcapas diferentes.

Ambos gestionan los problemas de memoria de forma parecida. Tienen sistemas de "Swapping" para mover un proceso o parte de él temporalmente de la memoria principal a una secundaria de almacenamiento en el caso de que en la principal haya poco espacio. Windows lo hace en los ficheros Pagefile.sys y Swapfile.sys, mientras que Linux lo suele hacer en una partición, aunque también lo puede hacer en uno o varios ficheros o deshabilitarlo.

Por lo tanto, podemos decir que la principal diferencia entre ambos es la manera en que se desarrolla cada uno. Además, el Kernel de Linux es mucho más sencillo, lo cual es bueno para los desarrolladores. Mientras, el de Windows intenta poner una capa de protección en su modo usuario para que los usuarios con menos conocimientos tengan menos posibilidades de dañar el sistema, y su estructura lo hace más estable frente, por ejemplo a fallos del Driver gráfico.

Pero todo esto ya está cambiando, en las últimas versiones de Windows 10 y 11, Microsoft está integrando el Kernel de Linux a su propio Kernel y esto ha permitido usar Linux dentro de Windows de forma nativa gracias al llamado Windows Subsystem for Linux (WSL, WSL2 WSLg), lo cual ha permitido mejorar la estabilidad y desempeño de Windows.

Kernel de Android Durante la última conferencia de Linux Plumbers 2021, Google dio a conocer sobre el éxito de la iniciativa de mover la plataforma Android para usar un Kernel normal de Linux en lugar de usar su propia versión del Kernel, que incluye cambios específicos para la plataforma Android.

Google menciona que dicho cambio de desarrollo es debido a la decisión de pasar después del año 2023 al modelo «Upstream First», que implica el desarrollo de todas las funciones nuevas del Kernel requeridas en la plataforma Android directamente en el Kernel principal de Linux y no en sus ramas separadas (la funcionalidad será primero se promocionará al Kernel principal

y luego se usará en Android, y no al revés).

Para 2023 y 2024, también se planea transferir al núcleo principal de todos los parches adicionales que quedan en la rama del Kernel común de Android.

En cuanto a un futuro próximo, para la plataforma Android 12 prevista para principios de octubre, se ofrecerán compilaciones del Kernel «Generic Kernel Image» (GKI), lo más parecido posible al Kernel 5.10 habitual.

Para estas compilaciones se proporcionará un lanzamiento regular de actualizaciones, que se colocarán en el repositorio ci.android.com. En el Kernel de GKI, las adiciones específicas de Android, así como los controladores relacionados con el Hardware de los fabricantes de equipos originales, se mueven a módulos de Kernel separados.

Esta nueva interfaz, conocida como Kernel Module Kjos, garantizará que la principal diferencia entre la imagen genérica del Kernel de Android (GKI) y la línea principal de Linux, sean solo los ganchos para todos los módulos específicos del proveedor.

Estos módulos no están vinculados a la versión principal del Kernel y se pueden desarrollar por separado, lo que simplifica enormemente el mantenimiento y la transferencia de dispositivos a nuevas ramas del Kernel. Las interfaces necesarias para los fabricantes de dispositivos se implementan en forma de ganchos que le permiten cambiar el comportamiento del Kernel sin realizar cambios en el código.

En total, el Kernel Android 12-5.10 ofrece 194 ganchos comunes, similares a los puntos de seguimiento, y 107 ganchos especializados que le permiten ejecutar controladores en un contexto no atómico. En el Kernel de GKI, los fabricantes de Hardware tienen prohibido aplicar parches específicos al Kernel principal, y los proveedores deben suministrar los componentes para el Hardware de soporte sólo en forma de módulos de Kernel adicionales, en los que se debe garantizar la compatibilidad con el Kernel principal.

Debemos recordar que la plataforma Android desarrolla su propia rama del Kernel: el «Android Common Kernel», sobre la base del cual se forman las compilaciones específicas separadas para cada dispositivo.

Con lo cual, a partir de cada rama de Android, se proporciona a los fabricantes múltiples diseños de Kernel para sus dispositivos. Por ejemplo, Android 11 ofreció una opción de tres núcleos base a la vez: 4.14, 4.19 y 5.4, y para Android 12, se ofrecerán los núcleos base 4.19, 5.4 y 5.10. La variante 5.10 está diseñada como una imagen de Kernel genérica, en la que las capacidades necesarias para los OEM se transfieren al flujo ascendente, se mueven a módulos o se transfieren al Kernel común de Android.

Antes de la llegada de GKI, el Kernel de Android pasó por varias etapas de preparación:

- La primera de ellas era sobre la base de los principales Kernels LTS (3.18, 4.4, 4.9, 4.14, 4.19, 5.4) y de los cuales se creó una bifurcación del «Android Common Kernel», al que se transferían parches específicos para Android (anteriormente, se alcanzaba el tamaño de los cambios varios millones de líneas).
- Después de ello sobre «Android Common Kernel», los fabricantes de Chips como Qualcomm, Samsung y MediaTek forman el SoC Kernel, que incluye complementos para admitir Hardware.
- Finalmente en el «Kernel de SoC», los fabricantes de dispositivos crean el «Kernel de dispositivo», incluidos los cambios relacionados con la compatibilidad con equipos adicionales, pantallas, cámaras, sistemas de sonido, etc.

Este enfoque complicó significativamente la entrega de actualizaciones con la eliminación de vulnerabilidades y la transición a nuevas ramas del Kernel. Si bien Google publica regularmente actualizaciones para su núcleo común de Android, los proveedores a menudo tardan en enviar estas actualizaciones o usan un solo Kernel durante todo el ciclo de vida del dispositivo, generando una alta fragmentación en el ecosistema Android, una obsolescencia anticipada y en el peor de los casos brechas de seguridad.

Otros Kernels GNU/Linux Actualmente existen una gran cantidad de distribuciones de GNU/Linux que vienen muy optimizadas intentando conseguir la mejor desenvoltura de su arquitectura y configuraciones de serie. En el caso de la configuración por omisión de Debian GNU/Linux y Ubuntu, están pensadas para que sean lo más robusta posible y que se use en todas las circunstancias imaginables, por ello están optimizadas de forma muy conservadora para tener un equilibrio entre eficiencia y consumo de energía. Pero es posible agregar uno o más Kernels GNU/Linux generados por terceros que contenga las optimizaciones necesarias para hacer más eficiente y competitivo en cuestiones de gestión y ahorro de recursos del sistema.

Hay varias opciones del Kernel GNU/Linux optimizado (Liquorix viene optimizado para multimedia y Juegos, por otro lado XanMod tiene uno para propósito general, otro aplicaciones críticas en tiempo real y otro más para

cálculos intensivos) de las últimas versiones estable del Kernel. Estos se pueden instalar[16] mediante el uso de los comandos *dpkg* o *apt* (después de agregarlo a nuestro repositorio */etc/apt/sources.list.d/*), de esta forma siempre podremos tener la última versión del Kernel junto con la actualización básica de nuestro sistema GNU/Linux.

Además, sí instalamos cualquiera de los distintos Kernels, siempre podemos seleccionar alguno de los instalados al momento de arrancar nuestro equipo para usarlo de acuerdo a las actividades requeridas en ese momento. Y en caso necesario, es fácil su desinstalación y continuar usando el Kernel que teníamos por defecto.

Por otro lado, existe una versión completamente libre del Kernel de GNU/Linux (Linux-Libre) el cual es un Kernel despojado de elementos de Firmware y controladores que contienen componentes no libres o fragmentos de código cuyo alcance está limitado por el fabricante.

Linux-libre es el núcleo recomendado por la Free Software Foundation y una pieza principal de las distribuciones GNU totalmente libres de fragmentos privativos o Firmwares incluidos en Linux sirven para inicializar los dispositivos o aplicarles parches que solventan fallas del Hardware que no pudieron ser corregidas antes de ser puestos a disposición de los usuarios.

Además, Linux-libre deshabilita las funciones del Kernel para cargar componentes no libres que no forman parte del suministro del Kernel y elimina la mención del uso de componentes no libres de la documentación. El Kernel de Linux-libre se utiliza en distribuciones como Dragora Linux, Trisquel, Dyne, Bolic, gNewSense, Parabola, Musix y Kongoni.

Estabilidad del Kernel La estabilidad de un núcleo no es tan difícil. El Kernel de Unix de Mac o el Kernel de Linux están diseñados de manera diferente pero resuelven el mismo problema. El sistema operativo Windows es igualmente robusto. Eso es evidente.

Pero la estabilidad real del día a día depende de otros factores. Particularmente en los controladores que conectan el sistema operativo al Hardware. Aquí es donde surgen las diferencias.

Apple tiene el momento más fácil, porque solo admiten un pequeño conjunto de Hardware seleccionado. Eso facilita su trabajo, el objetivo es pequeño. Apple ocasionalmente estropea esto, pero en su mayor parte, OS X

[16] Para ver las opciones de optimización del Kernel y como instalarlo ver la página Web de cada proyecto: Liquorix, XanMod, Linux-Libre.

ofrece una notable estabilidad diaria para las aplicaciones de escritorio.

Windows tiene un trabajo mucho más difícil. El sistema operativo admite una gama simplemente gigantesca de Hardware, y los fabricantes de Hardware hacen todo lo posible para proporcionar controladores de alta calidad. Este es el valor real de Windows, como paquete de controladores, es prácticamente imbatible.

Linux también intenta admitir una gran variedad de Hardware. Pero muchos fabricantes de Hardware son absolutamente indiferentes a la compatibilidad con Linux. Por lo tanto, el soporte de Hardware en Linux es mucho más impredecible. Si está ejecutando un servidor en Linux y el sistema solo se comunica con un disco duro y un adaptador de red, es probable que tenga una estabilidad impecable que supere a la industria.

Pero, si instala Linux en una computadora portátil y espera que funcione con la función de reposo / activación, la GPU, la tarjeta de sonido y un montón de extravagantes periféricos, entonces podría alejarse del rango de la estabilidad. Hay algunos controladores de código abierto, pero estos son significativamente peores que las versiones de los fabricantes. Por ejemplo, una GPU puede ser 4 o 5 veces más lenta.

Como usuario de escritorio de Linux, es probable que también instale aplicaciones que hacen cosas interesantes, de diversos proveedores, que pueden no estar del todo de acuerdo con las mejores prácticas.

Las Vulnerabilidades y Exposiciones Comunes El mundo está cada vez más interconectado y, como resultado de esto, la exposición a las vulnerabilidades de seguridad también ha aumentado dramáticamente. Las complejidades de mantener las plataformas de cómputo actuales hacen que sea muy difícil para los desarrolladores cubrir cada punto de entrada potencial. En 2019 hubo un promedio de más de 45 vulnerabilidades y exposiciones comunes registradas por día y estas siguen en aumento año con año.

Las vulnerabilidades y exposiciones comunes (Common Vulnerabilities and Exposures, CVE https://cve.mitre.org) que tienen los distintos sistemas operativos, es una lista de información registrada sobre vulnerabilidades de seguridad conocidas, en la que cada referencia tiene un número de identificación CVE-ID, descripción de la vulnerabilidad, que versiones del Software están afectadas, posible solución al fallo (si existe) o como configurar para mitigar la vulnerabilidad y referencias a publicaciones o entradas de foros o Blogs donde se ha hecho pública la vulnerabilidad o se demues-

tra su explotación. Además suele también mostrarse un enlace directo a la información de la base de datos de vulnerabilidades (https://nvd.nist.gov, https://openssf.org y https://docs.aws.amazon.com/security), en la que pueden conseguirse más detalles de la vulnerabilidad y su valoración.

El CVE-ID ofrece una nomenclatura estándar para identificación de la vulnerabilidad de forma inequívoca que es usada en la mayoría de repositorios de vulnerabilidades. Es definido y es mantenido por The MITRE Corporation (por eso a veces a la lista se la conoce por el nombre MITRE CVE List) con fondos de la National Cyber Security Division del gobierno de los Estados Unidos de América. Forma parte del llamado Security Content Automation Protocol.

Mitos en torno a Linux/Unix Hay varios mitos en torno a Linux/Unix y al Software libre, a saber:

- Linux/Unix se puede usar para revivir un equipo de cómputo viejo. La realidad es que si bien, hay múltiples distribuciones de Linux/Unix que corren en una gran cantidad de procesadores antiguos y actuales, los Drivers necesarios para reconocer periféricos como tarjetas gráficas, de red alámbrica e inalámbrica, entre muchos otros, no tienen soporte en Linux/Unix, lo cual hará imposible su uso en Linux/Unix. Esto es cierto en cualquier computadora no importa de cuál generación es el equipo de cómputo. La verdad de todo esto, es que los fabricantes están enfocados en producir Hardware y Drivers que corran en los sistemas operativos con mayor cuota de mercado y por el momento Linux/Unix en equipos personales no son de ellos.

- La compatibilidad del Hardware depende en gran medida de la versión de Kernel de GNU/Linux instalado, es de esperarse que en versiones anteriores del Kernel cierto Hardware no se pueda detectar, pero lo contrario también pasa, hay Drivers que solo corren correctamente en versiones anteriores del Kernel y no en las últimas versiones, lo que ocasiona que muchos usuarios se desesperen al tratar de usar sus equipos con GNU/Linux. Y en caso de logar que funcione el Hardware, se fuerza a los usuarios a usar una determinada versión del Kernel (y todas las aplicaciones de la distribución) no actualizable, por la imposibilidad de hacer funcionar el Hardware del equipo en una más moderna con la consiguiente obsolescencia del Software instalado en el equipo.

- Si tengo un Software ahora y quiero ejecutarlo dentro de cinco o diez años en el futuro ¿Por qué no debería ser capaz de hacerlo? Parte de la belleza del Open Source es que el código fuente está disponible, por lo que es más fácil mantener operativo el Software, de modo que no deje de funcionar cuando alguien deja de mantenerlo. Excepto que mantener el Software en Linux/Unix se está convirtiendo en un desafío tan grande que daría igual que fuese privativo. Porque sería complicado hacerlo funcionar en un tiempo razonable, incluso siendo desarrollador, podría costar mucho trabajo y es posible dejar algo sin funcionar en el camino.

- La retrocompatibilidad[17] es un enorme dolor de cabeza, tomar Software hecho para Linux/Unix de hace 10 o 5 años y ejecutarlo en una distribución moderna. Cualquier cosa de mínima complejidad o que use una GUI, simplemente no funciona. Mientras la retrocompatibilidad en Windows es simplemente increíble. En Linux/Unix somos dependientes de los repositorios en línea, y cuando una aplicación depende de ciertas librerías que empiezan a desaparecer de esos repositorios, nos encontramos en una pesadilla. Y mientras más viejo el Software, peor.

2.1 Windows

Microsoft Windows (véase [1]), conocido generalmente como Windows o MS Windows es el nombre de una familia de Software propietario (véase apéndice 14.1) de distribuciones de Software para PC, Smartphone -que perdió cuota de mercado con Android hasta desaparecer-, servidores y sistemas empotrados, desarrollados y vendidos por Microsoft y disponibles para múltiples arquitecturas, tales como x86, x86-64 y ARM.

Desde un punto de vista técnico, no son sistemas operativos, sino que contienen uno (tradicionalmente MS-DOS, o el más actual, cuyo núcleo es Windows NT) junto con una amplia variedad de Software; no obstante, es usual (aunque no necesariamente correcto) denominar al conjunto como sistema operativo en lugar de distribución.

[17] Siempre estamos en posibilidad de usar una Máquina Virtual que nos permite usar un programa desarrollado hace años o décadas en su entorno original, corriendo en un equipo moderno con un sistema operativo de última generación con todas las actualizaciones de seguridad pertinentes.

La versión más reciente de Windows es Windows 11 para equipos personales (que se ofrece como actualización gratuita para los equipos con licencia válida de Windows 10 que cumplan con los requisitos mínimos de Hardware exigidos), Windows Server 2022 para servidores.

Windows 11 tiene al menos siete ediciones con diferente conjunto de características y Hardware previsto, algunas de ellas son: Home, Pro, Pro Education, Pro for Workstations, Enterprise, Education, Mixed Reality. Además habrá un modo SE para hacer frente al Chrome OS de los Chromebooks de Google.

Windows 11 también se puede descargar como una imagen ISO para instalaciones nuevas, ejecución en máquinas virtuales, además de ser la versión de referencia que los fabricantes OEM que usarán para preinstalaciones en equipos nuevos.

Por su parte, Windows 10 tiene al menos doce ediciones con diferente conjunto de características y Hardware previsto, algunas de ellas son: Home, Pro, Enterprise, Enterprise LTBS/LTSC, Education, Mobile, S, Pro for Workstation, Team, Pro Education, loT (Embedded), N y KN. Se espera que cuente con soporte y actualizaciones hasta el 2025.

Todas las ediciones mencionadas tienen la capacidad de utilizar los paquetes de idiomas, lo que permite múltiples idiomas de interfaz de usuario. A pesar de la múltiple cantidad de ediciones, solamente Windows Home y Pro están orientadas para el común de los usuarios y vienen instaladas en equipos nuevos. Las demás ediciones se adquieren mediante otros tipos de compra.

Por su parte, Windows 10 llegó de forma oficial y gratuita a usuarios con licencia genuina de Windows 7, Windows 8.1 y Windows 8 así como a Insiders, siendo la primera versión que buscaba la unificación de dispositivos (escritorio, portátiles, teléfonos inteligentes, tabletas y videoconsolas) bajo una experiencia común, con lo que se esperaba eliminar algunos problemas que se presentaron con Windows 8.1.

Además está Windows PE que es un pequeño sistema operativo usado para instalar, desplegar y reparar Windows 10 en todas sus versiones de escritorio, servidor y para otras ediciones de Windows. En concreto, podemos preparar el disco antes de la instalación, instalar Windows con apps desde un disco local o una red, instalar imágenes de Windows, hacer modificaciones en Windows sin iniciar sesión, recuperar datos perdidos y un largo etcétera.

Seguridad Una de las principales críticas que reciben los sistemas operativos Windows es la debilidad del sistema en lo que a seguridad se refiere y el alto índice de vulnerabilidades críticas. El propio Bill Gates, fundador de Microsoft, ha asegurado en repetidas ocasiones que la seguridad es objetivo primordial para su empresa.

Partiendo de que no existe un sistema completamente libre de errores, las críticas se centran en la lentitud con la que la empresa reacciona ante un problema de seguridad que pueden llegar a meses o incluso años de diferencia desde que se avisa de la vulnerabilidad hasta que se publica la actualización que corrija dicha vulnerabilidad (parche). En algunos casos la falta de respuesta por parte de Microsoft ha provocado que se desarrollen parches que arreglan problemas de seguridad hechos por terceros.

Uno de los pilares en que se basa la seguridad de los productos Windows es la seguridad por ocultación, en general, un aspecto característico del Software propietario que sin embargo parece ser uno de los responsables de la debilidad de este sistema operativo debido a que, la propia seguridad por ocultación, constituye una infracción del uno de los principios de Kerckhoffs, el cual afirma que la seguridad de un sistema reside en su diseño y no en una supuesta ignorancia del diseño por parte del atacante.

Windows 11 Microsoft presentó el 24 de junio del 2021 la siguiente generación de su sistema operativo y que se puede descargar como actualización a partir del 5 de octubre del 2021. Windows 11 con un nuevo diseño que potencia las formas redondeadas y las transparencias, y enfocado a la productividad. Este diseño, que incluye un modo oscuro y uno claro, está pensado para transmitir sensación de calma y sobre todo para que el usuario tenga la información siempre a mano. Esto se aprecia en nuevas funciones, como los Widgets, que pueden personalizar tarjetas de información sobre el tiempo, el tráfico, o información local.

La compañía también ha rediseñado el menú de inicio y la barra de tareas con el acceso a aplicaciones como Teams, que ahora está integrado en Windows. Con Snap Layouts, Windows 11 permite personalizar la apariencia y la disposición de las ventanas. El usuario también podrá personalizar escritorios según el uso, ya sea para el trabajo, para el estudio o el ocio, con configuraciones separadas.

La nueva experiencia con las aplicaciones y son el 'Docking' –cuando el portátil se conecta a un monitor– hacen posible retomar el trabajo, incluso

con varias aplicaciones abiertas, en el mismo lugar donde el usuario lo dejó antes de apartarse del ordenador. Windows 11 también está diseñado para que siempre se sienta igual tanto si se usa en una tableta y su pantalla táctil como si se usa en un ordenador con teclado. Además, mejora la experiencia con el lápiz óptico y teclado táctil en pantalla.

Otro elemento que presenta novedades es Microsoft Store, que ha sido rediseñada para facilitar la búsqueda de aplicaciones, e incorpora también las aplicaciones de Android, que pueden colocarse en la barra de tareas. Con Windows 11 se tendrá una nueva tienda de aplicaciones, una que abrirá sus puertas a todo tipo de aplicaciones y juegos. Esto quiere decir que a partir de ahora la nueva generación de Windows contará con PWA (aplicaciones Web progresivas), UWP (aplicaciones universales) y los programas clásicos Win32 en un mismo lugar.

En cuanto al tema de actualizaciones, Microsoft abandona definitivamente el programa de entrega de actualizaciones semestrales de Windows 10 que definitivamente no pudo concretar con la estabilidad requerida. Windows 11 solo recibirá una actualización anual de características, funciones y soporte de nuevas tecnologías, lo que debería otorgarle tiempo suficiente para desarrollo y pruebas, entregando versiones más pulidas. Las actualizaciones acumulativas de seguridad y corrección de errores si mantendrán el actual ciclo de entrega mensual.

En octubre del 2022 se publicó[18] que Windows 10 se mantiene con una cuota de mercado de 71.29%, mientras que Windows 11 tiene una cuota de 15.44%, esta diferencia deduce por los altos requisitos mínimos de Hardware exigidos -4 GB RAM, TPM 2.0 y 60 GB de Disco-. Por otra parte, las cuotas de mercado de las versiones sin soporte de Windows son: Windows 8.1 de 2.45%, Windows 8 de 0.69%, Windows 7 de 9.61% y Windows XP de 0.39%.

Windows 365 Microsoft presentó en julio del 2021 la nueva versión de Windows en la nube que llevará el nombre de Windows 365. Este nuevo servicio de suscripción, está especialmente dirigido a empresas, que permitirá acceder a nuestra sesión de usuario desde cualquier equipo (el PC, el Mac, la tableta o teléfono Android, etc.), pues el Software y el sistema de

[18] Cuotas de mercado de diferentes sistemas operativos:

https://gs.statcounter.com/os-market-share/desktop/worldwide
https://netmarketshare.com

archivos están alojados en una máquina virtual remota, por lo que la configuración, documentos y herramientas disponibles serán idénticos desde donde accedamos.

Así, desde cualquier navegador (o bien usando la aplicación de Escritorio Remoto de Windows) podremos acceder a un Windows 11/10 disfrutando de una experiencia de arranque casi instantáneo, pero esa no será la única ventaja de esta plataforma. Aún más importante será la posibilidad de contar con varios 'ordenadores' en una misma cuenta, cada uno con distinta potencia (RAM, núcleos de procesador...) y capacidad de almacenamiento contratada, según el trabajo que necesitemos llevar a cabo en cada momento.

Microsoft ya ha confirmado que ofrecerá 12 configuraciones de Hardware distintas para sus equipos virtualizados (iniciando en $130 pesos mexicanos por mes). Así, las empresas podrán 'crear PCs' en cuestión de minutos y asignar cada uno a un empleado, eliminando los inconvenientes que conlleva el hecho de manejar Hardware físico.

Windows 11 SE Microsoft anunció en noviembre del 2021 el lanzamiento de una nueva versión de Windows 11 que hará compañía a las versiones 'Pro' y 'Home': su nombre es Windows 11 SE, llevábamos oyendo rumores al respecto desde junio y desembarca ahora para hacer frente al Chrome OS de los Chromebooks de Google, por lo que se destinará únicamente en portátiles escolares de bajo costo.

El sistema estará optimizado para su uso con MS Edge, MS Office y el resto de servicios de la nube de Microsoft, estará abierto a muchas más aplicaciones de terceros. En palabras de Paige Johnson, directora de marketing educativo de Microsoft, "Windows 11 SE también es compatible con aplicaciones de terceros, incluidas Zoom y Chrome, porque queremos dar a las escuelas la opción de usar lo que funcione mejor para ellas".

Winget Windows Package Manager (*winget*) es el gestor de paquetes de Windows 11 y 10 que permite usar comandos desde la terminal para instalar aplicaciones de forma rápida y sencilla (al más puro estilo de Linux). El único requisito para instalar *winget* es contar con Windows 10 1890 o versión posterior, es posible usar para actualizar una o todas las aplicaciones del sistema usando:

```
C: winget upgrade –all
```

también es posible usar *WInstall* interfaz gráfica no oficial de *winget* para elegir programas en un click desde una Web y luego instalarlos en Windows con *winget* con la misma rapidez y automatización.

Microsoft Open Source En Agosto del 2020 presentó la empresa de Redmond el nuevo sitio Microsoft Open Source en que el público puede navegar a través de todo el ecosistema de código abierto que ha estado construyendo en los últimos años. La Web no solo muestra los proyectos Open Source de Microsoft sino que cuenta con secciones para colaborar con la comunidad, descargar herramientas, explorar su código, y hasta encontrar oportunidades de trabajo.

Las dos partes más importantes de este nuevo sitio son las secciones "Get involved" y "Explore proyects". En la primera se puede revisar toda la actividad reciente en los proyectos Open Source de Microsoft alojados en GitHub, y además se cuenta con una larga lista de recursos para aprender a colaborar con proyectos de código abierto, y no necesariamente solo los que mantiene Microsoft.

La segunda sección es la lista de proyectos, y ahí nos encontramos los principales proyectos Open Source mantenidos por los ingenieros de Microsoft y la comunidad. La lista de proyectos es larga y podemos encontrar los proyectos de los empleados de la empresa patrocinados a través de Microsoft FOSS Fund.

Linux Dentro de Windows Desde el 2018 se inició la integración de GNU/Linux en Windows 10, con la actualización de Windows 10 Fall Creator Update con WSL (Windows Subsystem for Linux), se permitía instalar consolas de diversas distribuciones de GNU/Linux como un programa más. Y en el 2020, con la llegada de Windows 10 Build 2020 con WSL2, el cual cuenta con su propio Kernel de Linux que permite instalar de manera casi nativa diversas distribuciones de GNU/Linux con todo el ambiente gráfico permitiendo tener lo mejor de ambos mundos en un mismo equipo -sin hacer uso de programas de virtualización-, incluso es posible ejecutar varias distribuciones de Linux al mismo tiempo en pantalla.

Para usarlo hay que tener todas las actualizaciones de Windows y activar el Subsistema de Windows para Linux (WSL[19]). Reiniciando el sistema, ya podemos usar distribuciones de Linux desde Microsoft Store.

[19] https://docs.microsoft.com/en-us/windows/wsl/install-win10

En el Windows Insider Preview Build 20150 ha incluido soporte para GPU de Intel, AMD y NVIDIA y es compatible con Direct ML (una API de bajo nivel para aprendizaje automático soportado por DirectX 12) permitiendo el uso de las capacidades de computación por GPU de WSL para Linux.

En abril del 2021 se anunció la llegada de WSLg en la que se pueden correr aplicaciones (como gedit, Audacity, etc.) e IDEs con soporte para X11 y Wayland, PulseAudio de Linux con GUI (Graphical User Interface) con soporte para gráficos 3D acelerados por Hardware de forma independiente de la distribución de Linux en la que se instalaron, esta novedad está disponible a partir de la versión Windows Insider Preview Build 21364 y para todos los usuarios en Windows 10 May 2020 Update.

Android Dentro de Windows En el Windows Build 20185 ha incluido soporte para que Windows 10 permite no sólo sincronizar teléfonos Android, sino además mediante "your Phone" permite integrar las aplicaciones, notificaciones, mensajes, fotos, llamadas y otras opciones de teléfonos inteligentes en Android directamente en Windows, ejecutando las aplicaciones sin tener que abrirlas en el teléfono, aunque siguen proviniendo de ahí.

Además en noviembre de 2020, se ha informado que existe la posibilidad de que Microsoft permita la instalación y ejecución de aplicaciones para Android en Windows 11. Con cambios mínimos o directamente sin modificaciones en el código, los desarrolladores podrían enviar a Microsoft Store sus aplicaciones para que sean descargadas e instaladas en PCs, el proyecto tiene el nombre de Latte.

Microsoft Azure Durante los últimos años, Microsoft ha declarado en conferencias magistrales de eventos y en otros lugares que Linux es un sistema operativo de rápido crecimiento que se usa dentro de Microsoft Azure.

Han declarado con orgullo que el 50 % de las máquinas virtuales nuevas que se ejecutaban en Azure ejecutaban Linux. En Octubre del 2022, las estadísticas reportadas señalan:

- Más del 50 % de los núcleos de máquinas virtuales ejecutan Linux en Azure
- Las imágenes basadas en Linux comprenden el 60 % de las imágenes de Azure Marketplace

- Los 100 principales clientes de Microsoft implementan cargas de trabajo de Linux en Azure
- Azure Tuned Kernels proporciona un rendimiento de red un 25 % más rápido
- Microsoft es compatible con todas las principales distribuciones de Linux, como: Red Hat, SUSE, Ubuntu, Oracle Linux, Debian, CentOS, CoreOS y OpenSUSE (Relacionado: Azure también es compatible con FreeBSD)
- Azure ofrece dos servicios de orquestación de Kubernetes administrados con soporte nativo: Azure Kubernetes Service y Azure Red Hat OpenShift

2.2 UNIX y BSD

Unix (véase [3]) es un sistema operativo portable, multitarea y multiusuario; desarrollado en 1969 por un grupo de empleados de los laboratorios Bell de AT&T. El sistema, junto con todos los derechos fueron vendidos por AT&T a Novell Inc. Esta vendió posteriormente el Software a Santa Cruz Operation en 1995, y está, a su vez, lo revendió a Caldera Software en 2001, empresa que después se convirtió en el grupo SCO. Sin embargo, Novell siempre argumentó que solo vendió los derechos de uso del Software, pero que retuvo el Copyright sobre "UNIX". En 2010, y tras una larga batalla legal, esta ha pasado nuevamente a ser propiedad de Novell.

Solo los sistemas totalmente compatibles y que se encuentran certificados por la especificación Single UNIX Specification pueden ser denominados "UNIX" (otros reciben la denominación «similar a un sistema Unix»). En ocasiones, suele usarse el término "Unix tradicional" para referirse a Unix o a un sistema operativo que cuenta con las características de UNIX Versión 7 o UNIX System V o UNIX versión 6.

Berkeley Software Distribution o BSD (en español, «distribución de Software Berkeley») (véase [4]) fue un sistema operativo derivado de Unix que nace a partir de los aportes realizados a ese sistema por la Universidad de California en Berkeley. En los primeros años del sistema Unix sus creadores, los Laboratorios Bell de la compañía AT&T, autorizaron a la Universidad de Berkeley en California y a otras universidades, a utilizar el código fuente y

adaptarlo a sus necesidades. Durante los años 1970 y 1980 Berkeley utilizó el sistema para sus investigaciones en materia de sistemas operativos.

Cuando AT&T retiró el permiso de uso a la universidad por motivos comerciales, la universidad promovió la creación de una versión inspirada en el sistema Unix utilizando los aportes que ellos habían realizado, permitiendo luego su distribución con fines académicos y al cabo de algún tiempo reduciendo al mínimo las restricciones referente a su copia, distribución o modificación (véase apéndice 14.4).

Algunos sistemas operativos descendientes del sistema desarrollado por Berkeley son SunOS, FreeBSD, NetBSD, OpenBSD, DragonFlyBSD y Mac Big Sur. BSD también ha hecho grandes contribuciones en el campo de los sistemas operativos en general. Además, la licencia permisiva de BSD ha permitido que otros sistemas operativos, tanto libres como propietarios incorporaron código BSD. Por ejemplo, Microsoft Windows ha utilizado código derivado de BSD en su implementación de TCP/IP, y utiliza versiones recompiladas de la línea de comandos BSD para las herramientas de redes. También Darwin, el sistema en el cual está construido Mac Big Sur, el sistema operativo de Apple, está derivado en parte de FreeBSD 5. Otros sistemas basados en Unix comerciales como Solaris también utilizan código BSD.

Algunos proyectos activos descendientes del sistema BSD son:

FreeBSD (https://www.freebsd.org/es/)
Es un sistema operativo para computadoras basadas en las CPU de arquitectura Intel. También funciona con procesadores compatibles como AMD. Está basado en la versión 4.4 BSD-Lite del CSRG (Computer Systems Research Group) y fue escrito en C y C++. Tiene Licencia BSD. Este proyecto ha realizado una gran inversión de tiempo en ajustar el sistema para ofrecer las mejores condiciones de rendimiento con carga real y facilidad de uso al usuario final.

NetBSD (https://www.netbsd.org)
Está basado en un conjunto de aplicaciones Open Source, incluyendo 4.4 BSD-Lite de la Universidad de California en Berkeley, Net/2 (Berkeley Networking Release 2), el sistema gráfico X del MIT y aplicaciones del proyecto GNU. Tiene Licencia BSD. NetBSD ha invertido sus energías en proveer de un sistema operativo estable, multiplataforma, seguro y orientado a la inves-

tigación. Está portado a 56 arquitecturas de Hardware y suele ser el primero en implementar tecnologías nuevas, como IPv6.

OpenBSD (https://www.openbsd.org)
Está basado en 4.4 BSD y es un descendiente de NetBSD. El proyecto tiene el foco puesto de forma particular en la seguridad y criptografía. Los esfuerzos se centran en la portabilidad, cumplimiento de normas, corrección, seguridad y criptografía integrada. Tiene Licencia BSD. La filosofía del proyecto puede ser descrita en tres palabras: "Free, Functional and Secure" (Libre, Funcional y Seguro).

DragonFlyBSD (https://www.dragonflybsd.org)
Tiene como meta ofrecer un alto rendimiento y escalabilidad bajo cualquier entorno, desde computadoras de un solo usuario hasta enormes sistemas de clústeres. DragonFlyBSD tiene varios objetivos técnicos a largo plazo, pero el desarrollo se centra en ofrecer una infraestructura habilitada para SMP que sea fácil de entender, mantener y desarrollar.

2.3 Apple y sus macOS e iOS

Apple a la empresa multinacional estadounidense Apple Inc., dedicada al diseño, la confección y la comercialización de productos electrónicos y de Software, así como de los servicios en línea (a través de Internet) que los atañen. Es considerada como una de las sociedades más apreciables del mundo. Su función principal es la producción de aparatos digitales populares, las marcas más populares de esta compañía son: Mac, iPods, iPads e iPhones. Y cuya gama de productos ha ganado un nicho particular en el área de los Gadgets tecnológicos mediante su estética común, intensa mercadotecnia y pretendida alternatividad respecto a otras empresas hegemónicas como Microsoft.

En la actualidad los principales productos de Apple gozan de una amplia popularidad a nivel mundial, particularmente en lo vinculado con reproductores de música, computadores personales, tabletas, teléfonos, relojes inteligentes, Wearables y toda una red de Software que va desde sistemas operativos, programas de gestión de multimedios (como iTunes) o suites de edición profesional. Se trata de una empresa líder en innovación tecnológica y computarizada.

El sistema de Apple siempre se ha caracterizado por contar con detalles no solo sofisticados, sino también fuera de lo común, tratando de marcar la historia de la tecnología bajo Software con distintas mejoras, asistentes virtuales y muchos elementos que dejan gran satisfacción y que debes conocer.

Realizando un seguimiento de los últimos productos lanzados y presentados por los chicos de la manzanita nos damos cuenta de que sin duda son los mejores creando expectación. En Apple saben que para vender hay que mostrar cosas nuevas e innovar, ya que vivimos en un mundo en el que la tecnología evoluciona constantemente y si no lo haces, te quedas atrás. Pero..., ¿siempre son cosas nuevas las que nos muestran?.

Una de las cosas en las que destaca Apple por encima de sus competidores es en el campo de la investigación y la innovación. Las inversiones que realiza la empresa más valiosa del mundo en investigación son enormes, y no en vano. Porque si algo sabe hacer bien Apple es ir por delante de la competencia. En esto todos estamos de acuerdo, los de Cupertino saben mejor que nadie que innovar significa no tener competencia en el mercado. Sin duda, esto saben aprovecharlo y a veces, de tal manera que ni nos damos cuenta.

Pero en Apple tienen una fea costumbre que analizando un poco las últimas Keynotes nos damos cuenta. Los de Cupertino suelen vender como algo innovador y único cosas que ya hacía antes pero pasaba por alto. ¿Qué significa esto?, que en Apple saben como vender las características de sus productos para que nos parezcan espectaculares.

Las computadoras de Apple tienen un sistema operativo único y otras características que sólo están disponibles en las Macs. El diseño del Hardware de Apple unifica la marca, con productos como el iMac "todo en uno", el iMac original con sus colores brillantes y la gama de computadoras portátiles. Además, las nuevas características del sistema operativo que integran el uso de una computadora Mac, iPad, el teléfono o el iPod con iTunes de Apple y la tienda de aplicaciones, proporcionan a los clientes una experiencia de Apple aerodinámica.

Características de Mac

- Carpetas inteligentes en Finder
- Grabe la actividad de la pantalla macOS en QuickTime
- Activa las esquinas calientes de tu pantalla

- Reproduce música y películas
- Ejecutar Windows con Boot Camp
- Automatiza las tareas repetitivas
- Crea escritorios virtuales
- Ping archivos de forma inalámbrica con AirDrop
- Firmar documentos en Vista previa
- Autocompletar palabras a medida que escribe

Características de iOS

- Notificaciones innovadoras
- Widgets de máxima utilidad
- Puedes borrar las aplicaciones de fábrica
- Siri abre apps de terceros
- Varios idiomas para el teclado
- Reconocimiento facial en sus fotos
- Te permite controlar tu hogar
- 3D touch con muchos usos
- Dispone de mensajería interna: iMessage

Mac OS y macOS Ventura Mac OS (véase [5]) -del inglés Macintosh Operating System, en español Sistema Operativo Macintosh- es el nombre del sistema operativo propietario (véase apéndice 14.1) creado por Apple para su línea de computadoras Macintosh, también aplicado retroactivamente a las versiones anteriores a System 7.6, y que apareció por primera vez en System 7.5.1. Es conocido por haber sido uno de los primeros sistemas dirigidos a un gran público al contar con una interfaz gráfica compuesta por la interacción del Mouse con ventanas, íconos y menús.

Debido a la existencia del sistema operativo en los primeros años de su línea Macintosh resultó a favor de que la máquina fuera más agradable al usuario, diferenciándolo de otros sistemas contemporáneos, como MS-DOS, que eran un desafío técnico. El equipo de desarrollo del Mac OS original incluía a Bill Atkinson, Jef Raskin y Andy Hertzfeld.

Este fue el comienzo del Mac OS clásico, desarrollado íntegramente por Apple, cuya primera versión vio la luz en 1985. Su desarrollo se extendería hasta la versión 9 del sistema, lanzada en 1999. A partir de la versión 10 (Mac OS X), el sistema cambió su arquitectura totalmente y comenzó a basarse en BSD Unix, sin embargo su interfaz gráfica mantiene muchos elementos de las versiones anteriores.

Hay una gran variedad de versiones sobre cómo fue desarrollado el Mac OS original y dónde se originaron las ideas subyacentes. Pese a esto, los documentos históricos prueban la existencia de una relación, en sus inicios, entre el proyecto Macintosh y el proyecto Alto de Xerox PARC. Las contribuciones iniciales del Sketchpad de Ivan Sutherland y el On-Line System de Doug Engelbart también fueron significativas.

Versiones Antes de la introducción de los últimos sistemas basados en el microprocesador PowerPC G3, partes significativas del sistema se almacenaban en la memoria física de sólo lectura de la placa base. El propósito inicial de esto fue evitar el uso de la capacidad de almacenamiento limitada de los disquetes de apoyo al sistema, dado que los primeros equipos Macintosh no tenían disco duro. Sólo el modelo Macintosh Classic de 1991, podía ser iniciado desde la memoria ROM.

Esta arquitectura también permitió una interfaz de sistema operativo totalmente gráfica en el nivel más bajo, sin la necesidad de una consola de sólo texto o el modo de comandos de línea. Los errores en tiempo de arranque, como la búsqueda de unidades de disco que no funcionaban, se comunicaban al usuario de manera gráfica, generalmente con un ícono o con mensajes con el tipo de letra Chicago y un "timbre de la muerte" o una serie de pitidos.

Esto contrastaba con los PCs de la época, que mostraban tales mensajes con un tipo de letra monoespaciada sobre un fondo negro, y que requerían el uso del teclado y no de un ratón, para el acceso. Para proporcionar tales detalles en un nivel bajo, Mac OS dependía del Software de la base del sistema grabado en la ROM de la placa base, lo que más tarde ayudó a garantizar que sólo los equipos de Apple o los clones bajo licencia (con el contenido de la

memoria ROM protegido por derechos de autor de Apple, pudieran ejecutar Mac OS).

Mac OS puede ser dividido en tres familias:

- La familia Mac OS Classic, basada en el código propio de Apple Computer.
- El Sistema Operativo Mac OS X, desarrollado a partir de la familia Mac OS Classic y NeXTSTEP, el cual estaba basado en UNIX.
- MacOS Ventura es el reemplazo de macOS Monterrey (que fue el reemplazo de Mac OS X), disponible a partir de junio del 2022, que usando los procesadores de ARM han mostrado un gran desempeño en comparación con equipos INTEL y AMD de gama alta.

Linux Dentro de IOS Es posible tener un Linux completo en IOS además de poder hacer uso de Secure Shell (SSH) a una computadora con Linux. Para la primera forma, se puede ejecutar un sistema virtualizado utilizando Alpine Linux con iSH, que es de código abierto, pero debe instalarse utilizando la aplicación TestFlight propiedad de Apple.

Alternativamente hay aplicaciones de emulador de terminal de código abierto que proporcionan herramientas de código abierto dentro de un entorno restringido. Esta es la opción más limitada -en realidad no nos permite ejecutar Linux, pero estaremos ejecutando herramientas de Linux- pero brindan algunas funciones de línea de comandos. Por ejemplo:

- Sandboxed Shell, con más de 80 comandos e incluye Python 2 y 3, Lua, C, Clang, etc.
- a-Shell, otorga acceso al sistema de archivos e incluye Lua, Python, Tex, Vim, JavaScript, C yC++, junto con Clang y Clang++; y permite instalar paquetes de Python con pip.
- Blink Shell, permite la conexión con servidores.
- iSH, es un Shell Linux que usa *usermode x86* emulación y traducciones de *syscall*.

2.4 GNU/Linux

GNU[20]/Linux (véase [2]) también conocido como Linux, es un sistema operativo libre (véase apéndice 14.2) tipo Unix; multiplataforma, multiusuario y multitarea. El sistema es la combinación de varios proyectos, entre los cuales destacan GNU (encabezado por Richard Stallman y la Free Software Foundation) y el núcleo Linux (encabezado por Linus Torvalds). Su desarrollo es uno de los ejemplos más prominentes de Software libre: todo su código fuente puede ser utilizado, modificado y redistribuido libremente por cualquiera, bajo los términos de la GPL (Licencia Pública General de GNU) y otra serie de licencias libres.

A pesar de que «Linux» se denomina en la jerga cotidiana al sistema operativo, este es en realidad solo el Kernel (núcleo) del sistema. La idea de hacer un sistema completo se remonta a mediados de la década de 1980 con el proyecto GNU, así como una gran cantidad de los componentes que se usan hoy en día -además del núcleo-, que van desde los compiladores de GNU hasta entornos de escritorio. Sin embargo, tras la aparición de Linux en la década de 1990 una parte significativa de los medios generales y especializados han utilizado el término «Linux» para referirse a todo. Esto ha sido motivo de polémicas. Cabe señalar que existen derivados de Linux que no tienen componentes GNU -por ejemplo Android-, así como distribuciones de GNU donde Linux está ausente -por ejemplo Debian GNU/Hurd-.

A GNU/Linux se le encuentra normalmente en forma de compendios conocidos como distribuciones o distros, a las cuales se les ha adicionado selecciones de aplicaciones y programas para descargar e instalar las mismas. El propósito de una distribución es ofrecer GNU/Linux como un producto final que el usuario pueda instalar, cumpliendo con las necesidades de un grupo de usuarios o bien del público en general. Algunas de ellas son: Ubuntu, CentOS, Debian, Linux Mint, Arch Linux, Fedora, Red Hat, Oracle, Zorin, MX Linux, Parrot, Manjaro, Elementary, etc.

Algunas de ellas son especialmente conocidas por su uso en servidores de internet y supercomputadoras -donde GNU/Linux tiene la cuota más importante del mercado. Según el informe de International Data Corporation (IDC), GNU/Linux es utilizado por los más poderosos 500 sistemas de super-

[20] GNU -es un acrónimo recursivo de «GNU no es UNIX»- es un sistema operativo de Software libre, es decir, respeta la libertad de los usuarios. El sistema operativo GNU consiste en paquetes de GNU además de Software libre publicado por terceras partes con distintas licencias que conforman una distribución.

cómputo de alto desempeño del mundo[21]-, en cuanto a teléfonos inteligentes y tabletas tiene una cuota de 86% y con menor participación, el sistema GNU/Linux también se usa en el segmento de las computadoras de escritorio, portátiles, computadoras de bolsillo, sistemas embebidos, videoconsolas y otros dispositivos.

Creación El proyecto GNU, iniciado en 1983 por Richard Stallman, tiene el objetivo de crear un «sistema de Software compatible con Unix compuesto enteramente de Software libre». El trabajo comenzó en el año 1984. Más tarde, en 1985, Stallman fundó la Free Software Foundation para financiar el desarrollo de GNU, y escribió la Licencia Pública General de GNU en 1989. A principios de la década de 1990, muchos de los programas que se requieren en un sistema operativo -como bibliotecas, compiladores, editores de texto, el Shell Unix, y un sistema de ventanas- ya se encontraban en operación. Sin embargo otros elementos como los controladores de dispositivos y los servicios estaban incompletos.

Linus Torvalds ha declarado que si el núcleo de GNU hubiera estado disponible en el momento (1991), no se habría decidido a escribir su propio núcleo. Aunque no fue liberado hasta 1992 debido a complicaciones legales, el desarrollo de BSD -de los cuales NetBSD, OpenBSD y FreeBSD descienden- es anterior al de Linux. Torvalds también ha declarado que si BSD hubiera estado disponible en ese momento, probablemente no habría creado Linux.

En 1991 Torvalds asistía a la Universidad de Helsinki. Usuario de MINIX y de los programas provenientes de GNU, se mostraba interesado por los sistemas operativos. Comenzó a trabajar en su propio núcleo en ese año, frustrado por la concesión de licencias que utilizaba MINIX, que en ese momento se limitaba a uso educativo.

El núcleo Linux maduró hasta superar a los otros núcleos en desarrollo. Las aplicaciones GNU también reemplazaron todos los componentes de MINIX, porque era ventajoso utilizar el código libre del proyecto GNU con el nuevo sistema operativo. El código GNU con licencia bajo la GPL puede ser reutilizado en otros programas de computadora, siempre y cuando también se liberen bajo la misma licencia o una licencia compatible. Torvalds inició un cambio de su licencia original, que prohibía la redistribución comercial a la GPL. Los desarrolladores de ambas partes trabajaron para integrar com-

[21] Top500.org informó, en su lista de noviembre de 2017 -y así ha continuado hasta ahora-, que las 500 supercomputadoras más potentes del mundo utilizan Linux.

ponentes de GNU con el núcleo Linux, consiguiendo un sistema operativo completamente funcional.

Para darnos una idea del frenético crecimiento del Kernel de Linux, por ejemplo, en la versión 4.10 se añadieron 632,782 líneas de código nuevo y en el Kernel 4.12 se añadieron más 1.2 millones de líneas de código nuevas, teniendo un total de 24,170,860 líneas de código. El número de desarrolladores involucrados fue de 1821 colaboradores y 220 empleados hicieron un promedio de 231 cambios por día, casi 10 cambios por hora, diariamente se añadieron casi 20 mil líneas de código, y casi 800 líneas por hora en dicha versión.

Hay que precisar que, si bien el código alojado en el repositorio del Kernel es cuantioso, sólo una pequeña parte del mismo afectará a nuestras propias instalaciones de GNU/Linux, pues gran parte del código fuente es específico para cada una de las (múltiples) arquitecturas de Hardware compatibles con Linux.

De hecho, a principios de 2018, Greg Kroah-Hartman (responsable de mantenimiento del código), afirmó que "un portátil promedio usa alrededor de 2 millones de líneas de código del Kernel para funcionar correctamente", cuando en aquel momento, el Kernel completo ya contaba con 25 millones de líneas de código (que ya han aumentado a más de 28 millones en la versión 5.8).

GNU/Linux puede funcionar tanto en entorno gráfico como en modo consola. La consola es común en distribuciones para servidores, mientras que la interfaz gráfica está orientada al usuario final del hogar como empresarial. Así mismo, también existen los entornos de escritorio, que son un conjunto de programas conformado por ventanas, íconos y muchas aplicaciones que facilitan el uso de la computadora. Los entornos de escritorio más populares en GNU/Linux son: GNOME, KDE, LXQt, LXDE, Xfce, Unity, MATE, Cinnamon, Pantheon, Deepin, Budgie, PIXEL, Enlightenment, Trinity, Moksha, Ukui, entre muchos otros.

¿Qué es lo que está llevando a la gente a probar distribuciones de GNU/Linux y a utilizarlas como sistema operativo principal en sus equipos de cómputo? A continuación, vamos a exponer una lista con las razones por las que deberías probar una distribución de GNU/Linux -ya que es una sabia elección- como sistema operativo principal en tu equipo de cómputo:

Software Libre y Código Abierto muchos usuarios de internet no conocen el significado principal del Software libre ni del código abierto. Software libre son esos programas que se automanifiestan, por parte de sus autores, que puede ser copiado, modificado y redistribuido con o sin cambios o mejoras. El concepto de código abierto, es el Software desarrollado y distribuido libremente. Tiene beneficios prácticos ya que si alguien tiene una idea o piensa que puede mejorar el código puede modificarlo sin problemas.

Seguridad no descubrimos el agua tibia diciendo que el sistema operativo de Microsoft es el más atacado por virus y Malware y además, se han descubierto varios virus para Mac OS, unos que llevan ocultos mucho tiempo. Pero con GNU/Linux eso no pasa, ya que es un sistema suficientemente seguro y que no tenemos muchos registros de ataques a esta plataforma.

Aunque hay compañías Linuxeras, como Oracle, Novell, Canonical, Red Hat o SUSE, donde el grueso de distribuciones y Software Linux está mantenido por usuarios y colectivos sin ánimo de lucro. A diferencia de Microsoft y Windows, detrás de Linux no es habitual encontrarnos con una empresa con intereses empresariales, de manera que es más fácil evitar problemas de tipo legal o violaciones de nuestra privacidad o seguridad por parte de quienes han programado esa aplicación o versión de GNU/Linux que usamos. Un ejemplo es la recopilación de datos de uso. A diferencia de los sistemas operativos comerciales, en GNU/Linux no es habitual toparse con este problema.

Es Gratis aunque Mac OS X también es gratuito, está pensado para funcionar solamente en equipos de cómputo Apple. En cuanto a Windows, a pesar de la tendencia, sigue siendo de pago, a pesar de las muchas ofertas que hizo para cambiar de Windows 7 a Windows 10.

Si adquieres una computadora nueva con Windows, el precio incluye la licencia de compra. Por otro lado, todo el mundo sabe que los sistemas operativos de GNU/Linux son totalmente gratuitos y puedes instalarlos en cualquier equipo de cómputo. Las distribuciones más populares puedes descargarlas desde sus páginas oficiales e instalarlas las veces que quieras y en el número de equipos de cómputo que necesites. Además, no tendremos que pagar por utilizar el Software, sin embargo, podremos donar lo que nos plazca al proyecto para que sigan mejorándolo.

Fácil de Utilizar muchos de nosotros hemos utilizado un sistema operativo basado en GNU/Linux y no lo sabíamos. Aeropuertos, estaciones de tren, sistemas de gestión empresarial y ahora en el espacio con SpaceX, etc. Muchos de estos sistemas están basados en GNU/Linux.

Una de las barreras que durante años ha evitado a muchos usar Linux es su complejidad. O al menos lo era cuando la mayoría de tareas debías hacerlas desde la línea de comandos.

En la actualidad, distribuciones GNU/Linux como Ubuntu, Mint, Manjaro, Debian u OpenSUSE ofrecen una interfaz similar a Windows y con todas las herramientas y aplicaciones necesarias para empezar a disfrutar desde el primer día.

Si necesitas un nuevo Software, la mayoría de distribuciones cuentan con su propia tienda de aplicaciones o herramienta de gestión de aplicaciones. Todo está pensado para que cualquiera pueda manejarse sin problemas.

Está claro que existen versiones de GNU/Linux complejas, pero están enfocadas a un público muy concreto. Las distribuciones domésticas cumplen con creces con los requisitos de usuarios amateurs o recién llegados.

Versatilidad configurar un sistema a nuestro gusto, en Windows o en Mac OS X, es algo realmente difícil, pero con los sistemas operativos basados en GNU/Linux se puede tener un sistema operativo totalmente único y totalmente personalizable.

La naturaleza de GNU/Linux y su filosofía de código abierto y libre hace posible que contemos con cientos de versiones diferentes. Esto implica que podamos elegir una versión de GNU/Linux, o distribución, en función de para qué la queremos. ¿Para educación? ¿Para niños? ¿Para uso doméstico? ¿Para gestión de redes? ¿Para temas de seguridad? ¿Para reciclar un PC antiguo? Incluso las hay para arreglar problemas de Windows.

Esta variedad significa que no sólo podemos emplear GNU/Linux en una computadora doméstica. Los ejemplos más claros son Raspberry Pi, Jetson Nano, Pine64 y Arduino[22], son soluciones baratas y diminutas para montar tu propia computadora personal, tu centro multimedia o cualquier artilugio electrónico que desees diseñar. Y para hacerlo funcionar, cuentas con varias distribuciones Linux enfocadas a dicho Hardware.

[22] Son ordenadores del tamaño de una tarjeta de crédito que se conectan a un televisor, un teclado y ratón. Es una placa que soporta varios componentes necesarios en un ordenador común y cuyo precio inicial es de 15 dólares.

Usar GNU/Linux significa que puedes cambiar cualquier elemento de tu sistema operativo. Me refiero a ir más allá de los programas y aplicaciones por defecto. GNU/Linux cuenta con diferentes escritorios y gestores de ventanas, de manera que podemos elegir el que queramos, algo que permiten muchas distribuciones GNU/Linux. Mientras que Windows cuenta con un escritorio por defecto, en GNU/Linux podemos elegir entre: GNOME, KDE, LXQt, LXDE, Xfce, Unity, MATE, Cinnamon, Pantheon, Deepin, Budgie, PIXEL, Enlightenment, Trinity, Moksha, Ukui, etc. En la variedad está el gusto.

Además, cualquier configuración o elemento del sistema operativo es susceptible de ser alterado[23]. La única limitación es que seamos capaces o tengamos los conocimientos adecuados. Pero siempre podemos encontrar en internet un tutorial donde nos explique cómo hacerlo.

Existen distribuciones de Linux de tamaño muy reducido, por ejemplo: BasicLinux ocupa 2.8 MB, requiere un procesador 386 y 3 MB de RAM y cuenta con el escritorio gráfico JWM, Nanolinux ocupa 14 MB, utiliza SLWM como escritorio y cuenta con navegador, procesador de texto, hoja de cálculo, cliente IRC, etc.

Actualizaciones del Sistema Operativo hablando de actualizaciones, sus aplicaciones se actualizan prácticamente al día, en cuanto el desarrollador lanza dicha actualización. Por lo que siempre podemos tener nuestros programas y aplicaciones actualizadas.

Además para los usuarios que así lo requieran existen versiones de soporte a largo plazo (Long-Term Support , LTS) normalmente se asocia con una aplicación o un sistema operativo para el que obtendremos seguridad, mantenimiento y (a veces) actualizaciones de funciones durante un período de tiempo más largo.

Las versiones LTS se consideran las versiones más estables que se someten a pruebas exhaustivas y en su mayoría incluyen años de mejoras en el camino.

[23] BlendOS este prometedor sistema operativo, introduce muchas novedades, empezando porque ahora soporta distintas distribuciones: Arch (el principal), AlmaLinux, Crystal Linux, Debian, Fedora, Kali Linux, Neurodebian Bookworm, Rocky Linux y Ubuntu.

Además de estar disponible en siete entornos gráficos, y que se puede cambiar entre ellos con un sencillo comando. Los entornos en los que está son GNOME, KDE (Plasma), Cinnamon, Xfce, LXQt, MATE y Deepin. El comando para ir cambiando entre los escritorios disponibles es: *sudo system track*. Esta distribución es inmutable, por lo que es difícil que subir de versión estropee algo. Básicamente son imágenes completas a las que se le pueden hacer pequeños retoques, como instalar nuevo software. Pero casi todo va por contenedores.

Es importante tener en cuenta que una versión de Software LTS no implica necesariamente actualizaciones de funciones a menos que haya una versión más reciente de LTS. Sin embargo, obtendrá las correcciones de errores y las correcciones de seguridad necesarias en las actualizaciones de una versión de Soporte a largo plazo.

Se recomienda una versión LTS para consumidores, negocios y empresas listos para la producción porque obtiene años de soporte de Software y sin cambios que rompan el sistema con las actualizaciones. Si observamos una versión que no es LTS para cualquier Software, generalmente es la versión más avanzada con nuevas funciones y un período corto de soporte (por ejemplo, 6-9 meses) en comparación con 3-5 años de soporte en un LTS.

Tiendas de Aplicaciones lo mejor de las distribuciones de GNU/Linux es que tienen una característica en común, sus tiendas de aplicaciones. Ya que vamos a poder instalar cualquier tipo de programa que necesitemos con un Click. Recordamos que esto es algo que Windows está intentando con su propia tienda de aplicaciones, pero no están teniendo muy buenos resultados.

Compatibilidad muchos han experimentado problemas a la hora de actualizar sus sistemas operativos con los programas que tenían instalados. Pero eso con GNU/Linux, no pasa, ya que todas sus actualizaciones tienen retrocompatibilidad a largo plazo dentro de su distribución.

Hoy en día la mayoría de aplicaciones y servicios Online cuentan con versión compatible para cualquier sistema operativo. Siendo más fácil crear una aplicación multiplataforma, por lo que GNU/Linux cuenta con un catálogo de Software que poco o nada tiene que envidiar a Windows o Mac OS X.

En el catálogo destacan las aplicaciones gratuitas y de código abierto, pero también surgen proyectos comerciales, y en la lista se incluyen los juegos, cada vez más presentes en GNU/Linux.

Seguramente hay algún Software no disponible en GNU/Linux, pero es más que probable que encontremos una alternativa o, en su defecto, que podamos ejecutarlo mediante Wine o empleando máquinas virtuales como KVM/QEMU o VirtualBox.

En cuanto al Hardware, la comunidad GNU/Linux ha avanzado mucho en la creación de controladores o Drivers para emplear cualquier dispositivo o componente en GNU/Linux. Podemos encontrarnos con alguna excepción, pero la mayoría de dispositivos cuentan con un controlador compatible por

defecto.

Está en Todas Partes GNU/Linux está presente en la infraestructura de grandes empresas como Amazon, Facebook, Netflix, NASA, SpaceX, el gran colisionador de hadrones o IBM y en el año 2021 llegó a Marte en el sistema operativo del helicóptero que acompaña al rover Perseverance, etc. A nivel de usuario, muchos dispositivos emplean este sistema operativo, bien en alguna de sus versiones o a través de Android, que salvando las distancias, todavía conserva gran parte de su origen Linuxero. Por otro lado, las quinientas principales supercomputadoras emplean Linux como sistema operativo, ya que permite trabajar en todo tipo de entornos y situaciones.

Las grandes empresas de internet hace años que vieron en GNU/Linux una gran oportunidad, y si bien a nivel usuario doméstico no está tan extendido, nunca había sido tan fácil dar el paso. Para hacernos una idea, sólo hay que ver la lista de empresas que apoyan a GNU/Linux a través de The Linux Foundation. Una de las más recientes, la propia Microsoft.

La Comunidad GNU/Linux finalmente, hay que hablar de la fabulosa comunidad de GNU/Linux. Podemos preguntar lo que queramos en sus foros, cambiar el código, enviar tus programas, sin problemas. ¿Trabas en la configuración? Te lo solucionan sin preocupación, ¿consejos sobre Software? Hay cientos de hilos con soluciones. Y nosotros ponemos nuestro granito de arena con este trabajo.

Programas de Windows y macOS en Linux A medida que va pasando el tiempo, las diferencias entre los sistemas operativos se van volviendo irrelevantes. Máquinas virtuales, contenedores y otras tecnologías permiten que podamos utilizar cada día más títulos de nuestros programas preferidos aunque no tenga versión para nuestro sistema operativo.

Wine, la herramienta que actúa como un intérprete entre el núcleo Linux y las aplicaciones Windows ya lleva mucho tiempo entre nosotros. Desde hace poco tiempo, también tenemos una herramienta para los programas de macOS.

Wine para Aplicaciones de Windows Nació inicialmente como un proyecto que buscaba crear un emulador de Windows. Su acrónimo era inicialmente «WINdows Emulator», aunque viendo su evolución, y la forma de funcionar,

este acrónimo fue actualizado por «Wine Is Not an Emulator». Y es que en realidad no es un emulador, sino que este programa está formado por un cargador de programas binarios junto a un conjunto de herramientas de desarrollo que permiten portar en tiempo real el código de las aplicaciones de Windows a Unix. Además, trae por defecto una gran cantidad de bibliotecas y librerías de manera que no tengamos problemas de dependencias.

Principales Características este programa es capaz de ejecutar sin problemas cualquier programa diseñado para cualquier versión de Windows, desde la 3.x hasta Windows 10. Eso sí, solo es compatible con programas Win32 (tanto de 32 bits como de 64 bits), por lo que no vamos a poder ejecutar las apps UWP de la Microsoft Store, al menos por ahora.

Entre toda la variedad de librerías, bibliotecas y recursos, podemos encontrarnos con prácticamente todas las bibliotecas de interrupciones para programas, lo que permite hacer llamadas INT en tiempo real. De esta manera, los programas no saben que se están ejecutando en un sistema operativo que no es Windows, simplemente se ejecutan. Y lo hacen igual que en él. Si algún programa, o juego, tiene dependencias especiales (por ejemplo, una DLL concreta) podemos añadirla fácilmente a Wine. Todas las librerías se encuentran dentro del directorio «~/.wine/drive_c/windows/system32», que equivale al directorio System32 de Windows.

Por supuesto, Wine tiene soporte para una gran cantidad de recursos gráficos. Los programas se pueden dibujar tanto en una interfaz gráfica X11 (el escritorio) como desde cualquier terminal X. Es compatible con las tecnologías OpenGL, DirectX y cuenta con soporte total para GDI (y parcial para GDI32). También permite y gestionar varias ventanas a la vez (del mismo programa, o de diferentes) y es compatible con los temas msstyle de Windows.

También es compatible con los controladores de sonido de Windows, y tiene acceso a los puertos del PC, al Winsock TCP/IP y hasta a los escáneres.

¿Qué Programas y Juegos Puedo Ejecutar con Wine? por desgracia, a pesar de tener una gran compatibilidad, Wine no es capaz de ejecutar el 100% de los programas y juegos de Windows en Linux. Y algunos, aunque se pueden ejecutar, no funcionan del todo bien. Para saber si un programa se puede ejecutar, o no, en Linux, podemos buscarlo en la red del proyecto. Allí vamos a encontrar una gran base de datos que nos va a per-

mitir saber si un programa funciona va a funcionar, si no lo hace, o qué tal lo hace.

Además de poder buscar manualmente cualquier programa o juego, también vamos a encontrar una lista con los Top-10 que mejor funcionan. Los juegos «Platino» son los que funcionan de manera idéntica en Windows que en Linux, los «Oro» los que funcionan bien, pero requieren de alguna configuración especial y los «Plata», los que funcionan, pero tienen pequeños problemas. Los programas o juegos «Bronce» o «Basura» son los que no funcionan.

Saca Todo el Partido a Wine con Estos Programas Wine, al final, es la parte más importante para poder usar programas de Windows en Linux. Sin embargo, su configuración, sobre todo para los programas que no tienen una clasificación de platino, puede ser algo tediosa. Por suerte, existen programas que, aunque se basan igualmente en Wine, nos ayudan a configurar cada uno de estos programas de manera automática para que nosotros no tengamos que hacer nada más.

La instalación y configuración de los programas y juegos de Windows para usarlos en Linux es lo peor. Si no tenemos muchos conocimientos podemos perder mucho tiempo y, además, no conseguiremos que todo funcione del todo bien. Aquí es donde entra en juego *PlayOnLinux*. Este programa, gratuito y de código abierto, busca ayudarnos con la instalación y configuración de los programas y juegos para hacerlos funcionar en este sistema operativo.

PlayOnLinux nos ofrece una completa base de datos de programas con sus correspondientes configuraciones óptimas de manera que nosotros solo tengamos que seleccionar el programa que queremos, cargarle su instalador y dejar que este complete el proceso de puesta en marcha. Nada más. Cuando acabe la instalación ya podremos abrir el programa o juego y empezar a usarlo.

Podemos descargar esta herramienta de forma totalmente gratuita desde su página Web, o desde una termina:

```
# apt install playonlinux
```

Descargar e Instalar Wine hay muchas formas de instalar Wine en Linux. Sus desarrolladores tienen binarios específicos para cada distribución, así como unos completos repositorios desde los que vamos a poder descargar y actualizar el programa desde terminal:

```
# apt install wine
```

WINE no es una aplicación que se inicia por sí sola. Es un backend que se invoca cuando se inicia una aplicación de Windows. Lo más probable es que su primera interacción con WINE ocurra cuando inicie el instalador de una aplicación de Windows.

Ejecutar e Instalar Programas Windows una vez instalado, Wine se ejecutará al hacer doble clic sobre cualquier archivo .EXE. Además, te permitirá instalar programas, como si estuvieras en Windows y pondrá los accesos directos en el menú principal bajo la categoría «Wine».

A pesar de lo que mucha gente cree, Wine sirve no sólo para correr aplicaciones «sencillas» de Windows, sino incluso juegos complejos. Es más, está demostrado que terribles juegazos como Sim 3, Half Life 2, Command & Conquer 3, Star Wars: Jedi Knight, o importantes suites como Microsoft Office funcionan a la perfección.

Darling para Aplicaciones de macOS cumple una función similar a la de Wine con los programas de Windows, solo que no tiene ningún complejo en definirse como un emulador. Lo que hace es actuar como un traductor que permite ejecutar los programas de macOS usando los recursos de Linux. El nombre Darling (Querida) es la primera parte del nombre del núcleo de macOS (Darwin) y las primeras 3 letras de Linux. Supongo que la G final es para construir una palabra fácil de memorizar.

Hay que decir que a los desarrolladores de Darling la cosa les resulta más fácil que a los de Wine. No tienen que hacer ingeniería inversa ni reinventar nada dado que se basan en las partes de Darwin que están bajo licencias abiertas. El propio Darling se distribuye bajo la licencia GPL.

Iniciando Darling el programa no tiene interfaz gráfica. Lo ponemos en marcha desde la terminal con el comando:

```
$ darling shell
```

Al escribirlo, Darling creará un directorio raíz virtual o se conectará con uno existente. Además cargará los módulos del núcleo y construirá el sistema de archivos virtual donde ejecutaremos los programas.

Desde la línea de comandos podemos acceder a dos tipos de sistemas de archivos: el tradicional de macOS que incluye los directorios de nivel superior como */Applications*, */Users* y */System* entre otros. Por otro lado, el del sistema operativo anfitrión lo hallamos en una partición denominada */Volumes/SystemRoot*

Podemos verificar el núcleo con el siguiente comando:

```
$ uname
```

y averiguar la versión de macOS con:

```
$ sw_vers
```

salimos de la terminal con

```
$ exit
```

y apagamos el contenedor con:

```
$ darling shutdown
```

Instalación de Programas Si estás usando Linux en arranque dual con macOS y quieres ejecutar alguno de los programas que tienes instalado en la partición de Mac, puedes hacerlo con el comando:

```
$ /Volumes/SystemRoot/run/media/usuario/Macintosh HD
/Applications/nombre_app.app)
```

Muchos programas para macOS se distribuyen en formato .dmg. Para instalarlos en Darling hacemos:

```
Darling [~]$ hdiutil attach Downloads/aplicación.dmg /Vol-
umes/aplicacion
Darling [~]$ cp -r /Volumes/aplicación/aplicación.app /Ap-
plications/
```

En el caso de aplicaciones almacenadas en archivos comprimidos, lo descomprimimos y copiamos en la carpeta */Applications*. Lo mismo con aplicaciones previamente descargadas de la tienda de aplicaciones.

Por último nos quedan las aplicaciones *.pkg*, el formato de paquete nativo de macOS. Este formato implica ejecutar Scripts durante la instalación. Para poder usarlos debemos hacer:

```
Darling [~]$ installer -pkg aplicación.pkg -target /
```

Podemos desinstalar los programas con:

```
Darling [~]$ uninstaller nombre_del_paquete
```

Debemos entender que si bien Darling funciona muy bien con aplicaciones para la línea de comandos, solo tiene funcionalidades muy limitadas para las que necesitan una interfaz gráfica.

Instalación de Darling Si utilizas Debian o derivados, la instalación de Darling no tiene mayor problema. Solo tienes que escribir los comandos:

```
# apt install gdebi
# gdebi darling-dkms_X.X.X.testing_amd64.deb
# gdebi darling_X.X.X.testing_amd64.deb
```

reemplaza las X por el número de versión de los paquetes que descargarás o bien se puede descargar los archivos fuentes del proyecto para su compilación e instalación.

Aprender a Usar Linux Existen diversos sitios Web que están enfocados a explorar detalladamente cada distribución actual o antigua, a un nivel técnico acompañado de grandes y útiles análisis técnicos sobre los mismos, lo que facilita el aprendizaje puntual sobre qué distribución usar o empezar a usar sin tanta incertidumbre, algunos de estos lugares son:

- ArchiveOS https://archiveos.org
- Distro Chooser https://distrochooser.de/es/
- Distro Watch https://distrowatch.com
- Linux Distribution List https://lwn.net/Distributions/

¿Qué otros sabores de GNU/Linux hay?

https://upload.wikimedia.org/wikipedia/commons/1/1b/Linux_Distribution_Timeline.svg

Existen distintas distribuciones de GNU/Linux[24] para instalar, una de las más ampliamente usadas es Debian GNU/Linux[25] y sus derivadas como Ubuntu. La comunidad de GNU/Linux te apoya para obtener, instalar y que de una vez por todas puedas usar GNU/Linux en tu computadora.

Puedes conocer y descargar las diferentes distribuciones desde:

https://es.wikipedia.org/wiki/Anexo:Distribuciones_Linux

https://en.wikipedia.org/wiki/List_of_Linux_distributions

y ver cuál es la que más te conviene:

https://en.wikipedia.org/wiki/Comparison_of_Linux_distributions

o probar alguna versión Live[26]:

https://livecdlist.com/

también las puedes correr como máquina virtual para VirtualBox:

https://www.osboxes.org/

o máquina virtual para QEMU/KVM:

https://docs.openstack.org/image-guide/obtain-images.html

https://github.com/palmercluff/qemu-images

https://bierbaumer.net/qemu/

[24] Una lista de las distribuciones de Linux y su árbol de vida puede verse en la página Web http://futurist.se/gldt/

[25] Algunas de las razones para instalar GNU/Linux Debian están detalladas en su página Web https://www.debian.org/intro/why_debian.es.html

[26] Linux es uno de los sistemas operativos pioneros en ejecutar de forma autónoma o sin instalar en la computadora, existen diferentes distribuciones Live -descargables para formato CD, DVD, USB- de sistemas operativos y múltiples aplicaciones almacenados en un medio extraíble, que pueden ejecutarse directamente en una computadora, estos se descargan de la Web generalmente en formato ISO.

por otro lado, existen diferentes servicios Web que permiten instalar, configurar y usar cientos de sistemas operativos Linux y Unix desde el navegador, una muestra de estos proyectos son:

Distrotest https://distrotest.net
JSLinux https://bellard.org/jslinux
OnWorks https://www.onworks.net

Ahora, Windows 10 Build 2020 con WSL2[27] (Windows Subsystem for Linux), tiene su propio Kernel de Linux que permite instalar de manera casi nativa diversas distribuciones de GNU/Linux permitiendo tener lo mejor de ambos mundos en un mismo equipo.

En la red existen múltiples sitios especializados y una amplia bibliografía para aprender a usar, administrar y optimizar cada uno de los distintos aspectos de Linux, nosotros hemos seleccionado diversos textos que ponemos a su disposición en:

Sistemas operativos

2.5 Android

Android (véase [6]) es un sistema operativo basado en el núcleo Linux (véase apéndice 14.2). Fue diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes, tabletas y también para relojes inteligentes, televisores y automóviles. Inicialmente fue desarrollado por Android Inc., empresa que Google respaldó económicamente y más tarde, en 2005, compró. Android fue presentado en 2007 junto a la fundación del Open Handset Alliance (un consorcio de compañías de Hardware, Software y telecomunicaciones) para avanzar en los estándares abiertos de los dispositivos móviles. El primer móvil con el sistema operativo Android fue el HTC Dream y se vendió en octubre de 2008. Android es el sistema operativo móvil (SmartPhone y tabletas) más utilizado del mundo, con una cuota de mercado del 86% al año 2020, muy por encima del 13.9% de iOS.

El éxito del sistema operativo lo ha convertido en objeto de litigios sobre patentes en el marco de las llamadas guerras de patentes entre las empresas de teléfonos inteligentes. Según los documentos secretos filtrados en 2013 y 2014,

[27] https://docs.microsoft.com/en-us/windows/wsl/install-win10

el sistema operativo es uno de los objetivos de las agencias de inteligencia internacionales.

La versión básica de Android es conocida como Android Open Source Project (AOSP). El 25 de junio de 2014 en la Conferencia de Desarrolladores Google I/O, Google mostró una evolución de la marca Android, con el fin de unificar tanto el Hardware como el Software y ampliar mercados. El 17 de mayo de 2017, se presentó Android Go. Una versión más ligera del sistema operativo para ayudar a que la mitad del mundo sin Smartphone consiga uno en menos de cinco años. Incluye versiones especiales de sus aplicaciones donde el consumo de datos se reduce al máximo.

Arquitectura del Sistema Android los componentes principales del sistema operativo de Android[28]:

Aplicaciones: las aplicaciones base incluyen un cliente de correo electrónico, programa de SMS, calendario, mapas, navegador, contactos y otros. Todas las aplicaciones están escritas en lenguaje de programación Java.

Marco de trabajo de aplicaciones: los desarrolladores tienen acceso completo a las mismas API del entorno de trabajo usadas por las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes; cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del Framework). Este mismo mecanismo permite que los componentes sean reemplazados por el usuario.

Bibliotecas: Android incluye un conjunto de bibliotecas de C/C++ usadas por varios componentes del sistema. Estas características se exponen a los desarrolladores a través del marco de trabajo de aplicaciones de Android. Algunas son: System C library (implementación biblioteca C estándar), bibliotecas de medios, bibliotecas de gráficos, 3D y SQLite, entre otras.

[28] Android tiene la base de Linux, por ello en cualquier dispositivo que soporte dicho sistema operativo es posible instalar una aplicación para acceder a la terminal de línea de comandos -por ejemplo ConnectBot-, y en ella podemos correr los comandos de BASH como en un sistema GNU/Linux.

Runtime de Android: Android incluye un conjunto de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java. Cada aplicación Android ejecuta su propio proceso, con su propia instancia de la máquina virtual Dalvik. Dalvik ha sido escrito de forma que un dispositivo puede ejecutar múltiples máquinas virtuales de forma eficiente. Dalvik ejecutaba hasta la versión 5.0 archivos en el formato de ejecutable Dalvik (.dex), el cual está optimizado para memoria mínima. La Máquina Virtual está basada en registros y corre clases compiladas por el compilador de Java que han sido transformadas al formato .dex por la herramienta incluida DX. Desde la versión 5.0 utiliza el ART, que se compila totalmente al momento de instalación de la aplicación.

Personalización muchos conocen a Android como el sistema operativo móvil más personalizable. Pero para los que no lo saben, recordamos que está basado en el núcleo de Linux y que muchos desarrolladores están queriendo llevar Android a un sistema operativo de escritorio.

Núcleo Linux: Android depende de Linux para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores. El núcleo también actúa como una capa de abstracción entre el Hardware y el resto del Software.

Android sobre KVM (MicroDroid) En enero de 2021 se anunció por parte de Google que trabajan en MicroDroid, una versión minimalista de Android para máquinas virtuales sobre KVM. No es la primera alternativa a Android, nos encontramos con las capas de personalización Android Go, AOSP o las imágenes GSI. En el caso de MicroDroid, estaríamos ante una limitada imagen de Linux basada en Android, para este proyecto, Google está trabajando en adaptar la máquina virtual de Chrome OS (crosvm), que ya se utiliza para ejecutar aplicaciones de Linux en Chrome OS. De esta forma con MicroDroid podría ejecutar pequeñas máquinas virtuales junto a Android, posiblemente para aplicaciones y uso relacionado con DRM, esta modificación constaría con el mínimo de componentes para iniciar el sistema permitiendo aislar datos entre aplicaciones y sistemas operativos en el mismo dispositivo, así como cambiar instantáneamente entre sistemas operativos.

2.6 Chromebook y Chrome OS

Para entender la razón de ser de los Chromebooks, primero tenemos que entender qué es Chrome OS. Se trata de un sistema operativo creado por Google y diferente a Android. Está basado en el Kernel de Linux, y utiliza Chrome como su interfaz de usuario principal. Esto quiere decir que su aspecto es prácticamente idéntico al de Chrome, pero con algunos añadidos como una barra de tareas, un explorador de archivos y otros elementos presentes en cualquier sistema operativo.

Fue anunciado a mediados del 2009 como un intento de crear un sistema basado en la nube y en aplicaciones Web. Esto hacía que, cuando se estaba conectado a internet se pudieran hacer muchas cosas gracias a herramientas como Google Drive o las aplicaciones de la Chrome Web Store, pero que cuando dejaba de tener internet se limitaba mucho sus funciones.

En cualquier caso, y pese a lo limitado que era en sus primeros años, poco a poco Google lo ha hecho evolucionar. Primero se empezaron a añadir opciones a las aplicaciones de Google para poderse utilizar sin conexión, algo que también benefició a los usuarios que usaran Chrome en otros sistemas operativos.

Pero la evolución más grande fue llegando después. El primer gran paso fue el anuncio de la compatibilidad para ejecutar aplicaciones de Android, y se fue implementando directamente la tienda de aplicaciones Google Play de Android para hacer que la experiencia de instalarlas fuera tan nativa como en Android. Aun así, hay que decir que la llegada de Android a Chrome OS ha sido lenta, y han tardado algunos años en hacer que todo vaya funcionando como debería.

Y a mediados de 2018 se anunció que Google Chrome también podrá utilizar aplicaciones creadas para los sistemas GNU/Linux. Con ello, el catálogo de aplicaciones diseñadas para funcionar sin conexión se multiplica beneficiando a la comunidad de desarrolladores libres, aunque también es de esperar que tarde algunos años en estar todo perfectamente integrado, ya que todavía se están lanzando poco a poco mejoras.

Chrome OS es hoy en día un sistema operativo completo. Tiene lo básico, aplicaciones nativas y compatibilidad con Android, que se une al reproductor de medios, gestor de archivos, configuración de impresoras, etcétera. Además, al igual que el navegador, Chrome OS tiene también una versión libre llamada Chromium OS, que pese a no tener la tecnología nativa de Google sirve para que la comunidad de desarrolladores independientes pueda ayudar a

mejorarlo.

Ahora bien, los Chromebook son equipos de cómputo personales que utilizan como sistema operativo Chrome OS, desarrollado por Google y que, a diferencia de Windows, OS X y Linux, están pensados para utilizarse permanentemente conectados a internet, ya que se basan casi completamente en la nube.

Chromebook Apps también se incluye un reproductor multimedia, y todo se sincroniza permanentemente en la nube. Por ello, si pretendemos utilizar un Chromebook sin conexión a internet, su funcionalidad es más limitada que la de otros equipos de cómputo. De hecho, las aplicaciones se instalan a través de Chrome Web Store, la tienda de aplicaciones integrada en Google Chrome, con lo que algunas de las herramientas más habituales (como Office o Skype, por ejemplo) tendrían que verse reemplazadas por Google Drive y Google Hangouts, aplicaciones nativas de Google.

Chrome Web Store no obstante, también se pueden utilizar de forma local sin recurrir a la red, ya que muchos de los servicios de Google disponen de un modo sin conexión que, una vez volvemos a disponer de internet, se sincronizaran sin problemas.

¿Cómo es un Chromebook? en un Chromebook podemos utilizar dispositivos USB sin problemas, como memorias y discos externos, Webcams, teclados y ratones, y por lo general suelen venir con una cantidad de almacenamiento inferior a lo que estamos acostumbrados (ya que lo que se pretende es que todo esté en la nube, y no en nuestro disco duro local). De hecho, al adquirir uno se nos obsequia con 100 GBytes de espacio en Google Drive.

Igualmente, su precio suele ser bastante asequible (desde 179 dólares o 130 euros) y no requieren de un Hardware potente para funcionar, siendo la ligereza de recursos una de sus mayores bondades. Por su parte, los equipos de cómputo portátiles con Chrome OS son lo que llamamos Chromebook, mientras que si preferimos el formato Mini PC, estaremos ante un Chromebox.

El inicio del sistema es prácticamente instantáneo y todo está listo para funcionar en cuestión de segundos, y dadas sus características, un Chromebook es un equipo ideal para navegar por internet ante todo.

Se accede desde la barra de herramientas en la parte inferior de la pantalla a las aplicaciones que tengamos instaladas, que en realidad se trata de un atajo a las apps que tengamos instaladas en Google Chrome.

Chromebook Integración por supuesto, los Chromebook también son multiusuario, con la ventaja de que con simplemente iniciar sesión con otra cuenta de Gmail todo estará tal y como si lo hubiésemos configurado con ella (aplicaciones, servicios, historial y demás), y por este mismo motivo se complementan a la perfección con otros dispositivos (ya sean equipos de cómputo, Smartphones o Tablets) en los que utilicemos los servicios de Google, gracias a la sincronización en la nube.

Además, los Chromebook también presumen de no necesitar antivirus, pues al almacenarse todo en la nube la seguridad está integrada por defecto y corre por parte de Google.

Microsoft en un Chromebook En el 2020 las empresas Parallels[29] y Google llegaron a un acuerdo para ofrecer a los usuarios la posibilidad de ejecutar aplicaciones Windows en Chrome OS. Ellas aseguran que en Chrome OS la integración será completa: las aplicaciones se ejecutarán cada una en su propia ventana, como las nativas, y no dentro de un Windows virtualizado.

Aunque ninguna de las dos compañías ha ofrecido aún una lista de aplicaciones compatibles con esta función que será lanzada en el 2021, John Solomon (vicepresidente de Chrome OS) ha afirmado que Microsoft Office será una de ellas.

El problema es que, por ahora, estas nuevas funcionalidades no estarán disponibles para todos los usuarios de Chrome OS, sino únicamente para los de Chrome OS Enterprise, la versión empresarial del mismo.

Nota: en últimas fechas han aparecido proyectos que permiten instalar diversas distribuciones de GNU/Linux en los Chromebook, esto es debido a que Google deja de dar soporte a sus equipos después de algunos años de que salieron al mercado, pese a que el equipo es totalmente funcional.

Chrome OS Flex En febrero del 2022, Google anunció su nueva versión de Chrome OS para equipos de cómputo PCs y Macs, la propuesta es Chrome

[29] Empresa (propiedad de Corel desde hace un año) desarrolladora del Software homónimo de virtualización que es especialmente popular entre los usuarios de Mac.

OS Flex y cuya descarga es totalmente gratuita, tiene como propósito atender las necesidades de escuelas y empresas rehusando equipo (procesador Intel o AMD 64 bits, 4 GB RAM, 16 GB de almacenamiento, etc). Esta versión tiene la misma interfaz gráfica y herramientas básicas que se encuentran en Chrome OS; entre ellas el navegador Chrome, se ofrece soporte para sincronización de ajustes y marcadores. Además, si nuestro equipo cumple con las especificaciones básicas, tendremos a nuestra disposición a Google Assistant e integraciones diversas con dispositivos Android.

Es de notar que Chrome OS Flex no tiene soporte para la Play Store o para las aplicaciones de Android y al parecer no hay intención de añadir esta compatibilidad. Tampoco se puede ejecutar Windows en una máquina virtual de Parallels Desktop.

Se puede descargar Chrome OS Flex para crear una unidad USB booteable y la instalación reemplazará al sistema operativo del equipo donde se instale (Mac, Windows o Linux). Pero no está optimizado para sacar provecho de todos los puertos, sensores o accesorios que pueden estar presentes en el equipo. Esto nos proporcionará una experiencia lo más cercana posible a Chrome OS sin comprar un Chromebook pese a sus limitaciones.

2.7 Otros Sistemas Operativos

Sistemas Operativos para PC

1. Fuchsia OS.- Es un sistema operativo versátil y adaptable esta basado en el microkernel Zircon en desarrollo por parte de Google, está disponible desde un repositorio de Git y está ya siendo usado en los Nest Hub, se espera su uso en la domótica que prepara Google como parte del internet de las cosas.

2. Dahlia OS.- Este sistema operativo combina lo mejor de GNU/Linux y Fuchsia OS es moderno, seguro, liviano y receptivo. Se mantiene minimalista al incluir solamente las aplicaciones necesarias, pero es posible agregar todos nuestros favoritos de otros sistemas operativos usando aplicaciones Containers y proporciona una tienda para aplicaciones Flutter de terceros.

3. HyperOS.- La nueva estrategia de la firma asiática Xiaomi busca integrar la experiencia de Hombre x Coche x Hogar gracias a HyperConnect. Será la capa que englobará a todos los dispositivos del ecosistema

Xiaomi y gracias al nuevo protocolo HyperConnect que sirvira para crear una "red dinámica en tiempo real autónoma entre dispositivos" del ecosistema como Smartphones, tabletas, dispositivos IoT como el aire acondicionado, etc.

4. KataOS.- Es un sistema operativo centrado en la seguridad y los sistemas embebidos que está construido casi enteramente con Rust. No emplea Linux ni Fuchsia, sino el micronúcleo seL4, el cual, según Google, "pone la seguridad al frente y en el centro". Lo que se pretende con este sistema operativo es proporcionar "una plataforma segura verificable que protege la privacidad del usuario porque es lógicamente imposible que las aplicaciones violen las protecciones de seguridad del Hardware incluidas en el kernel, además de que los componentes del sistema son seguros de forma verificable".

5. ToaruOS.- Es un sistema operativo escrito desde cero y provisto con su propio Kernel, cargador de arranque, biblioteca C estándar, administrador de paquetes, componentes de espacio de usuario y una interfaz gráfica con un administrador de ventanas compuesto. Se inició como un proyecto de investigación en la Universidad de Illinois en el 2010 y a partir del 2012 es desarrollado por la comunidad interesada.

6. Essence, es un sistema operativo con su propio Kernel y escritorio construido desde cero por un entusiasta desde 2017 y se destaca por su original escritorio y pila de gráficos que permite dividir ventanas en pestañas, lo que permite trabajar en una ventana con varios programas a la vez y agrupar aplicaciones en ventanas según las tareas a resolver. El administrador de ventanas funciona al nivel del Kernel del sistema operativo y la interfaz se crea utilizando su propia biblioteca gráfica y un motor de Software vectorial que admite efectos animados complejos completamente vectoriales.

7. eComStation.- Seguro que muchos recuerdan el mítico OS/2 de IBM, sistema operativo que perdura con eComStation, derivado de este adaptado al Hardware moderno. A diferencia de otras alternativas de la lista, este no es gratuito y sus precios comienzan desde 145 dólares para la versión doméstica. Muchas aplicaciones libres como Firefox, OpenOffice o VLC han sido portadas a este sistema operativo.

8. Haiku.- BeOS fue un sistema operativo lanzado en el año 1991 con muy buenas intenciones a nivel de optimización e interfaz. Sin embargo, como les sucedió a muchos otros, terminó sucumbiendo en este complicado mercado. Su legado ha sido continuado por Haiku, un sistema de código abierto que lleva ya años en desarrollo.

9. ReactOS.- Es una alternativa a la arquitectura Windows NT de Microsoft totalmente abierta que no utiliza ningún tipo de código propietario. No obstante, es compatible con muchos de los controladores y aplicaciones de Windows. Como punto negativo, su desarrollo no es tan rápido como muchos esperarían en un entorno tan cambiante como este.

10. FreeDOS.- Alternativa libre a DOS cuyo desarrollo sigue activo en estos momentos. Se trata de un entorno bastante estable, pero que carece de interfaz gráfica o multitarea. Es compatible a todos los niveles con MS-DOS y sus programas.

11. Solaris.- El sucesor de SunOS, de Sun Microsystems, empezó como una distribución propietaria de UNIX, pero en 2005 fue liberado como OpenSolaris. Más tarde, Oracle compró Sun y le cambió el nombre a Oracle Solaris.

12. Illumos.- Basado en Open Solaris, este proyecto nació por parte de algunos de los ingenieros originales del sistema. En realidad, busca ser una base para crear distribuciones de este sistema operativo. OpenIndiana es una de las más conocidas y utilizadas.

13. DexOS.- Un sistema operativo de 32 Bits escrito para la arquitectura x86 en lenguaje ensamblador. Está diseñado para programadores que desean tener acceso directo al Hardware (incluyendo CPU y gráficos) con un código bien comentado y documentado.

14. Syllable.- Sistema operativo nacido como fork de AtheOS, un clon de AmigaOS, aunque comparte mucho código con Linux. No tiene demasiada utilidad para los usuarios domésticos, aunque es compatible con arquitecturas x86.

15. AROS Research Operating System.- Es otro sistema que implementa en código abierto las APIs de AmigaOS, con cuyos ejecutables es compatible a nivel binario en procesadores de 68k, además de ser compatible

a nivel de código con otras arquitecturas como x86 para la que se ofrece de manera nativa. Es portable y puede correr hospedado en Windows, Linux y FreeBSD.

16. MenuetOS.- Llamado también como MeOS, su característica más destacada es que está programado completamente en lenguaje ensamblador. Está diseñado para funcionar en equipos muy básicos aunque soporta hasta 32 GigaBytes de RAM. Con decir que el sistema cabe en un disquete de 1.44 Megabytes, está dicho todo. Aún así se las arregla para incluir un escritorio gráfico y controladores para teclados, video, audio, USB o impresoras.

17. Visopsys.- Se trara de un sistema gratuito y libre bajo GPL que ha estado en desarrollo desde 1997, como hobby de un solo programador, Andy McLaughlin. Soporta arquitecturas x86, está escrito en C y ensamblador y no se basa en ningún sistema preexistente, si bien utiliza código del kernel Linux, ofrece herramientas comunes de GNU y parte de la interfaz gráfica de usuario como los iconos, resultaran familiares a los usuarios de KDE Plasma.

18. mOS.- Sistema operativo usado en centros de datos y para cómputo de alto rendimiento (High Performance Computing HPC), se basa en el Kernel de Linux pero tiene su propio núcleo ligero LWK, el Kernel gestiona un pequeño número de núcleos de la CPU para asegurarse la compatibilidad y el LWK Kernel gestiona el resto del sistema.

19. KolibriOS.- Es un pequeño sistema operativo poderoso y rápido para PCs. Solamente requiere unos pocos megas de espacio en disco y 8 MB de RAM para funcionar, además de incluir varias aplicaciones básicas.

20. SerenityOS es un sistema operativo Unix con aspecto de Windows de los 90s creado por un único programador como un proyecto terapéutico y está pensado para equipos X86 de escritorio.

21. BlendOS este prometedor sistema operativo Linux, introduce muchas novedades, empezando porque ahora soporta distintas distribuciones: Arch (el principal), AlmaLinux, Crystal Linux, Debian, Fedora, Kali Linux, Neurodebian Bookworm, Rocky Linux y Ubuntu. Además de estar disponible en siete entornos gráficos, y que se puede cambiar entre ellos con un sencillo comando. Los entornos en los que está son

GNOME, KDE (Plasma), Cinnamon, Xfce, LXQt, MATE y Deepin. Esta distribución es inmutable, por lo que es difícil que subir de versión estropee algo. Básicamente son imágenes completas a las que se le pueden hacer pequeños retoques, como instalar nuevo Software, pero casi todo va por contenedores.

Sistemas Operativos para móviles

1. PinePhone.- Usa un sistema operativo basado en sistemas operativos de código abierto impulsado por la comunidad Linux, ha sido portado a 16 diferentes distribuciones de Linux y 7 diferentes interfaces gráficas de usuario como: Mobian, Manjaro con interfaz plasma, Ubuntu Touch, postmarketOS, LuneOS, Nemo Mobile, Maemo Leste, Tizen, entre otros. Además la compañía Pine64 es el segundo fabricante de teléfonos (después de OpenMoko) que ofrece el arranque desde una tarjeta microSD, que permite a los usuarios probar múltiples sistemas operativos, antes de instalarse en la memoria Flash interna.

2. HarmonyOS.- Sistema operativo desarrollado por Huawei para reemplazar a Android en sus equipos, es un sistema operativo similar a la idea de Fuchsia OS, con la idea que pueda instalarse tanto en un ordenador, como en un teléfono, tableta, relojes, como en un coche conectado, en donde todos estos dispositivos se conecten entre sí con una sola cuenta, dando así un paso hacia adelante en la utopía de la convergencia.

3. PostmarketOS.- Sistema operativo de Software libre y código abierto en desarrollo principalmente para teléfonos inteligentes y tabletas -es una idea genial, la persecución de tener Linux en los dispositivos Smartphone, como otra alternativa a los sistemas Android e iOS-, haciéndose las primeras pruebas en teléfonos que ya no tienen uso. Distribución basada en Alpine Linux. Puede usar diferentes interfaces de usuario, por ejemplo Plasma Mobile, Hildon, LuneOS UI, MATE, GNOME 3 y XFCE.

4. Plasma Mobile.- Es un sistema en fase de desarrollo por KDE que permite la convergencia con los usuarios de KDE para escritorio.

5. Lomiri.- Sistema operativo basado en Linux que soporta dos sabores: Ubuntu Touch y Manjaro. Ambos basados en Unity 8 que están en constante desarrollo.

6. Windows Phone.- Sistema operativo móvil desarrollado por Microsoft, como sucesor de Windows Mobile. A diferencia de su predecesor fue enfocado en el mercado de consumo en lugar del mercado empresarial.

7. Symbian OS.- Era un sistema operativo que fue producto de la alianza de varias empresas de telefonía móvil, entre las que se encuentran Nokia, Sony Ericsson y otros, el objetivo de Symbian fue crear un sistema operativo para terminales móviles.

8. BlackBerry OS.- Es un sistema operativo móvil desarrollado por Research In Motion para sus dispositivos BlackBerry.- Es multitarea y tiene soporte para diferentes métodos de entrada adoptados por RIM para su uso en computadoras de mano, particularmente la trackwheel, trackball, touchpad y pantallas táctiles.

9. HP webOS.- Se trata de un sistema operativo multitarea para sistemas embebidos basado en Linux, desarrollado por Palm Inc., ahora es propiedad de Hewlett-Packard Company.

10. GrapheneOS.- Es un sistema operativo móvil centrado en la privacidad y la seguridad con compatibilidad con aplicaciones Android, que está desarrollado como un proyecto de código abierto sin ánimo de lucro. Se centra en la investigación y el desarrollo de tecnología de privacidad y seguridad, incluyendo mejoras sustanciales en el Sandboxing, la mitigación de exploits y el modelo de permisos.

11. Sailfish OS.- Es un Sistema Operativo móvil seguro y optimizado para funcionar en Smartphones y tabletas, y también fácilmente adaptable a todo tipo de dispositivos integrados y casos de uso. Es el único Sistema Operativo móvil independiente basado en el código abierto, sin ningún vínculo con las grandes corporaciones, respaldado por unos sólidos derechos de propiedad intelectual, que incluyen todos los derechos de propiedad intelectual y marcas comerciales. En resumen, es una plataforma abierta con un modelo de contribución de código abierto activo.

12. Bada.- Fue un sistema operativo para teléfonos móviles desarrollado por Samsung (Bada «océano» o «mar» en coreano). Diseñado para cubrir teléfonos inteligentes de gama alta como gama baja.

3 El Sistema Operativo GNU/Linux

GNU/Linux se ve y se siente muy parecido a cualquier otro sistema UNIX, y de hecho la compatibilidad con UNIX ha sido una importante meta del diseño del proyecto Linux. No obstante, Linux es mucho más joven que la mayor parte de los sistemas UNIX. Su desarrollo se inició en 1991, cuando un estudiante finlandés, Linus Torvalds, escribió y bautizó un pequeño pero autosuficiente núcleo para el procesador 80386, el primer procesador de 32 bits verdadero en la gama de CPU compatibles con el PC de Intel.

En los albores de su desarrollo, el código fuente de Linux se ofrecía gratuitamente en internet. En consecuencia, su historia ha sido una colaboración de muchos usuarios de todo el mundo que se han comunicado casi exclusivamente a través de internet. Desde un núcleo inicial que implementaba parcialmente un subconjunto pequeño de los servicios de UNIX, Linux ha crecido para incluir cada vez más funcionalidades UNIX.

En sus inicios, el desarrollo de Linux giraba en gran medida alrededor del núcleo del sistema operativo central: el ejecutivo privilegiado que administra todos los recursos del sistema e interactúa directamente con el Hardware. Desde luego, se requiere mucho más que este núcleo para producir un sistema operativo completo.

Resulta útil hacer la distinción entre el núcleo (Kernel) de Linux y un sistema Linux: el núcleo en Linux es una identidad de Software totalmente original desarrollada desde cero por la comunidad Linux (suele encontrarse en el directorio */boot* en el sistema de archivos); el sistema Linux, tal como lo conocemos hoy, incluye una multitud de componentes, algunos escritos desde cero, otros tomados en préstamo de otros proyectos o creados en colaboración con otros equipos como el proyecto GNU de la Free Software Foundation.

El sistema Linux básico es un entorno estándar para aplicaciones y programación de los usuarios, pero no obliga a adoptar mecanismos estándar para controlar las funcionalidades disponibles como un todo.

GNU/Linux en sus inicios trabajaba en modo consola o terminal -línea de comandos o *Shell*- usando la interfaz de línea de comando (CLI por Command Line Interface) o interfaz de usuario de terminal (TUI por Terminal User Interface). A medida que Linux ha madurado, se ha hecho necesaria otra capa de funcionalidad encima del sistema Linux: El servidor de Pantalla

Un servidor de pantalla es un programa cuya tarea principal es coordinar la entrada y salida de sus clientes hacia y desde el resto del sistema operativo, el Hardware y entre sí. El servidor de pantalla se comunica con sus clientes a

través del protocolo del servidor de pantalla. Un servidor de visualización es un componente clave en cualquier interfaz gráfica de usuario, específicamente el sistema de ventanas. Es el componente básico de la interfaz gráfica de usuario (GUI) que se encuentra entre la interfaz gráfica y el Kernel.

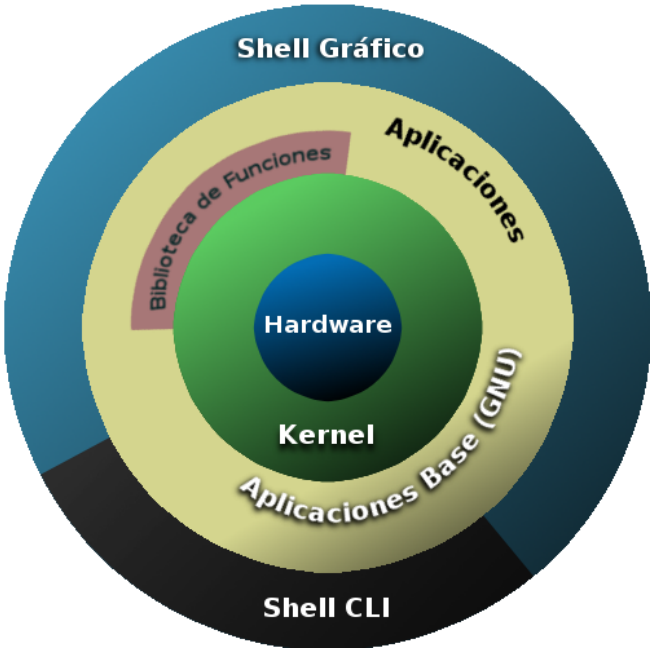

Figura 1: GNU/Linux

Entonces, gracias a un servidor de pantalla, podemos usar la computadora con GUI. Sin él, solo estaría restringido a una interfaz de línea de comandos, sería placentero, pero la GUI también tiene magia. No confundir el servidor de visualización con el entorno de escritorio (GNOME, KDE, LXQt, LXDE, Xfce, Unity, MATE, Cinnamon, Pantheon, Deepin, Budgie, PIXEL, Enlightenment, Trinity, Moksha, Ukui, etc.), los entornos de escritorio usan uno, pero en una capa por debajo.

El servidor de pantalla se comunica con sus clientes a través del protocolo del servidor de pantalla. Hay tres protocolos principales de servidor de pantalla disponibles: X11, Mir (desarrollado por Canonical) y Wayland.

X Window System, a menudo denominado simplemente X, es el más antiguo de todos (1984), terminó siendo el sistema de ventanas predeterminado para la mayoría de los sistemas operativos similares a UNIX, incluido

GNU/Linux. El servidor X.Org es la implementación gratuita y de código abierto del servidor de visualización X Window System administrado por la Fundación X.Org. Es una aplicación que interactúa con aplicaciones cliente a través del protocolo X11 para dibujar cosas en una pantalla y enviar eventos de entrada como movimientos del mouse, Clics y pulsaciones de teclas. Normalmente, uno iniciaría un servidor X que esperará a que las aplicaciones de los clientes se conecten a él. Xorg se basa en un modelo cliente/servidor y, por lo tanto, permite que los clientes se ejecuten de forma local o remota en una máquina diferente. La mayoría de las funciones que proporciona el protocolo X Server ya no se utilizaban. Casi todo el trabajo que hizo X11 se volvió a delegar a las aplicaciones individuales y al administrador de ventanas. Y, sin embargo, todas esas características antiguas siguen ahí, pesando sobre todas estas aplicaciones, perjudicando el rendimiento y la seguridad. Es por ello y otras cosas que nace Wayland[30].

[30] Wayland fue iniciado por Kristian Hogsberg, un desarrollador de X.Org, como un proyecto personal en 2008. Es un protocolo de comunicación que especifica la comunicación entre un servidor de pantalla y sus clientes. Wayland se desarrolla como un proyecto gratuito y de código abierto impulsado por la comunidad con el objetivo de reemplazar el sistema de ventanas X11 o Xorg, por un sistema de ventanas moderno, seguro y más simple.

En Wayland, el compositor es el servidor de visualización. Compositor, es un administrador de ventanas que proporciona a las aplicaciones un Buffer fuera de la pantalla para cada ventana. El administrador de ventanas compone los búffers de la ventana en una imagen que representa la pantalla y escribe el resultado en la memoria de visualización. El protocolo Wayland permite al compositor enviar los eventos de entrada directamente a los clientes y permite que el cliente envíe el evento de daños directamente al compositor.

La principal ventaja de Wayland sobre X es que comienza desde cero. Una de las principales razones de la complejidad de X es que, a lo largo de los años, su función ha cambiado. Como resultado, hoy en día, X11 actúa en gran medida como un protocolo de comunicaciones "realmente terrible" entre el cliente y el administrador de ventanas. Wayland también es superior cuando se trata de seguridad. Con X11, es posible hacer Keylogging al permitir que cualquier programa exista en segundo plano y lea lo que está sucediendo con otras ventanas abiertas en el área de X11. Con Wayland, esto simplemente no sucederá, ya que cada programa funciona de forma independiente.

Sin embargo, el sistema X Window todavía tiene muchas ventajas sobre Wayland. Aunque Wayland elimina la mayoría de los defectos de diseño del Xorg, tiene sus propios problemas. A pesar de que el proyecto Wayland ha estado activo durante más de diez años, las cosas no son 100% estables, de ahí que aun sigamos usando Xorg. Aún hoy, la mayoría de los videojuegos y las aplicaciones de gráficos intensivos para Linux todavía se escriben para X11. Además, muchos controladores de gráficos de código cerrado, como los de las GPU NVIDIA, aún no ofrecen soporte completo para Wayland. Desde mi punto de vista aún tendremos Xorg para una década más, aunque ya comiencen a salir algunos

Una distribución de GNU/Linux incluye todos los componentes estándar del sistema Linux, más un conjunto de herramientas administrativas que simplifican la instalación inicial y desinstalación de paquetes del sistema.

GNU/Linux puede funcionar tanto en entorno gráfico (GUI por Graphical User Interface) como en modo consola o terminal -línea de comandos o *Shell*- usando la interfaz de línea de comando (CLI por Command Line Interface) o interfaz de usuario de terminal (TUI por Terminal User Interface). La consola es común en distribuciones para servidores, mientras que la interfaz gráfica está orientada al usuario final del hogar como empresarial. Así mismo, también existen los entornos de escritorio como: GNOME, KDE, LXQt, LXDE, Xfce, Unity, MATE, Cinnamon, Pantheon, Deepin, Budgie, PIXEL, Enlightenment, Trinity, Moksha, Ukui, etc., que son un conjunto de programas conformado por ventanas, íconos y muchas aplicaciones que facilitan el uso de la computadora.

3.1 Interfaz de Usuario

Tanto el programador como el usuario de Linux manejan principalmente el conjunto de programas de sistemas que se han escrito y están disponibles para ejecutarse. Estos programas efectúan llamadas al sistema operativo necesarias para apoyar su función, pero las llamadas al sistema en sí están contenidas dentro del programa y no tienen que ser obvias para el usuario.

GNU/Linux puede funcionar tanto en entorno gráfico[31] (Graphical User Interface, GUI) como en modo línea de comandos (Command-Line Interface, CLI) también conocida como consola o *Shell*. La consola es común en distribuciones para servidores, mientras que la interfaz gráfica está orientada al usuario final del hogar, como empresarial.

Los entornos de escritorio pertenecen a la interfaz gráfica, son un conjunto de programas conformado por ventanas, íconos, imágenes y muchas

proyectos como Sway que se favorecen de las bondades de Wayland.

[31] Un servidor de pantalla en GNU/Linux es un programa que es responsable de la coordinación de entrada y salida de sus clientes, hacia y desde en resto del sistema operativo, y entre el Hardware y el sistema operativo. El servidor de visualización proporciona el marco para un entorno gráfico para que se pueda utilizar el Mouse y el teclado para interactuar con las aplicaciones. El servidor de pantalla se comunica con sus clientes a través del protocolo del servidor de pantalla como: X11, Wayland o Mir. El servidor de visualización es un componente clave en cualquier interfaz gráfica de usuario, específicamente el sistema de ventanas. No debemos confundir el servidor de visualización con el entorno de escritorio. El entorno de escritorio utiliza un servidor de pantalla debajo.

aplicaciones que facilitan el uso de la computadora. Los entornos de escritorio más populares en GNU/Linux son: GNOME, KDE, LXQt, LXDE, Xfce, Unity, MATE, Cinnamon, Pantheon, Deepin, Budgie, PIXEL, Enlightenment, Trinity, Moksha, Ukui, etc. Dependiendo de la distribución se pueden tener uno o más escritorios instalados, por ejemplo en Debian GNU/Linux están disponibles los más usados y si el usuario los instala, puede decidir al iniciar sesión cuál usar.

3.1.1 Interfaz Gráfica de Usuario

La interfaz gráfica de usuario es un tipo de visualización que permite al usuario elegir comandos, iniciar programas y ver listas de archivos y otras opciones utilizando las representaciones visuales (iconos) y las listas de elementos del menú. Las selecciones pueden activarse bien a través del teclado o con el ratón.

Para los autores de aplicaciones, las interfaces gráficas de usuario ofrecen un entorno que se encarga de la comunicación con la computadora. Esto hace que el programador pueda concentrarse en la funcionalidad, ya que no está sujeto a los detalles de la visualización ni a la entrada a través del ratón o del teclado. También permite a los programadores crear programas que realicen de la misma forma las tareas más frecuentes, cómo guardar un archivo, porque la interfaz proporciona mecanismos estándar de control como ventanas y cuadros de diálogo. Otra ventaja es que las aplicaciones escritas para una interfaz gráfica de usuario son independientes de los dispositivos: a medida que la interfaz cambia para permitir el uso de nuevos dispositivos de entrada y salida, como un monitor de pantalla grande o un dispositivo óptico de almacenamiento, las aplicaciones pueden utilizarlos sin necesidad de cambios.

¿Qué es un Entorno de Escritorio? un entorno de escritorio es un conjunto de Software para ofrecer al usuario de una computadora una interacción amigable y cómoda. El Software es una solución completa de interfaz gráfica de usuario, ofrece iconos, barras de herramientas, carpetas, fondos de pantalla y Widgets de escritorio, e integración entre aplicaciones con habilidades como, arrastrar y soltar. En general cada entorno de escritorio se distingue por su aspecto y comportamiento particular, aunque algunos tienden a imitar características de escritorios ya existentes. El primer entorno moderno de escritorio que se comercializó fue desarrollado por Xerox en los

años 80. Actualmente el entorno más conocido es el ofrecido por la familia Windows aunque existen otros como los de: Macintosh (Classic y Cocoa) y de código abierto como: GNOME, KDE, LXQt, LXDE, Xfce, Unity, MATE, Cinnamon, Pantheon, Deepin, Budgie, PIXEL, Enlightenment, Trinity, Moksha, Ukui, etc.

¿Qué son los Gestores de Ventanas? un gestor de ventanas o en inglés Windows Manager, es un programa que controla la ubicación y apariencia de las aplicaciones bajo el sistema X Windows. Las computadoras suelen ofrecer una interfaz gráfica de usuario que facilita la interacción con el sistema operativo. Las plataformas Windows y Macintosh ofrecen métodos de visualización y control de las ventanas e interacción con las aplicaciones, estandarizados por sus vendedores. En cambio el sistema gráfico X Windows, popular en el ámbito de sistemas Unix y similares, como GNU/Linux, permite al usuario escoger entre varios gestores según sus gustos o necesidades. Los gestores de ventanas difieren entre sí de muchas maneras, incluyendo apariencia, consumo de memoria, opciones de personalización, escritorios múltiples o virtuales y similitud con ciertos entornos de escritorio ya existentes. Estos se dividen en 3 tipos, que son los siguientes:

- Stacking: Aquellos que imitan las apariencias y funcionalidades de Windows y Mac OS X, por ende, gestionan las ventanas como pedazos de papel en un escritorio, que pueden ser apiladas unas sobre otras.
- Tiling: Aquellos de tipo "mosaico" donde las ventanas no se superponen, y donde suelen hacerse un uso muy extenso de atajos de teclado, y se obtiene una menor dependencia del uso del ratón.
- Dynamics: Aquellos que permiten alterar dinámicamente el diseño de las ventanas entre mosaicos o flotantes.

Las acciones asociadas al gestor de ventanas suelen ser, abrir, cerrar, minimizar, maximizar, mover, escalar y mantener un listado de las ventanas abiertas. Es también muy común que el gestor de ventanas integre elementos como: el decorador de ventanas, un panel, un visor de escritorios virtuales, iconos y un tapiz.

Entornos de Escritorios más Conocidos:

KDE (https://kde.org)

proyecto que fue iniciado en octubre de 1996 por el programador alemán Matthias Ettrich, quien buscaba crear una interfaz gráfica unificada para sistemas Unix. En sus inicios imitó a CDE (Common Desktop Environment), un entorno de escritorio utilizado por varios Unix. Este es un entorno de escritorio con multitud de aplicaciones e infraestructura de desarrollo para diversos sistemas operativos como GNU/Linux, Mac OS X, Windows, etc. Los principales componentes de Software elaborados por KDE se agrupan bajo el nombre KDE Frameworks, KDE Plasma y KDE Applications. Las aplicaciones KDE están traducidas a aproximadamente 88 idiomas y están construidas con los principios de facilidad de uso y de accesibilidad moderna en mente y funcionan de forma completamente nativa en GNU/Linux, BSD, Solaris, Windows y Mac OS X.

GNOME (https://www.gnome.org)

este proyecto fue iniciado por los programadores mexicanos Miguel de Icaza y Federico Mena, forma parte oficial del proyecto GNU. Nació como una alternativa a KDE bajo el nombre de GNU Network Object Model Environment (Entorno de Modelo de Objeto de Red GNU). Actualmente, GNOME se está traduciendo a 193 idiomas. Está disponible en las principales distribuciones GNU/Linux, incluyendo Fedora, Debian, Ubuntu, Manjaro Linux, Red Hat Enterprise Linux, SUSE Linux Enterprise, CentOS, Oracle Linux, Arch Linux, Gentoo, SteamOS, entre otras. También, se encuentra disponible en Solaris, un importante sistema operativo UNIX y en Sistemas operativos Unix-like como FreeBSD.

Xfce (https://www.xfce.org)

es un entorno de escritorio libre para sistemas tipo Unix como GNU/Linux, BSD, Solaris y derivados. Su objetivo es ser rápido y ligero, sin dejar de ser visualmente atractivo y fácil de usar. Consiste en varios componentes empaquetados por separado que en conjunto proporcionan la funcionalidad completa del entorno de escritorio, pero se pueden seleccionar por separado para que el usuario pueda adaptar el ambiente de trabajo a sus necesidades. Puede ser instalado en varias plataformas como: Linux, NetBSD, FreeBSD, OpenBSD, Solaris, Cygwin y MacOS X, sobre x86, PPC, Sparc, Alpha.

LXDE (https://lxde.org)

es un entorno de escritorio libre para Unix y otras plataformas POSIX[32], como Linux o BSD. El nombre corresponde a "Lightweight X11 Desktop Environment", que en español significa Entorno de escritorio X11 ligero. Es un proyecto que apunta a entregar un nuevo entorno de escritorio ligero y rápido. No está diseñado para ser tan complejo como KDE o GNOME, pero es bastante usable y ligero, y mantiene una baja utilización de recursos y energía. A diferencia de otros ambientes de escritorio, los componentes no se integran firmemente. Al contrario, los componentes son independientes, y cada uno de ellos se puede utilizar independientemente con muy pocas dependencias. Usa Openbox como gestor de ventanas predeterminado y apunta a ofrecer un escritorio ligero y rápido basado en componentes independientes que pueden ser utilizados en otros entornos.

LXQt (https://lxqt.github.io)
es un entorno de escritorio libre y de código abierto para Linux, resultado de la fusión entre los proyectos LXDE y Razor-qt. LXQt conjuga la filosofía de LXDE con las librerías QT, usa el gestor de ventanas del escritorio Razor-qt, es un escritorio muy liviano y es considerado por muchos como el sucesor de LXDE.

Gestores de Ventanas más Conocidos:

Enlightenment (https://www.enlightenment.org)
también conocido simplemente como E, es un gestor de ventanas X11 ligero para UNIX y GNU/Linux. Uno de sus objetivos es llegar a ser un entorno de escritorio completo. Es muy configurable y muy atractivo visualmente. Durante un tiempo fue el gestor de ventanas de GNOME.

IceWM (https://ice-wm.org)
es un gestor de ventanas para el X Windows System gráfico de infraestructura escrito por Marko Macek. Se ha codificado desde cero en C++ y es

[32] Significa Portable Operating System Interface. Consiste en una familia de estándares especificadas por la IEEE con el objetivo de facilitar la interoperabilidad de sistemas operativos. Además, POSIX establece las reglas para la portabilidad de programas. Por ejemplo, cuando se desarrolla Software que cumple con los estándares POSIX existe una gran probabilidad de que se podrá utilizar en sistemas operativos del tipo Unix. Si se ignoran tales reglas, es muy posible que el programa o librería funcione bien en un sistema dado pero que no lo haga en otro.

liberado bajo GNU. IceWM es ligero y personalizable. Se puede configurar a partir de archivos de texto almacenados en un directorio Home del usuario, haciendo fácil la personalización y copia de configuraciones. Posee soporte oficial para menús de GNOME y KDE previamente disponible como un paquete separado.

Windows Maker (https://www.windowsmaker.org)
es un popular gestor de ventanas para X Windows System diseñado para emular NeXT del GUI como OpenStep compatible, ha sido descrito como "uno de los más útiles y universales gestores de ventanas disponibles. Windows Maker tiene la reputación de ser rápido, eficiente y altamente estable y es muy popular entre las soluciones de código abierto para su uso tanto en nuevas como en viejas máquinas. Como con la mayoría de gestores de ventanas, soporta un montón de temas disponibles.

Fluxbox (http://fluxbox.org)
es un gestor de ventanas X basado en Blackbox 0.61.1.Su objetivo es ser ligero y personalizable, y cuenta con un apoyo mínimo de iconos gráficos. Su interfaz de usuario sólo tiene una barra de tareas y un menú al que se puede acceder pulsando con el botón derecho sobre el escritorio. Todas las configuraciones básicas están controladas por ficheros de texto. Fluxbox puede mostrar algunos Eye Candies: colores, gradientes, bordes y una que otra apariencia básica. Las versiones recientes soportan esquinas redondeadas y elementos gráficos. Fluxbox también tiene varias características de las cuales Blackbox carece, incluyendo ventanas con pestañas y un título configurable.

Openbox (http://openbox.org/new/)
es un famoso gestor de ventanas libre para el sistema de ventanas X, licenciado bajo la GNU General Public License. Openbox fue originalmente derivado de Blackbox 0.65.0, pero ha sido totalmente reescrito en el lenguaje de programación C y desde la versión 3.0 no se basa en ningún código de Blackbox. Su sistema de menú tiene un método para utilizar los menús dinámicos.

Otros existen muchos otros gestores de ventanas que pudiéramos mencionar, tales como: 9wm, Awesome, AfterStep, Scwm, Blackbox, Bspwm, Byobu, Cinnamon, Cwm, Deepin, Dwm, Fvwm, Icewm, Jwm, Kahakai,

Lumina, Qtile, Wmii, WindowLab, Ratpoison, Sawfish, Sway, wm2, Wmx, StumpWM, Twm, Waimea, Xmonad, I3, E16.

3.1.2 Línea de Comandos y Órdenes

La mayoría de los usuarios de ordenadores de hoy sólo están familiarizados con la interfaz gráfica de usuario o GUI, los vendedores y los expertos les han enseñado que la interfaz de línea de comandos o CLI es una cosa espantosa del pasado. Es una pena, porque una buena interfaz de línea de comandos es una maravillosa y expresiva forma de comunicarse con el ordenador, muy parecida a lo que el lenguaje escrito es para los seres humanos. Se ha dicho que "las interfaces gráficas de usuario hacen fáciles las tareas fáciles, mientras que las interfaces de línea de comandos hacen posibles las tareas difíciles" y eso es muy cierto aún hoy.

Dado que Linux fue desarrollado desde la familia de sistemas operativos Unix, comparte la misma rica herencia de herramientas de línea de comandos que Unix. Unix saltó a la fama en los primeros años ochenta (aunque fue desarrollado una década antes), antes de que se extendiera la adopción de las interfaces gráficas de usuario, y por eso, se desarrolló una amplia interfaz de línea de comandos en su lugar. De hecho, una de las razones más potentes para que los primeros que utilizaron Linux lo eligieran sobre, digamos, Windows NT, era la poderosa interfaz de línea de comandos que hacía las "tareas difíciles posibles".

Trabajando en Línea de Comandos Las órdenes se pueden agrupar en varias categorías; la mayor parte de ellas están orientadas hacia archivos o directorios. Por ejemplo los programas de sistema que manipulan directorios son *mkdir* para crear un directorio nuevo, *rmdir* para eliminar un directorio, *cd* para cambiar el directorio actual a otro y *pwd* para visualizar el nombre de la ruta absoluta del directorio actual (de trabajo).

El programa *ls* lista los nombres de los archivos del directorio actual. Cualquiera de las más de 20 opciones de *ls* puede hacer que se exhiban también las propiedades de los archivos, por ejemplo, la opción *-l* pide un listado largo, que muestra el nombre de cada archivo, su dueño, la protección, su tamaño, la fecha y hora en que se creó. El programa *cp* crea un archivo nuevo que es una copia de uno ya existente. El programa *mv* cambia de lugar un archivo dentro del árbol de directorios. En la mayor parte de los casos, este movimiento sólo requiere un cambio de nombre de archivo, pero si es

necesario el archivo se copia en su nueva posición y la copia vieja se elimina. Los archivos se eliminan con el programa *rm*.

Para mostrar el contenido de un archivo en la terminal, un usuario puede ejecutar *cat*. El programa *cat* toma una lista de archivos y los concatena, copiando el resultado en la salida estándar, que normalmente es la terminal. Claro que en la terminal el archivo podría exhibirse con demasiada rapidez como para leerse. El programa *more* exhibe el archivo pantalla por pantalla y hace una pausa hasta que el usuario teclea un carácter para continuar con la siguiente pantalla. El programa *head* exhibe sólo las primeras líneas del archivo; *tail* muestra las últimas líneas.

Estos son algunos de los programas que usa Linux, además hay editores de texto (*ed, sed, emacs, nano, vi*, etc.), compiladores (de C, C++, Java, Python, etc.), formateadores de texto (LaTeX, *troff*, etc.), programas para ordenar (*sort*) y comparar (*cmp, diff*) archivos, buscar patrones (*grep, awk*) y muchas otras actividades.

La ejecución de una orden se efectúa con una o más llamadas al sistema operativo. Por lo regular, el Shell ejecuta una pausa cuando se le pide ejecutar una orden, queda en espera a que ésta se termine de ejecutar. Existe una sintaxis sencilla (un signo & al final de la línea de órdenes) para indicar que el Shell no debe esperar hasta que termine de ejecutarse la orden. Una orden que deja de ejecutarse de esta manera mientras el Shell sigue interpretando órdenes subsecuentes es una orden en segundo plano, o que se ejecuta en segundo plano. Los procesos para los cuales el Shell sí espera, se ejecutan en primer plano.

El Shell del sistema GNU/Linux ofrece un recurso llamado control de trabajos (y visualizarlos con los comandos *ps* o *top*) implementado especialmente en el núcleo. El control de trabajos permite transferir procesos entre el primer y segundo plano. Los procesos pueden detenerse y reiniciarse según diversas condiciones, como que un trabajo en segundo plano requiera entradas desde la terminal del usuario. Este esquema hace posible la mayor parte del control de procesos que proporcionan las interfaces de ventanas, pero no requiere Hardware especial. Cada ventana se trata como una terminal, y permite a múltiples procesos estar en primer plano (uno por ventana) en cualquier momento. Desde luego pueden haber procesos de segundo plano en cualquiera de las ventanas.

Cuando uno se inicia en el mundo de GNU/Linux no nota la diferencia

entre:

- Terminal
- Shell
- Prompt
- Línea de comandos

Esto es lo que debes saber al respecto:

Terminal es una aplicación gráfica que ejecuta un Shell por defecto. Puede haber muchas aplicaciones de emulador de terminal como Terminator, Konsole, etc.

Shell es difícil visualizarlo por separado de la terminal. La terminal ejecuta un Shell, generalmente Bash. Al igual que las terminales, también hay varios shells: zsh, bash, fish, etc.

Prompt esto es lo que ve antes del espacio donde escribe los comandos. No existe un estándar establecido para el aviso. En algunas terminales antiguas, simplemente tendría un cursor parpadeante en el lugar donde puede escribir los comandos. En las terminales modernas, el mensaje le brinda información como nombre de usuario, nombre de Host y directorio de trabajo actual.

Línea de comandos no es algo específico de Linux. Cada sistema operativo tiene una interfaz de línea de comandos. Muchos lenguajes de programación tienen interfaces de línea de comandos. Es un término para la interfaz donde puedes ejecutar y ejecutar comandos.

Por lo general, escribimos los comandos en una terminal después del Prompt y el Shell los interpreta.

Emuladores de Terminal Cuando utilizamos una interfaz gráfica de usuario, necesitamos otro programa llamado emulador de terminal para interactuar con el Shell. Si buscamos en nuestros menús de escritorio, probablemente encontraremos uno. KDE usa Konsole y GNOME usa Gnome-terminal, aunque es probable que se llame simplemente "Terminal" en nuestro menú. Hay muchos otros emuladores de terminal disponibles para Linux, pero todos hacen básicamente lo mismo; nos dan acceso al Shell.

Podemos instalar algunos emuladores de terminal usando:

```
# apt install sakura miterm xterm rxvt kitty lilyterm \
guake qterminal eterm roxterm terminator tilix tilda \
terminology yakuake konsole terminix deepin-terminal \
kitty sakura gnome-terminal stterm mate-terminal \
xfce4-terminal lxterminal cool-retro-term
```

El Shell Los programas, tanto los escritos por el usuario como los de sistemas, normalmente se ejecutan con un intérprete de órdenes. El intérprete de órdenes en Linux es un proceso de usuario como cualquier otro y recibe el nombre de Shell (concha o cáscara) por que rodea al núcleo del sistema operativo.

El Intérprete de Órdenes de Consola o Shell es un programa informático, cuya función consiste en interpretar órdenes, y un lenguaje de consola. El más conocido es Bash[33]. Es una Shell de Unix compatible con POSIX[34] y el intérprete de comandos por defecto en la mayoría de las dis-

[33] Su nombre es un acrónimo de Bourne-again Shell ("Shell Bourne otra vez"), haciendo un juego de palabras (Born-Again significa "nacido de nuevo") sobre la Bourne Shell (sh), que fue uno de los primeros intérpretes importantes de Unix.

[34] Significa Portable Operating System Interface. Consiste en una familia de estándares especificadas por la IEEE con el objetivo de facilitar la interoperabilidad de sistemas operativos. Además, POSIX establece las reglas para la portabilidad de programas. Por ejemplo, cuando se desarrolla Software que cumple con los estándares POSIX existe una gran probabilidad de que se podrá utilizar en sistemas operativos del tipo Unix. Si se ignoran tales reglas, es muy posible que el programa o librería funcione bien en un sistema dado pero que no lo haga en otro.

Sin ir más lejos, examinemos en la última versión de la especificación IEEE 1003 (designación formal de los estándares POSIX). Al buscar la referencia sobre el comando du, podemos ver que solamente las opciones -a, -H, -k, -L, -s, y -x son obligatorias. Otras, tales como -c y -h, aparecen en la versión de du provista por el proyecto GNU. Esto significa

tribuciones GNU/Linux, además de Mac OS. También se ha llevado a otros sistemas como Windows y Android.

Algunas alternativas a Bash son: SH (Bourne Shell), TCSH/CSH (C Shell), KSH (Korn Shell), Fish (Friendly Interactive Shell), ZSH (Z Shell), TSCH (TS Shell), Dash (Debian Almquist Shell), DSH (Distributed Shell)[35].

El Shell indica que está listo para aceptar otra orden exhibiendo una señal de espera (*Prompt*) y el usuario teclea una orden en una sola línea. En el Bourne Shell y sus derivados como Bash el *Prompt* que nos permite escribir los diferentes comandos, generalmente termina con el carácter:

- $ para usuario sin privilegios
- # para el administrador, conocido como *root*

Podemos instalar algunos intérpretes de órdenes usando:

```
# apt install csh tcsh ksh fish zsh tsch dash dsh busybox
```

Sintaxis Estándar de Comandos Se ha definido un estándar para comandos POSIX (Portable Operating System Interfaz) por la IEEE, pero no todos los comandos la siguen, Muchos comandos definidos por POSIX tienen una forma moderna y una sintaxis obsoleta, por compatibilidad con sistemas viejos. Por ejemplo el estándar especifica que las opciones de los comandos (no así, los operandos) pueden aparecer en cualquier orden, pero algunos comandos pueden requerir que las opciones aparezcan en un orden particular y muchas veces esto no está mencionado explícitamente el la página del manual del comando.

Un comando consiste en una sucesión de palabras separadas por espacios en blanco. La primera palabra es el nombre del comando y las palabras subsecuentes son sus argumentos u opciones. Los operandos consisten de las opciones del programa, si hay alguna, seguidas de sus operandos, si hay alguno. Las opciones controlan o modifican lo que el comando hace. Los

que si utilizamos estas últimas en un Script desarrollado en Linux e intentamos hacerlo funcionar en FreeBSD o AIX, es muy probable que no funcione como esperamos.

[35] Para conocer cuál Shell estamos usando, abrimos una terminal y tecleamos:

```
$ echo $0
```

operadores especifican nombres de trayectorias o con lo que va a trabajar el comando.

Las opciones se especifican con una sucesión de palabras. Cada palabra es un grupo de opciones o una opción argumento. Las opciones generalmente se denotan por letras. Se pueden escribir en cualquier orden y se pueden combinar varias opciones en un solo grupo (siempre y cuando no reciban ningún argumento). Cada grupo de opciones está precedido por un guión (-). Por ejemplo, los comandos:

```
$ ls -al
$ ls -l -a
```

son equivalentes e indican que el comando *ls* (list archives) sea llamado con las opciones *a* (all files) y *l* (long listing).

Hay otras convenciones comunes para especificar argumentos:

- Dos guiones (- -) indica el final de las opciones. Esto es útil cuando quieres pasar argumentos que empiecen con un - a un comando, por ejemplo:

  ```
  $ rm - - -miArchivo
  ```

es una forma de indicarle que el archivo es *-miArchivo*, también podemos usar:

```
$ rm ./-miArchivo
```

- Un solo guión representa a la entrada estándar en un contexto en que el programa espera una trayectoria. Por ejemplo, el comando:

  ```
  $ diff - miArchivo
  ```

encuentra las diferencias entre la entrada estándar y el archivo *miArchivo*

3.1.3 SubShells

los subShells son un concepto fundamental en las secuencias de comandos Bash que pueden resultar potentes y confusos, especialmente para los principiantes. Le permiten crear entornos de ejecución aislados dentro de sus Scripts, proporcionando una forma de gestionar procesos y ejecutar comandos de forma eficaz sin afectar el entorno del Shell principal.

¿Qué es Exactamente un subShell? un subShell, también conocido como Shell secundario, es una instancia separada del Shell que se genera a partir del proceso de Shell actual. Hereda el entorno y las variables de su Shell principal, pero funciona de forma independiente, lo que permite la ejecución aislada de comandos y Scripts. Cuando se crea un subShell, se ejecuta en un proceso separado, distinto del Shell principal. Esto significa que cualquier cambio realizado en el entorno dentro del subShell, como modificar variables o definir funciones, está aislado y no persiste en el Shell principal después de que termina el subShell.

Creando un SubShell hay varias formas de crear un subShell en Bash, cada una con sus propios matices y casos de uso:

Paréntesis o Llaves los comandos entre paréntesis se ejecutan en un subShell. Esta es una de las formas más comunes y sencillas de crear un subShell en Bash.

```
# Crear un subshell
$ (pwd; ls; whoami)
```

El uso de llaves {...} alrededor de un conjunto de comandos también puede crear una subShell:

```
$ { sleep 3; echo "Hola desde el subshell"; }
# Subshell creado
$ echo "Regresando al shell padre"
```

Sustitución de Comandos

```
# Asigna la salida de un subshell a una variable
$ output=$(pwd; ls; whoami)
```

La sustitución de comandos crea un subShell y captura su salida, que puede asignarse a una variable o usarse en otro comando.

Invocación Explícita del subShell el comando integrado Bash se puede utilizar para iniciar un subShell y ejecutar comandos dentro de él explícitamente. La opción *-c* le permite especificar los comandos que se ejecutarán.

```
# Ejecutar un subshell
$ bash -c "ls; whoami"
```

Relación entre el Shell principal y el subShell el Shell principal y sus subShells tienen una relación jerárquica. Como mencioné, el subShell hereda las variables de entorno, funciones y otras configuraciones del Shell principal, pero cualquier modificación realizada en el entorno dentro del subShell está aislada y no afecta al Shell principal.

Para demostrar este aislamiento, considere el siguiente ejemplo:

```
# shell padre
$ echo "Shell padre: valor de psell $pshell"
# Crear un subshell y modificar pshell
$ (pshell=10; echo "Subshell: Valor de pshell $pshell")
# Checar el valor de pshell en el shell padre
$ echo "Shell padre: valor de pshell $pshell"
```

En este ejemplo, la variable pshell primero se desactiva en el Shell principal, lo que significa que no tiene valor. Dentro del subShell creado por paréntesis(), asignamos el valor 10 a la variable pshell e imprimimos su valor. Sin embargo, después de que finaliza el subShell, el valor de pshell del Shell principal permanece sin establecer, lo que demuestra el aislamiento del entorno del subShell.

Es importante tener en cuenta que los subniveles aíslan el entorno y afectan el alcance de las variables y funciones definidas dentro de ellos. No se puede acceder a las variables y funciones definidas en un subShell fuera de ese subShell, incluso si el subShell es parte de un Script o función más grande.

Cómo Comprobar si se ha Generado un subShell se puede comprobar si el Shell actual es un subShell inspeccionando el valor de la variable de entorno $BASH_SUBSHELL. Esta variable se establece en un valor distinto de cero en los subniveles, lo que indica el nivel de anidamiento.

```
$ echo $BASH_SUBSHELL
# Imprime 0 en el shell padre, no zero en los subshells
```

Otra excelente manera de verificar si ha generado una subShell usando el comando ps –forest. Consideremos ahora el siguiente ejemplo:

```
$ bash
$ bash
$ ps –forest
```

La ejecución del comando ps –forest muestra el anidamiento del subShell. Hablaremos de subShell anidadas en la siguiente sección.

SubShells Anidados los subniveles se pueden anidar, lo que significa que un subnivel puede generar otro subnivel dentro de sí mismo. Cada subShell anidado hereda el entorno de su subShell principal y la variable $BASH_SUBSHELL se incrementa en consecuencia.

```
$ echo "Parent shell: $BASH_SUBSHELL" (
echo "Subshell 1: $BASH_SUBSHELL" (
echo "Subshell 2: $BASH_SUBSHELL" ))
```

Aquí hay otro ejemplo:

```
$ bash
$ bash
$ bash
$ ps –forest
```

Aquí ingresamos el comando bash cuatro veces para crear cuatro subShell. Puede salir de cada subShell usando el comando bash *exit*.

¿Los Scripts de Shell se Ejecutan en subShell? ¡La respuesta es sí! De forma predeterminada, los Scripts de Shell se ejecutan en subShell. Esto significa que cualquier cambio realizado en las variables de entorno u otras configuraciones del Shell dentro del Script no se propaga al Shell principal. Sin embargo, este comportamiento se puede modificar utilizando el comando fuente o el punto (.), que ejecuta el Script en el contexto actual del Shell, permitiendo que cualquier modificación realizada en el entorno se refleje en el Shell principal:

```
# Ejecutar el script en contexto actual del shell
$ source .bashrc
```

Alternativamente, puede usar el comando exec dentro del Script para reemplazar el proceso de Shell actual con el proceso del Script, evitando que el Script se ejecute en un subShell.

```
$ exec bash
$ exec bash
$ exec bash
$ ps –forest
```

Hacer uso de SubShells los subShells se pueden utilizar de varias formas creativas para lograr los comportamientos deseados en los Scripts y comandos de Bash. Veamos algunas formas en que podemos hacer uso de subShell.

Poner las Listas de Procesos en Segundo Plano al incluir una lista de comandos entre paréntesis, puede crear un subShell que ejecute los comandos en segundo plano, lo que le permitirá continuar trabajando en el Shell principal mientras los procesos del subShell se ejecutan simultáneamente.

```
# Subshell corriendo en segundo plano
$ (sleep 10; echo "Esto corre en segundo plano") &
```

Coprocesamiento el coprocesamiento es similar a ejecutar un comando en segundo plano, la única diferencia es que también crean un subShell. Los coprocesos son útiles para tareas que requieren ejecución paralela o comunicación entre procesos.

```
# Creación de un coproceso que dormira 60 segundos
$ coproc sleep 60
```

Procesamiento Paralelo con SubShell los subShells también se pueden utilizar para el procesamiento paralelo, lo que permite ejecutar múltiples tareas simultáneamente. Al encerrar cada tarea entre paréntesis y separarlas con el operador &, puede crear subShell que se ejecuten en paralelo.

```
# Ejemplo de procesamiento en paralelo
$ (task1.sh) & (task2.sh) & (task3.sh) &
# Esperar a que todas las tareas se completen
$ wait
```

En el ejemplo anterior, task1.sh, task2.sh y task3.sh se ejecutan en subShells separadas, ejecutándose en paralelo. El comando de espera garantiza que el Shell principal espere a que se completen todos los procesos del subShell antes de continuar.

Esta técnica puede ser especialmente útil para tareas computacionalmente intensivas o que requieren mucho tiempo, donde la paralelización puede mejorar significativamente el tiempo de ejecución general.

Por qué a Veces Deberías Evitar el uso de subShell en Bash si bien los subShells ofrecen características y funcionalidades valiosas, hay situaciones en las que es posible que desee evitar la creación de subShell innecesarias. La creación de una subShell implica generar un nuevo proceso, lo que puede generar cierta sobrecarga, especialmente cuando se realiza con frecuencia. Además, los cambios realizados en las variables de entorno u otras configuraciones del Shell dentro de un subShell no se propagan al Shell principal, lo que a veces puede provocar comportamientos inesperados o inconsistencias. El uso excesivo de subShells también puede hacer que los Scripts sean más complejos y difíciles de leer, especialmente para aquellos que no están familiarizados con el concepto. Además, cada subShell consume recursos adicionales del sistema, como memoria y descriptores de archivos, lo que puede resultar problemático en entornos con recursos limitados o cuando se trata de una gran cantidad de subShells.

3.2 Hardware en GNU/Linux

El soporte del Hardware en Linux es un asunto complicado, es lo que más problemas da. Linux soporta la mayor parte del Hardware, pero a veces pueden existir problemas[36]:

[36] Mito: Linux/Unix se puede usar para revivir un equipo de cómputo viejo. La realidad es que si bien, hay múltiples distribuciones de Linux/Unix que corren en una gran cantidad de procesadores antiguos y actuales, los Drivers necesarios para reconocer periféricos como tarjetas gráficas, de red alámbrica e inalámbrica, entre muchos otros, no tienen soporte en Linux/Unix, lo cual hará imposible su uso en Linux/Unix. Esto es cierto en cualquier

- si es Hardware muy antiguo, muy moderno o muy raro.
- si es un dispositivo exclusivo para Windows, como los Winmodems (linmodems.org).

Si el Hardware es "de verdad" (no Winmodems), de marca conocida y actual, casi con toda seguridad estará soportado por Linux.

Los instaladores de Linux reconocen prácticamente todo el Hardware durante la instalación, por lo que la mejor manera de evitar problemas con el Hardware es instalarlo desde el principio. Si añadimos Software para nuestro Hardware posteriormente a la instalación nos costará hacerlo funcionar. ¡Incluso puede ser más rápido instalar el sistema desde cero!.

¿Cómo puedo saber si mi Hardware está soportado por Linux (antes de comprarlo y cometer un error de forma irreparable)? Fácil: consultando en internet.

¿Qué sabe Linux de mi Hardware? El Kernel se encarga de la gestión del Hardware usando herramientas como *Udev* (sistema de nombrado del Hardware), *Hotplug* (mecanismo de avisos), *Dbus* (comunicaciones entre procesos) o *Hal* (capa de abstracción de Hardware), y mapea todo el Hardware en archivos de dispositivos ubicados en los directorios */dev* y */sys*.

Algunos comando usados para conocer el Hardware son:

- lscpu - Información de procesador
- lshw - Lista de Hardware en Linux

computadora no importa de cuál generación es el equipo de cómputo. La verdad de todo esto, es que los fabricantes están enfocados en producir Hardware y Drivers que corran en los sistemas operativos con mayor cuota de mercado y por el momento Linux/Unix en equipos personales no son de ellos.

La compatibilidad del Hardware depende en gran medida de la versión de Kernel de GNU/Linux instalado, es de esperarse que en versiones anteriores del Kernel cierto Hardware no se pueda detectar, pero lo contrario también pasa, hay Drivers que solo corren correctamente en versiones anteriores del Kernel y no en las últimas versiones, lo que ocasiona que muchos usuarios se desesperen al tratar de usar sus equipos con GNU/Linux. Y en caso de logar que funcione el Hardware, se fuerza a los usuarios a usar una determinada versión del Kernel (y todas las aplicaciones de la distribución) no actualizable, por la imposibilidad de hacer funcionar el Hardware del equipo en una más moderna con la consiguiente obsolescencia del Software instalado en el equipo.

- hwinfo - Información del Hardware en Linux
- lspci - Lista PCI
- lsscsi - Listar dispositivos SCSI
- lsusb - Lista de los buses usb y detalles del dispositivo
- inxi - Script mega Bash para usuarios no técnicos
- lsblk - Lista de dispositivos de bloque
- df - espacio en disco de los sistemas de archivos
- fdisk - Informa y permite modificar las particiones de disco
- mount - Permite montar y desmontar y ver sistema de archivo montados
- free - Información de la memoria RAM y Swap
- lsmem - Lista los rangos de memoria disponible.
- hdparm - Información de disco duro
- lsmod - Información de los módulos de los drivers cargados al Kernel

Archivos del directorio */proc*, contienen información accesible usando el comando *cat:*

- Información de CPU
- Información del Kernel de Linux
- Dispositivos Sata / SCSI
- Particiones

Componentes de un Sistema GNU/Linux El sistema GNU/Linux se compone de tres cuerpos principales de código, al igual que la mayor parte de las implementaciones de UNIX.

- El núcleo se encarga de mantener todas las abstracciones importantes del sistema operativo, incluidas cosas tales como memoria virtual y procesos.
- Las bibliotecas del sistema definen un conjunto estándar de funciones a través de las cuales las aplicaciones pueden interactuar con el núcleo y que implementan gran parte de la funcionalidad del sistema operativo que no necesita todos los privilegios del código del núcleo.
- Las utilerías del sistema son programas que realizan tareas de administración especializadas individuales. Algunos programas utilitarios pueden invocarse una sola vez para asignar valores iniciales y configurar algún aspecto del sistema; otros llamados demonios[37] podrían ejecutarse de forma permanente, realizando tareas tales como responder a conexiones de red entrantes, aceptar solicitudes de ingreso al sistema desde terminales, o actualizar archivos de bitácora.

Principios de Diseño Unix y posteriormente Linux se diseñaron como sistemas de tiempo compartido. La interfaz estándar con el usuario (el Shell) es sencilla y puede ser sustituida por otra si se desea[38]. El sistema de archivos es un árbol invertido con múltiples niveles, que permite a los usuarios crear sus propios subdirectorios. Cada archivo de datos de usuario es tan solo una secuencia de Bytes.

El sistema UNIX/Linux fue diseñado por programadores para programadores; por ello, siempre ha sido interactivo, y las funciones para desarrollar programas siempre han tenido prioridad. Tales recursos incluyen a los programas *make, gcc, git*, etc.

[37] En sistemas UNIX/LINUX se conoce como demonio o Daemon (Disk And Execution Monitor) a un proceso que se ejecuta en segundo plano del sistema operativo, se ejecuta en todo momento y no posee interacción directa con el usuario, también se le conoce genéricamente como servicio o proceso, del cual no percibimos su ejecución. Un demonio realiza una operación específica en tiempos predefinidos o en respuesta a ciertos eventos del sistema.

[38] Algunos de los distintos tipos de Shell son: Shell Bourne, Shell Zsh, Shell C, Shell Korn, Shell Bourne-Again (mejor conocido como Bash, Bourne again shell), etc.

Los archivos de disco y los dispositivos de Entrada/Salida (E/S) se tratan de la manera más similar posible. Así, la dependencia de dispositivos y las peculiaridades se mantienen en el núcleo hasta donde es posible; aún en el núcleo, la mayor parte de ellas están confinadas a los Drivers de los dispositivos.

Un archivo en Unix/Linux es una secuencia de Bytes. Los diferentes programas esperan distintos niveles de estructura, pero el núcleo no impone ninguna estructura a los archivos. Por ejemplo, la convención para los archivos de texto es líneas de caracteres *ASCII* separadas por un solo carácter de nueva línea (que es el carácter de salto de línea en *ASCII*), pero el núcleo nada sabe de esta convención.

Los archivos se organizan en directorios en estructura de árbol. Los directorios también son archivos que contienen información sobre cómo encontrar otros archivos. Un nombre de camino, trayectoria o ruta de un archivo es una cadena de texto que identifica un archivo especificando una ruta a través de la estructura de directorios hasta el archivo. Sintácticamente, una trayectoria consiste en nombres de archivos individuales separados por el carácter diagonal. Por ejemplo */usr/local/fuente*, la primera diagonal indica la raíz del árbol de directorios, llamado directorio *raíz* o *root*. El siguiente elemento, *usr*, es un subdirectorio de la raíz, *local* es un subdirectorio de *usr* y *fuente* es un archivo o directorio que está en el directorio *local*. No es posible determinar a partir de la sintaxis del nombre de una trayectoria si la *fuente* es un archivo ordinario o un directorio.

Un archivo puede conocerse por más de un nombre en uno o más directorios. Tales nombres múltiples se denominan enlaces, también se manejan enlaces simbólicos, que son archivos que contienen el nombre de una ruta de otro archivo o directorio. Las dos clases de enlaces también se conocen como enlaces duros y enlaces blandos. Los enlaces blandos (simbólicos), a diferencia de los duros pueden apuntar a directorios y pueden cruzar fronteras de sistemas de archivos (apuntar a otros sistemas de archivos) y el sistema operativo trata igualmente todos los enlaces.

El nombre del archivo *"."* en un directorio es un enlace duro al directorio mismo. El nombre de archivo *".."* es un enlace al directorio padre. Por tanto, si el directorio actual es */usr/jlp/programa*, entonces *../bin/wdf* se refiere a */usr/jpl/bin/wdf*.

Los dispositivos de Hardware tienen nombres en el sistema de archivos. El núcleo sabe que estos archivos especiales de dispositivos o archivos especiales son interfaces con dispositivos, pero de todos modos el usuario accede a ellos

prácticamente con las mismas llamadas al sistema que otros archivos.

3.3 Sistema de Archivos y Estructura de Directorios

El Sistema de Archivos de Unix/Linux o cualquier sistema de archivos, generalmente es una capa bajo el sistema operativo la cual maneja el posicionamiento de tus datos en el almacenamiento, sin este el sistema no puede saber dónde empieza y termina un archivo.

Éste consiste en un conjunto de archivos, cada uno de los cuales tiene un nombre, hay tres clases de archivos:

- Archivos regulares, que contienen información.
- Directorios o carpetas, que contienen un conjunto de archivos. Los conjuntos de archivos se identifican por el nombre del directorio que los agrupa.
- Archivos especiales FIFO (First In First Out), algunas veces llamados Pipes, que proveen un canal para comunicación entre procesos independientes y que tienen un nombre asociado.

Como la mayoría de los sistemas operativos modernos, Unix/Linux organiza su sistema de archivos como una jerarquía de directorios. La jerarquía de directorios en un árbol invertido, un directorio especial, root '/', es la raíz de la jerarquía. Un directorio puede contener subdirectorios, archivos regulares y archivos especiales. Unix/Linux trata a los subdirectorios como un tipo particular de archivo. Además Unix/Linux ve a un archivo, no importando su tipo, como una sucesión de Bytes, almacenados en dispositivos como Discos, CD, DVDs o unidades de almacenamiento USB.

Un nombre de archivo puede tener hasta 255 caracteres en la mayoría de los sistemas. y contener cualquier carácter excepto '/' o el carácter NULL; sin embargo, algunos caracteres como '&' y ' ' pueden causar problemas por sus significados especiales para su interpretación en la línea de comandos. Los nombres de archivos son sensibles a las mayúsculas y minúsculas.

En los sistemas Unix/Linux, cada usuario tiene un directorio personal, en el cual almacena sus archivos. Dicho directorio se conoce como el hogar (Home) del usuario. Una vez que entres en sesión, cada programa que se ejecutase encuentra situado en un directorio de trabajo.

Tipos de Sistema de Archivos de GNU/Linux Cuando intentas instalar Linux, notaras que ofrece distintos sistemas de archivos como los siguientes:

> Ext, Ext2, Ext3, Ext4, JFS, XFS, Btrfs, FAT32, exFAT, NTFS y Swap

Así que, ¿qué son estos sistemas de archivos que ofrece Linux?

- Ext: Antiguo y descontinuado debido a sus limitaciones.
- Ext2: Primer sistema de archivos de Linux que permite 2 Terabytes de datos.
- Ext3: Evolución del Ext2, con actualizaciones y retrocompatibilidad[39].
- Ext4: Es más rápido y permite archivos mucho más grandes con una velocidad significativa[40].
- F2FS: (Flash-Friendly File System) es un sistema de archivos que toma en cuenta las características de los dispositivos de almacenamiento basados en memorias Flash NAND como las unidades de estado sólido (SSD) y tarjetas eMMC y SD.
- JFS: Sistemas de archivos hechos por IBM. Funcionan bien con archivos grandes y pequeños, pero existen reportes de fallas y los archivos se corrompen después de un largo tiempo de uso.
- XFS: Sistema de archivos creado por SGI que funciona lento con archivos pequeños.
- Btrfs: Hecho por Oracle, no es tan estable como Ext en algunas distribuciones, pero es un buen reemplazo, si es necesario.
- FAT32: Establecido en 1996, es robusto, versátil pero anticuado, sólo permite guardar archivos de hasta 4 GB.

[39] El único problema que tiene es que los servidores no utilizan este tipo de sistema de archivos debido a que no soporta recuperación de archivos o Snapshots del disco.

[40] Es una muy buena opción para discos de estado sólido SSD, además puedes darte cuenta que cuando intentas instalar cualquier distribución de Linux este es el sistema de archivo por defecto que sugiere Linux.

- exFAT: Es una actualización de FAT32 para acabar con la limitación de 4 GB por archivo.
- NTFS: Es una alternativa a FAT32 sin sus limitaciones, usada en discos duros y unidades externas.
- Swap: Es un espacio de intercambio que es utilizado para almacenar datos temporales, reduciendo así el uso de la RAM, normalmente es del doble del tamaño de la RAM del equipo.

Otras opciones de sistemas de archivos son: ZFS, ReiserFS, UFS, FFS, HFS, APFS. Algunas de sus características de los sistemas de archivo más usados son:

Sistema	Tamaño máx. volumen	Tamaño máx. archivo
$FAT32$	8 TB	4 GB
$NTFS$	1.845^{7} TB	256 TB
$EXT4$	1.153^{6} TB	17.5921 TB

Qué es una partición una partición es una subdivisión realizada por Software del dispositivo de almacenamiento. El sistema operativo considerará en la práctica a cada partición como si fuera un medio de almacenamiento independiente. Esto resulta útil por ejemplo si tenemos datos a los que queremos acceder desde diferentes sistemas operativos, si hay información que queremos ocultar o datos que no queremos que se pierdan al instalar desde cero una nueva versión de Linux.

Breve introducción a las particiones en Linux antes de poder instalar en el disco duro de nuestro ordenador un sistema operativo y, para poder gestionar la información tanto en este como en dispositivos de almacenamiento externos como pendrives o tarjetas de memoria, es necesario realizar una serie de pasos previos.

Aunque muchas veces los discos rígidos vienen listos para usar e incluso con Software utilitario, lo más seguro es que solo estén preparados para Windows. Es entonces cuando tendremos que borrar su contenido (siempre se recomienda hacer una copia de lo que contenga antes de hacerlo).

Una restricción que debemos tener en cuenta es que por razones obvias no podemos modificar la partición del sistema operativo que estamos utilizando. Es el motivo de por qué para realizar cualquier cambio necesitaremos otro

sistema operativo. Es posible utilizar uno instalado en nuestro ordenador o dispositivo extraíble, una distribución Linux en modo Live -en este modo el sistema operativo se ejecuta en la memoria de la computadora por lo que no requiere instalación- diseñada para tareas de mantenimiento o reparación o el medio de instalación de cualquier distribución Linux.

Cualquier medio de almacenamiento con permiso de escritura necesita al menos de una partición de un tamaño igual o menor al de su capacidad. Si queremos que esa partición almacene datos, debe ser formateada con un sistema de archivos.

Una ventaja de las distribuciones Linux sobre Windows es que pueden leer los datos de las particiones que usa este último sin necesidad de Software adicional. Con respecto a la cantidad de particiones que se pueden crear, en la actualidad no existen restricciones, salvo las dadas por la lógica. No tiene sentido crear múltiples particiones que no son lo suficientemente grandes para resultar útiles.

Dijimos que cada partición necesita de un sistema de archivos. Un sistema de archivos es una forma de organizar los datos almacenados. Aunque en un disco pueden convivir múltiples sistemas de archivos, cada partición podrá tener solo uno.

Además de los datos almacenados, los sistemas de archivos incluyen la información necesaria para acceder a estos como su nombre, su tamaño y los permisos de acceso. Se incluye un índice que además de los contenidos señala su localización para permitir acceder a ellos más rápidamente.

Siempre es conveniente antes de cambiar el sistema de archivos de una partición hacer copias de los datos importantes, formatearla y volver a copiar la información. Hay quién dice que se puede cambiar el sistema de archivos sin formatear, pero es un riesgo que no se justifica.

Un sistema de archivos establece la forma de guardar los datos, recopilar información de los mismos y posibilitar el acceso a ellos. Las tablas de particiones indican el tipo y tamaño de particiones presentes en un dispositivo de almacenamiento y su localización. Además, señalan la presencia de sistemas operativos con gestores de arranque.

Tipos de particiones habíamos dicho que las particiones son divisiones creadas mediante Software. Su uso tiene las siguientes ventajas:

- Permite dedicar un dispositivo a diferentes usos.

- Facilita la organización asignando diferentes particiones a distintos tipos de datos
- Permite adecuar el uso del dispositivo de acuerdo a las necesidades del momento.
- Es posible asignar distintos permisos de acceso o cifrar cada una de ellas.

Si tratas de instalar Linux en un ordenador de varios años atrás te encontrarás que no puedes crear más de 4 particiones. Esa restricción no existe en los equipos más modernos ya que utiliza un sistema de tablas de particiones diferente -de todas formas, si recién te estás iniciando, lo mejor es dejar que el instalador se ocupe de todo-.

Tenemos dos tipos de tablas de particiones disponibles. MBR (Master Boot Record) para los equipos más antiguos y GPT (GUID Partition Table) para los más modernos. GPT es superior a MBR no solo porque trabaja con discos de mayor tamaño sino que permite la recuperación en caso de daño físico en el dispositivo.

Para superar sus limitaciones, MBR dispone de dos clases de particiones:

- Partición Primaria: Solo puede haber 4 estando una sola activa. Son útiles para almacenar el sistema operativo con su correspondiente gestor de arranque, la computadora accederá a la partición indicada como activa para completar el procedimiento de arranque.
- Partición Extendida: Para superar el límite de 4 particiones primarias existe la posibilidad de crear una partición extendida que actúa como contenedor para nuestro tercer tipo de partición, la partición lógica.
- Partición Lógica: Las particiones lógicas tienen parcialmente las funciones de las particiones primarias. La principal restricción es que no pueden contener el gestor de arranque. Es por eso que es necesario hacerlo en una partición lógica.

Tanto Windows como los instaladores de las diferentes distribuciones Linux incluyen su propia herramienta para el trabajo con particiones. Los escritorios GNOME y KDE disponen además de sus propias herramientas que pueden venir preinstaladas o estar en los repositorios. En el caso de GNOME se llama Gparted (Que también puede descargarse como una distribución Linux independiente) y en el de KDE Editor de particiones.

Sistema de Archivos en el sistema de archivos de Linux (Ext2/3/4, UFS, ReiserFS, FFS, XFS, Btrfs, etc.) se tiene asociado un elemento en la tabla que guarda a los archivos y directorios dentro del sistema de archivos, que contiene un número entero único. Este número identifica la ubicación del archivo dentro del área de datos llamado *inodo (i-node)*.

Cada *inodo* contiene información de un fichero o directorio. En concreto, en un *inodo* se guarda la siguiente información:

- El identificador del dispositivo que alberga al sistema de archivos.
- El número de *inodo* que identifica al archivo dentro del sistema de archivos.
- La longitud del archivo en Bytes.
- El identificador de usuario del creador o un propietario del archivo con derechos diferenciados.
- El identificador de grupo de un grupo de usuarios con derechos diferenciados.
- El modo de acceso: capacidad de leer, escribir, y ejecutar el archivo por parte del propietario, del grupo y de otros usuarios.
- Las marcas de tiempo con las fechas de última modificación (*mtime*), acceso (*atime*) y de alteración del propio *inodo* (*ctime*).
- El número de enlaces (*Hard Links*), esto es, el número de nombres (entradas de directorio) asociados con este *inodo*.
- Tabla de direccionamiento donde se detallan los bloques del disco duro en que está almacenado el fichero.

podemos conocer la información del *inodo* de un archivo/directorio/enlace usando:

```
$ stat nombre
```

o bien mediante:

```
$ ls -i
```

si conocemos el *inodo* de un archivo y queremos conocer los nombres de archivo de los enlaces, usamos *(por ejemplo 3432343)*:

```
$ find / -inum 3432343
```

y si queremos conocer el número de inodos de disco podemos usar:

```
$ df -i
```

Además están las Dentries que tienen la función de definir la estructura de un directorio, éstas conjuntamente con los inodos, serán los encargados de representar un fichero en la memoria. Notemos que en cada directorio del sistema existen dos entradas especiales:

- La entrada con un punto (.) hace referencia al propio directorio.
- La entrada con dos puntos (..) hace referencia al inodo del directorio que contiene el directorio (el padre).

El área de datos ocupa el resto del disco y es equivalente a la zona de datos en el *FAT*. En esta zona, como su nombre indica, están almacenados los ficheros y directorios de nuestro sistema.

Estructura de Directorios en GNU/Linux Además de los sistemas de archivos que difiere de la de Windows, la estructura de directorios en Linux es distinta, y es necesario conocerla para encontrar ficheros de configuración, instalar ciertos paquetes en el lugar adecuado, localizar las fuentes del Kernel, o la imagen de este, nuestros ficheros personales, entre otros.

De hecho, la Fundación Linux mantiene un estándar de jerarquía del sistema de archivos, este define la estructura de directorios y el contenido de los directorios en las distribuciones Linux. Gracias a este estándar es posible encontrar la misma estructura de directorios en (casi) todas las distribuciones de Linux[41] que a continuación describiremos brevemente:

/ es el directorio principal, la *raíz* o *root*. Contiene el resto de directorios, es decir, todos los demás serían subdirectorios de este (incluso si están en particiones o discos diferentes). Sin duda es el más importante.

[41] Recordemos que Linux se basa en UNIX y, por tanto, toma prestada su jerarquía de sistema de archivos de UNIX. Encontramos una estructura similar en sistemas operativos similares a UNIX, como BSD y MacOS.

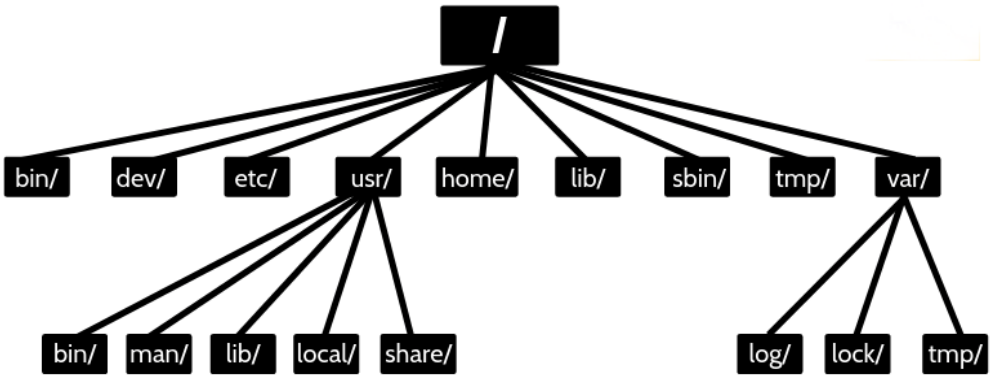

Figura 2: Jerarquía del sistema de archivos de Linux.

/bin es el directorio donde se almacenan los binarios, es decir, los programas que emplea el sistema para labores administrativas como los comandos *cp, echo, grep, mv, rm, ls, kill, xkill, ps, su, tar*, etc.

/sbin la S es de System, y como su nombre indica, aquí se almacenan los binarios o programas que emplea el propio sistema operativo para tareas de arranque, restauración, etc. Por ejemplo, *fsck, mount, mkfs, reboot, swapon.*

/boot es el directorio de arranque, donde está la o las imágenes del Kernel Linux que se cargarán durante el arranque, también directorios y configuración del propio gestor de arranque.

/etc muy importante para el administrador, ya que aquí residen los ficheros de configuración de los componentes del sistema y otros programas instalados.

/dev es un directorio muy especial donde se encuentran los dispositivos de bloques o caracteres, es decir, ficheros que representan la memoria, particiones, discos, dispositivos de Hardware, etc. Ya sabes que en LINUX y UNIX todo es un archivo, y no unidades como en Windows. Por ejemplo, el disco duro o particiones serían */dev/sda, /dev/hda, /dev/scd*, /dev/fd/, etc.

/proc es otro directorio muy especial, más que un directorio es una interfaz por decirlo de un modo sencillo. Y aquí el sistema nos presenta los procesos[42] como directorios numerados con el identificador de procesos *PID* (Process ID). Dentro de cada uno de ellos estaría toda la información necesaria para la ejecución de cada proceso en marcha. Además, encontrarás ficheros de los que extraer información importante, como *cpuinfo, meminfo*, etc. Es precisamente de estos ficheros de los que extraen información algunos comandos que usamos habitualmente, como por ejemplo, cuando hacemos uso de *free* para consultar la memoria disponible, este comando realmente estaría mostrando el contenido de */proc/meminfo* de una forma ordenada.

/media o /mnt son los directorios donde se establecen generalmente los puntos de montaje. Es decir, cuando insertamos algún medio extraíble o recurso de red compartido, etc., que hayamos montado, estaría aquí si lo hemos puesto como punto de montaje. El primero es más específico para medios que se montan de una forma temporal.

/home es el directorio para los usuarios estándar. Por ejemplo, aquí se almacenan dentro de directorios separados (uno para cada usuario con su nombre), los ficheros personales. Por ejemplo, */home/antonio* sería mi directorio personal.

/lib o /lib64 es donde se alojan las bibliotecas necesarias para los programas ejecutables presentes en el sistema. En /lib64 estarían las de las aplicaciones de 64 bits y en */lib* estarían las aplicaciones de 32 bits.

/opt es un directorio que almacenará los paquetes o programas instalados en el sistema que son de terceros. Por ejemplo, si instalamos algún antivirus, Chrome, Arduino IDE o ciertos paquetes grandes, suelen instalarse aquí.

[42] Existen procesos activos y dormidos, procesos huérfanos y procesos zombies.
Los procesos activos son aquellos que están en ejecución en el sistema y los procesos dormidos son aquellos que esperan algún recurso o señal para continuar con su ejecución.
Los procesos huérfanos son aquellos que se siguen ejecutando a pesar que su proceso padre concluyó su operación.
Los procesos zombies es un proceso que ha concluido pero aún están presentes en la tabla de procesos.

/root no hay que confundirlo con /, una cosa es el directorio raíz o *root* y otra muy diferente */root*. En este caso, se puede asemejar a un */home* pero es exclusivo para el usuario *root* o usuario administrador.

/srv almacena ficheros y directorios relativos a servidores que tienes instalados en el sistema, como Web, FTP, CVS, etc.

/sys junto con */dev* y */proc*, es otro de los especiales. Y como */proc*, realmente no almacena nada, sino que es una interfaz también. En este caso, son ficheros virtuales con información del Kernel e incluso, se pueden emplear algunos de sus ficheros para configurar ciertos parámetros del Kernel.

/tmp es el directorio para ficheros temporales de todo tipo. Es empleado por los usuarios para almacenar de forma temporal ciertos ficheros o incluso para almacenar Caché o ciertos ficheros volátiles de navegadores Web, etc. No obstante, hay otro directorio para lo mismo en */var/tmp*.

/var se trata de un directorio con directorios y ficheros que suelen crecer de tamaño, como bases de datos, *logs*, etc. Es precisamente los *logs* o registros del sistema por lo que es más popular este directorio, y allí encontrarás muchísima información de todo lo que ocurre en el sistema: */var/logs/*. Dentro de dicho directorio encontrarás separados por directorios, los *logs* de multitud de Software, incluido el sistema.

/usr son las siglas de *User System Resources*, y actualmente almacena ficheros de solo lectura relativos a utilidades del usuario, como los paquetes que instalamos mediante el gestor de paquetes en nuestra distribución. Dentro hay como una jerarquía de árbol de directorios vistos hasta ahora (casi todos) como si de un segundo nivel se tratase. Vas a encontrar */usr/bin*, */usr/lib*, */usr/sbin*, */usr/src,* etc., que por lo dicho anteriormente y sus nombres, es intuitivo saber lo que almacenan. Por ejemplo en /usr/src es donde permanecerán los ficheros de código fuente.

Ten en cuenta que no todas las distribuciones de Linux siguen este esquema y puede haber pequeñas variaciones, pero si se adaptan al estándar, no tendrás problemas al navegar por la estructura de archivos.

Rutas Absolutas y Relativas cuando se empieza a manejar un intérprete de comandos, una de las cosas que más cuesta es acostumbrarte a encontrar y hacer referencia a elementos del sistema de ficheros. Mientras que en un entorno gráfico tenemos que hacer Clic en carpetas y subcarpetas hasta llegar al elemento deseado, en el intérprete de comandos tendremos que conseguir lo mismo, pero indicando el lugar mediante una cadena de texto compuesta por los nombres de las carpetas que hay que recorrer hasta el lugar donde se encuentra el elemento deseado. Según el sistema cada nombre de carpeta se separa por un carácter especial, que en Linux es la diagonal (/).

Estas rutas serán usadas por los comandos para saber dónde encontrar los elementos sobre los que se tiene que realizar la acción correspondiente[43]. Hay dos formas de utilizar rutas, una es de forma absoluta y la otra de forma relativa. Vamos a explicar la diferencia a continuación:

Rutas Absolutas el sistema de ficheros es una estructura jerárquica que en el caso de Linux tiene una raíz que se indica cuando se pone solamente el carácter diagonal / . En la raíz están los directorios principales del sistema que a su vez tendrán subdirectorios en su interior. Cuando quiero indicar dónde se encuentra un elemento usando una ruta absoluta, tendré que indicar todos los directorios por los que hay que pasar empezando desde la raíz del sistema. O lo que es lo mismo, siempre empezarán por /. Veamos algunos ejemplos:

```
/etc/apt/sources.list
/var/log/syslog
/home/alumno/.bashrc
/usr/bin/
```

estas rutas suelen ser bastante largas, pero tienen como ventaja que funcionan siempre, independientemente del lugar desde el que se ejecute la orden[44].

[43] Por ejemplo, si quiero posicionarme en un directorio determinado, utilizaré el comando cd y para indicar el sitio adonde quiero ir usaré una ruta, por ejemplo cd /home/. El comando cp copia elementos, en este caso necesitaremos dos rutas una para el origen (elemento que quiero copiar) y otra para el destino (elemento nuevo que voy a crear o lugar donde voy a dejar las copias). Por lo tanto podría poner:
cp /etc/passwd /home/copia_passwd.

[44] Es muy recomendable utilizar la facilidad que brinda el BASH de completar el nombre de un elemento del sistema de ficheros pulsando la tecla tabulador. Ahorrará mucho tiempo y errores.

Rutas Relativas las rutas relativas indican el camino para encontrar un elemento, pero basándonos en el directorio desde el que se ejecuta la orden. Son mucho más cortas que las absolutas, pero para saber si son correctas o no, tenemos que saber siempre desde dónde se han utilizado.

Un atajo fundamental para la construcción de rutas relativas es conocer que al escribir .. en la ruta hace referencia al directorio padre. Por lo tanto si ejecuto:

```
$ cd ..
```

estoy dando la orden de cambiar de directorio al padre del actual, es decir, al que está justo antes en la estructura jerárquica. El único elemento que no tiene padre es la propia raíz del sistema (/).

Las rutas relativas harán referencia a un elemento que se encuentre en el directorio desde el que ejecutamos la orden, o usará los dos puntos para ascender a directorios superiores. Siempre que sean correctos, podemos combinarlos de la forma que necesitemos separando cada directorio por una diagonal. Por ejemplo, una ruta correcta podría ser: ../../fotos/personales/

Permisos de Archivos y Directorios GNU/Linux, al ser un sistema diseñado fundamentalmente para trabajo en red, la seguridad de la información que almacenamos en nuestros equipos (y no se diga en los servidores) es fundamental, ya que muchos usuarios tendrán o podrán tener acceso a parte de los recursos de Software (tanto a aplicaciones, como a información) y Hardware que están gestionados en estos equipos de cómputo. ¿Ahora podemos ver la necesidad de un sistema de permisos?

En GNU/Linux, los permisos o derechos que los usuarios pueden tener sobre determinados archivos contenidos en él se establecen en tres niveles claramente diferenciados. Estos tres niveles son los siguientes:

- Permisos del propietario.
- Permisos del grupo.
- Permisos del resto de usuarios (o también llamados "los otros").

Para tener claros estos conceptos, en los sistemas en red siempre existe la figura del administrador, superusuario o root. Este administrador es el encargado de crear y dar de baja a usuarios, así como también, de establecer

los privilegios que cada uno de ellos tendrá en el sistema. Estos privilegios se establecen tanto para el directorio de trabajo (Home) de cada usuario como para los directorios y archivos a los que el administrador decida que el usuario pueda acceder.

Permisos del propietario el propietario es aquel usuario que genera o crea un archivo/carpeta dentro de su directorio de trabajo, o en algún otro directorio sobre el que tenga derechos. Cada usuario tiene la potestad de crear, por defecto, los archivos que quiera dentro de su directorio de trabajo. En principio, él y solamente él será el que tenga acceso a la información contenida en los archivos y directorios que hay en su directorio trabajo o Home -bueno, no es del todo cierto esto, ya que el usuario *root* siempre tiene acceso a todos los archivos y directorios del sistema-.

Permisos del grupo lo más normal es que cada usuario pertenezca a un grupo de trabajo. De esta forma, cuando se gestiona un grupo, se gestionan todos los usuarios que pertenecen a éste. Es decir, es más fácil integrar varios usuarios en un grupo al que se le conceden determinados privilegios en el sistema, que asignar los privilegios de forma independiente a cada usuario.

Permisos del resto de usuarios por último, también los privilegios de los archivos contenidos en cualquier directorio, pueden tenerlos otros usuarios que no pertenezcan al grupo de trabajo en el que está integrado el archivo en cuestión. Es decir, a los usuarios que no pertenecen al grupo de trabajo en el que está el archivo, pero que pertenecen a otros grupos de trabajo, se les denomina resto de usuarios del sistema.

¿cómo puedo identificar todo esto? sencillo, abre una terminal y escribe lo siguiente:

```
$ ls -l
```

entregará varias salidas como esta:

Veamos por partes: El primer carácter al extremo izquierdo, representa el tipo de archivo, los posibles valores para esta posición son los siguientes:

- \- Archivo

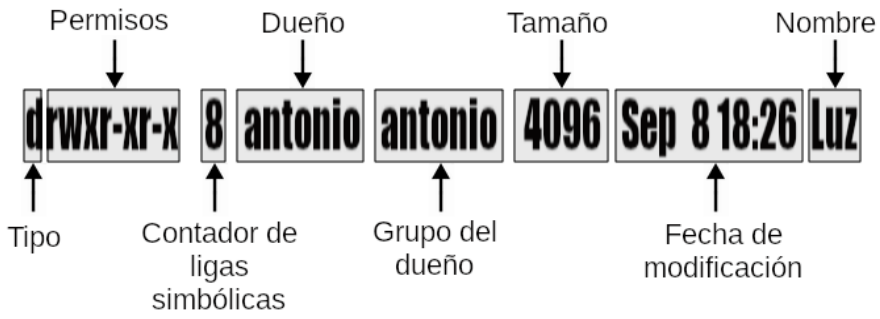

Figura 3: Estructura de permisos en la salida de: ls -l

- d Directorio
- l Liga simbólica
- s Socket (paso de datos entre dos procesos)
- p Pipe
- c Dispositivo de carácter (comunicación de Hardware)
- b Dispositivo de bloque (para el control de dispositivos)

Los siguientes 9 restantes, representan los permisos del archivo y deben verse en grupos de 3 y representan:

- - Sin permiso
- r Permiso de lectura
- w Permiso de escritura
- x Permiso de ejecución

Los tres primeros representan los permisos para el propietario del archivo, los tres siguientes son los permisos para el grupo del archivo y los tres últimos son los permisos para el resto del mundo u otros.

Luego viene el contador de ligas simbólicas, el dueño del archivo, grupo al que pertenece, el tamaño en Bytes, la fecha de última modificación y finalmente el nombre del archivo o directorio.

Permisos Especiales Aún hay otro tipo de permisos que hay que considerar. Se trata del Bit de permisos SUID (Set User ID), el Bit de permisos SGID (Set Group ID) y el Bit de permisos de persistencia (Sticky Bit).

Setuid el Bit Setuid es asignable a ficheros ejecutables, y permite que cuando un usuario ejecute dicho fichero, el proceso adquiera los permisos del propietario del fichero ejecutado. El ejemplo más claro de fichero ejecutable y con el Bit Setuid es:

```
$ su
```

Podemos ver que el Bit está asignado como "s" en:

```
$ ls -l /bin/su
```

Para asignar[45] este Bit a un fichero sería:

```
# chmod u+s /bin/su
```

Y para quitarlo:

```
# chmod u-s /bin/su
```

Setgid el Bit Setid permite adquirir los privilegios del grupo asignado al fichero, también es asignable a directorios. Esto será muy útil cuando varios usuarios de un mismo grupo necesiten trabajar con recursos dentro de un mismo directorio.

Para asignar este Bit hacemos lo siguiente:

```
$ chmod g+s /carpeta_compartida
```

Y para quitarlo:

```
$ chmod g-s /carpeta_compartida
```

[45] Debemos utilizar este Bit con extremo cuidado ya que puede provocar una escalada de privilegios en nuestro sistema.

Sticky este Bit suele asignarse en directorios a los que todos los usuarios tienen acceso, y permite evitar que un usuario pueda borrar ficheros/directorios de otro usuario dentro de ese directorio, ya que todos tienen permiso de escritura.

Podemos ver que el Bit está asignado como "t" en el directorio */tmp*.

Para asignar este Bit hacemos lo siguiente:

```
# chmod o+t /tmp
```

Y para quitarlo:

```
# chmod o-t /tmp
```

Entrada y Salida Estándar los procesos pueden abrir archivos a discreción, pero la mayor parte de los procesos esperan a que estén abiertos tres descriptores de archivos (numerados 0, 1 y 2) cuando inician. Estos descriptores se conocen como entrada estándar (0), salida estándar (1) y error estándar (2). Es común que los tres estén abiertos en la terminal del usuario. Así, el programa puede leer lo que el usuario teclea leyendo la entrada estándar, y puede enviar salidas a la pantalla del usuario escribiendo en la salida estándar. El descriptor de archivo de error estándar también está abierto para escritura, y se usa para los mensajes de error.

Standard Input la Entrada estándar, en inglés *standard input* (mejor conocido como *stdin*) es el mecanismo por el cual un usuario le indica a los programas la información que estos deben procesar. Por omisión, el teclado es la entrada estándar. La entrada estándar representa los datos que necesita una aplicación para funcionar, como por ejemplo un archivo de datos o información ingresada desde la terminal y es representado en la terminal como el tipo 0.

Standard Output la Salida estándar, en inglés *standard output* (mejor conocido como *stdout*) es el método por el cual el programa puede comunicarse con el usuario. Por omisión, la salida estándar es la pantalla donde se ejecutaron las instrucciones. La salida estándar es la vía que utilizan las aplicaciones para mostrarte información, allí podemos ver el progreso o simplemente los mensajes que la aplicación quiera darte en determinado momento y es representado en la terminal como el tipo 1.

Standard Error por último existe un flujo conocido como Error estándar, en inglés *standard error output* (mejor conocido como *stderr*) que es utilizado por las instrucciones para desplegar mensajes de error que surjan durante el transcurso de su ejecución. Al igual que *stdout*, el error estándar será la pantalla donde se procesaron las instrucciones. El error estándar es la forma en que los programas te informan sobre los problemas que pueden encontrarse al momento de la ejecución y es representado en la terminal como el tipo 2.

Redirección Mediante Pipe las tuberías (Pipe) unen la salida estándar de un comando con la entrada estándar de otro, es decir, la salida de un comando se emplea como entrada del siguiente. Para ello se emplea el símbolo pipe "|". El uso de tuberías evita la generación constante de archivos intermedios reduciendo el tiempo de procesamiento.

Redirección Hacia el Dispositivo Nulo en GNU/Linux, */dev/null* es un archivo especial al que se envía cualquier información que quiera ser descartada. Aunque al principio no lo parezca, el uso del dispositivo nulo es muy útil.

Metacarácter o Shell Globbing los metacaracteres o Shell Globbing son caracteres que tienen un significado especial en la línea de comandos, para usar estos caracteres como caracteres ordinarios en la línea de comandos, debes marcarlos de manera especial para que el Shell no los interprete. Por ejemplo:

- La diagonal inversa (\) siempre sirve como carácter de escape para uno o más caracteres que le suceden, lo cual le da a esos caracteres un significado especial. Si un carácter es un metacaracter, el significado especial es el carácter mismo. Por ejemplo \\ usualmente significa una solo \ y \$ el signo de pesos. En estos casos, la diagonal inversa le quita su significado a los caracteres.

- Cuando un texto se encierra entre comillas (" "), la mayoría de los de los metacaracteres que estén en dicho texto aon tratados como caracteres normales, excepto por $, que generalmente indica sustituciones a realizar.

- Otra forma de citar muy similar a la anterior, pero más fuerte es usar comillas sencillas (' ') ya que ni siquiera el carácter $ es interpretado

También existen otros metacaracteres que son comodines que el sistema permite usar para especificar los nombres de archivos que satisfacen el filtro especificado a la hora de buscar, eliminar o filtrar nombres de archivo, estos metacaracteres son: *, ?, [ ] y [^][46].

- ***** Se utiliza para reemplazar cero o más caracteres. Puede ser sustituido por cualquier cadena de caracteres, ejemplo:

  $ ls a*.pdf

- **?** Sustituye un carácter cualquiera, ejemplo:

  $ ls a?chivo.pdf

- [ ] Se usa para definir rangos o conjuntos de caracteres a localizar, para definir los rangos se debe usar el guión -, si son varios caracteres se separan por coma, ejemplo:

  $ ls [Aa]rchivo[0-9].pdf

- [^] o **[!]**Este caso es contrario al anterior, este representa que se busque algo exceptuando lo que se encuentra entre los corchetes, también trabaja con rangos, ejemplo:

  $ls [^A]rchivo.pdf

Cambiar de Usuario en Linux El comando *su* (Switch User) se utiliza para cambiar de usuario cuando estamos dentro de la consola de Linux, ejemplo:

$ su antonio

si delante del usuario ponemos - nos abrirá una nuevo Shell con las preferencias del usuario al que cambiemos, por ejemplo:

[46] Véase también el uso de las secuencias (véase 4.7).

```
$ su - administracion
```

Por otro lado, si usamos el comando *su* sin usuario, nos permitirá ingresar como el usuario administrador del sistema *root* (por defecto), pidiendo la clave o *Password* de dicho usuario, ejemplo:

```
$ su
```

El usuario root en GNU/Linux es el usuario que tiene acceso administrativo al sistema. Los usuarios normales no tienen este acceso por razones de seguridad. Sin embargo, en múltiples sistemas derivados de Debian GNU/Linux no incluye el usuario root (por ejemplo en todos los derivados de Ubuntu). En su lugar, se da acceso administrativo a usuarios individuales, que pueden utilizar la aplicación *sudo* para realizar tareas administrativas. La primera cuenta de usuario que creó en su sistema durante la instalación tendrá, de forma predeterminada, acceso a *sudo*.

Cuando se necesite ejecutar una aplicación que requiere privilegios de administrador, *sudo* le pedirá que escriba su contraseña de usuario normal. Esto asegura que aplicaciones incontroladas no puedan dañar su sistema, y sirve como recordatorio de que está a punto de realizar acciones administrativas que requieren que tenga cuidado.

Para usar *sudo* en la línea de comandos, simplemente escriba *sudo* antes del comando que desea ejecutar, *sudo* le pedirá su contraseña[47]:

```
$ sudo apt update
```

Algunos Comandos para Conocer el Sistema en el que Trabajamos
Siempre es una buena práctica saber que componentes de Hardware se tienen en el equipo en que corremos nuestro GNU/Linux, esto nos puede ayudar a lidiar problemas de compatibilidad cuando se trata de instalar paquetes, controladores en nuestro sistema.

El primer comando a ejecutar es aquel que nos dé información del sistema en el que trabajamos, este es *uname* con la bandera *-a*, que nos dará información de la versión del Kernel, la versión y distribución del sistema operativo y en qué tipo de Hardware es que estamos corriendo nuestro GNU/Linux, para ello usamos:

[47] *Sudo* recordará su contraseña durante un periodo de tiempo (predeterminado a 15 minutos). Esta característica se diseñó para permitir a los usuarios realizar múltiples tareas administrativas sin tener que escribir su contraseña cada vez.

```
$ uname -a
```

Para conocer las características de nuestro CPU, podemos usar varios comandos, entre ellos está *lscpu*, este nos dará la arquitectura del sistema, el número de CPUs, Cores, el modelo de la familia de CPU, Cache del CPU, Threds, entre otros. Para usarlo escribimos:

```
$ lscpu
```

si sólo queremos conocer el número de cores del equipo, usamos:

```
$ nproc
```

Podemos visualizar la memoria total, usada, libre y Swap (de intercambio) de nuestro equipo, usando:

```
$ free -g
```

Para visualizar información sobre los discos duros, unidades de estado sólido, en nuestro equipo, podemos usar *lsblk* que nos reportará dicha información, para ello escribimos:

```
$ lsblk -a
```

o bien podemos usar el comando *df* o en modo administrador a comando *fdisk*, escribiendo:

```
$ df -h
# fdisk -l
```

Si necesitamos conocer la información sobre los controladores USB y todos los dispositivos que están conectados a ellos, usamos:

```
$ lsusb
```

En caso de necesitar conocer los dispositivos PCI que pueden incluir puertos USB, tarjetas gráficas, adaptadores de red, entre otros, más los dispositivos que están conectados a ellos usamos:

```
$ lspci
```

Si lo que deseamos es un listado detallado del Hardware de nuestro equipo de cómputo, entonces podemos usar cualquiera de estos comandos (que previamente deberemos instalar):

```
# lshw
# dmidecode
# hwinfo
```

Así como podemos conocer las características de nuestro equipo de cómputo, podemos también conocer todos los comandos (-c), alias (-a), comandos internos (-b), palabras reservadas Bahs (-k) y funciones de Bash (-A function) disponibles para el usuario, para ello usamos:

```
$ compgen -c
$ compgen -a
$ compgen -b
$ compgen -k
$ compgen -A function
```

Correr Múltiples Comandos en uno Solo Supongamos que se tienen que ejecutar varios comandos uno tras otro -sin conocer si el comando anterior fue exitoso para ejecutar el nuevo-, para este propósito se usa el separador ";", de esta manera se pueden ejecutar una serie de comandos en una línea, ejemplo:

```
$ mkdir tmp ; ls ; cd tmp ; ls
```

en este ejemplo se crea en directorio, luego visualiza el contenido del directorio, para luego cambiar de directorio y finalmente visualiza el contenido del directorio recién creado.

Si necesitamos ejecutar múltiples comandos en uno solo, sólo si el comando anterior fue exitoso, se usa el separador "&&", ejemplo:

```
$ mkdir tmp && ls && cd tmp && ls
```

Notemos que si ejecutamos:

```
$ mkdir tmp & ls & cd tmp & ls
```

el resultado obtenido en nada se parece a lo que esperamos, ya que los comandos pasarán a ser ejecutados en segundo plano "&" y sin orden alguno, por lo que el resultado no será el deseado.

Ya vimos cómo usar un solo comando para ejecutar varios comandos. Pero, ¿qué debo hacer en caso que el primer comando no se ejecute correctamente?, ¿deseo ejecutar el comando siguiente?. Podemos usar || para que si el primer comando falla se ejecute el segundo, pero si no falla no ejecutará el segundo comando, por ejemplo:

```
$ cd miDirectorio || mkdir miDirectorio && cd miDirectorio
```

También podemos combinar los comandos && y || los cuales se comportarán como el operador ternario de C y C++ (condición? expresión_verdadera; expresión_falsa):

```
$ comando1 && comando2 || comando3
```

por ejemplo:

```
$ [-f archivo.txt ] && echo "Archivo existe" || echo "Archivo
no existe"
```

podemos combinar ;, && y || para correr múltiples comandos. Como por ejemplo:

```
$ sudo apt update && sudo apt upgrade && sudo apt clean
```

Argumentos Extendidos xargs (argumentos extendidos) lee datos de la entrada estándar (esto es, Standard Input) y los convierte en comandos. El comando o comandos en cuestión se pasan al segundo comando como parámetro o argumento y a continuación se ejecuta/n una o varias veces. Si se prescinde de un comando especial como parámetro, xargs utiliza automáticamente el comando echo.

¿Cuál es la sintaxis de xargs en Linux? si quieres usar el comando xargs en la terminal, utiliza la sintaxis:

```
$ Primer_Comando | xargs [Opciones] [Segundo_Comando]
```

de esta forma el segundo comando se ejecuta con los argumentos del primero.

¿Cuáles son las opciones del comando xargs? hay numerosas opciones para xargs. A continuación, las más importantes:

-0 o -null: con esta opción, cada carácter se toma literalmente y el carácter NULL separa los argumentos.

-a o -arg-file: con esta opción, los argumentos se leen de un archivo en vez de la entrada estándar.

-d o -delimiter: con esta opción, también cada carácter se toma literalmente. La separación la marca el carácter delimitador y no los espacios en blanco.

-p o -interactive: con esta opción, antes de cada ejecutar xargs, se pregunta si se debe proceder.

Ejemplo de xargs la mejor forma de mostrar la funcionalidad del comando xargs es con algunos ejemplos.

```
$ find . -name "*.txt" | xargs rm
```

en este ejemplo, xargs se utliza con el comando de find y el comando de rm. Como resultado, todos los archivos con la extensión .txt se eliminan.

```
$ find . -name "*.txt" | xargs grep "Factura"
```

en este ejemplo, xargs ayuda a encontrar todos los archivos que contienen la palabra "Factura". Para ello, además de find se usa también el comando grep.

Instalar Paquetes Saber cómo instalar paquetes y programas en Linux y cualquiera de sus distribuciones es uno de los temas más complejos para los nuevos usuarios. El método más común para instalar un nuevo Software en Linux es a través de la Terminal. No obstante, los comandos varían según el tipo de gestor de paquetes que posee el sistema.

Uno de los apartados que debes tener en cuenta antes de instalar un paquete o programa en Linux, es conocer qué es un sistema de gestión de paquetes. En términos simples, hace referencia a un conjunto de formatos de archivos y herramientas empleadas para actualizar, instalar o desinstalar

algún Software en el sistema operativo[48]. En la actualidad, se divide en dos grandes sistemas de gestión de paquetes: Debian y Red Hat.

Distribuciones como Fedora, Mandriva, SuSE, CentOS, entre otros emplean el sistema de gestión de Red Hat, que se caracteriza por utilizar la extensión de archivos (*.rpm*). Por otro lado, las distribuciones como Ubuntu, Peppermint, Linux Mint, y muchos otros hacen uso del sistema *dpkg* de Debian, el cual se representa con la extensión (*.deb*) en sus archivos. Ambos sistemas de gestión trabajan de forma diferente para mantenerse activos en el dispositivo y, a su vez, gestionar las descargas.

Los procesos de instalación entre un sistema de gestión y otro difieren en diversos aspectos, especialmente en los comandos empleados. Por ello es fundamental repasar cuáles son los comandos utilizados para instalar un Software en Linux según el sistema de gestión de paquetes. En el caso de Debian GNU/Linux , puedes hacer uso del programa *apt-get* para instalar el programa deseado desde un repositorio[49]. Puedes escribir únicamente *apt* o utilizar *apt-get* en el comando.

Por otro lado, también utilizar el programa *dpkg* para la instalación de Software con extensión (*.deb*). Por otro lado, los sistemas basados en (*.rpm*) hacen uso de los comandos esenciales para las distribuciones de Linux Red Hat. Estos son *yum* y *dnf*. En el caso de *yum*, el usuario puede instalar o actualizar programas directamente desde el repositorio oficial de Linux, o bien desde un repositorio de terceros. Mientras que el comando *dnf* facilita la administración de programas.

Para Debian GNU/Linux usamos:

```
# apt install paquete
# dpkg -i paquete
```

En Ubuntu y derivados se usa:

[48] El usuario root en GNU/Linux es el usuario que tiene acceso administrativo al sistema, los usuarios normales no tienen este acceso por razones de seguridad. Sin embargo, en múltiples sistemas no incluye el usuario root. En su lugar, se da acceso administrativo a usuarios individuales, que pueden utilizar la aplicación *sudo* para realizar tareas administrativas. La primera cuenta de usuario que se creó en su sistema durante la instalación tendrá de forma predeterminada acceso a *sudo*.

[49] Un repositorio es un lugar en la red que siempre está actualizado desde donde nuestra distribución de GNU/Linux puede buscar y descargar fácilmente todo tipo de programas y herramientas para su instalación en nuestro equipo.

```
$ sudo apt install paquete
$ sudo dpkg -i paquete
```

Para openSUSE:

```
$ sudo rpm -Uvh paquete
$ su zypper intall paquete
```

En Fedora se usa:

```
$ sudo rpm -Uvh paquete
$ su -c 'yum install paquete'
# yum install paquete
```

Para Mandriva:

```
$ sudo rpm -Uvh paquete
$ su urpmi *.rpm
```

En Gentoo, FreeBSD se usa:

```
# emerge -vq paquete
```

Para Red Hat, Fedora o CentOS se usa:

```
$ sudo yum install paquete
$ sudo rpm -i paquete
$ sudo dfn install paquete
# yum install paquete
# dfn install paquete
```

En Archn Manjaro Linux se usa:

```
# pacman -s paquete
```

Para paquetes SNAP:

```
$ snap install paquete
```

Instalar Paquetes en Debian y Derivados Una de las funciones del usuario administrador es hacer la instalación, actualización y borrado de paquetes, el comando más usado actualmente es apt (Advanced Packaging Tool, herramienta avanzada de empaquetado que es una interfase del paquete *apt-get*), es un programa de gestión de paquetes *.deb* creado por el proyecto Debian, *apt* simplifica en gran medida la instalación, actualización y eliminación de programas en GNU/Linux. Existen también programas que proporcionan estos mismos servicios como: *apt-get*, *aptitude*, *tasksel* y *dpkg*.

Una vez siendo el usuario *root*, podemos solicitar la descarga de las actualizaciones disponibles, usando:

```
# apt update
```

para conocer los paquetes que se necesitan actualizar, usamos:

```
# apt list –upgradeable
```

para actualizar los paquetes instalados en el sistema[50], usamos:

```
# apt upgrade
```

algunas veces ciertos paquetes ya no se usarán al actualizar el sistema, para borrarlos usamos:

```
# apt autoremove
```

para buscar paquetes que podemos instalar, usamos:

```
# apt search nombre
```

para conocer las características de un paquete a instalar, usamos:

```
# apt show nombre
```

para instalar uno o más paquetes, usamos:

[50] Podemos hacer la actualización en un solo comando, usando:

```
# apt update && apt upgrade && apt clean
```

```
# apt install paquete
```

para remover uno o más paquetes, usamos:

```
# apt purge paquete
```

al final, se solicita el borrado de los archivos *.deb* de los paquetes descargados (Caché de descarga), mediante:

```
# apt clean
```

podemos conocer todos los paquetes *.deb* instalados en el sistema, usando:

```
$ apt list –installed
```

o

```
$ dpkg -l
```

Existen otro tipo de paquetes para GNU/Linux que son multiplataforma como son: *Snap, Flatpak, Pyhon 3*. Para conocer si tenemos alguno de ellos instalados usamos:
para conocer todos los paquetes *Snap* instalados en el sistema, usamos:

```
$ snap list
```

para conocer todos los paquetes *Flatpak* instalados en el sistema, usamos:

```
$ flatpak list
```

para conocer todos los paquetes *python3* instalados en el sistema, usamos:

```
$ pip3 list
```

3.4 Trabajando en Línea de Comandos

GNU/Linux es un potente sistema operativo visual y de línea de comandos[51]. En esta última se tiene una potente herramienta, en ella se encuentran desde los comandos básicos hasta los más avanzados[52]. Los comandos que se pueden usar son de dos tipos: los pertenecientes al GNU Core Utilities y los externos (que vienen instalados en la distribución de Linux o debe instalar el usuario), algunos de ellos son:

Para manipulación de archivos y directorios[53]

> *ls, pwd, cd, mkdir, rmdir, cp, mv, rename, rm, ln, unlink, cat, touch, cmp, diff, wc, tail, head, more, less, paste, cut, tr, fold, nano*

Comandos generales

> *man, help, info, whatis, which, whereis, clear, w, time, whoami, date, cal, uptime, uname, df, du, free, bc, history, echo*

Administración y Permisos

> *chmod, chown, chgrp, su, useradd, usermod, deluser, passwd, lsattr, chattr, id*

[51] Android tiene la base de Linux, por ello en cualquier dispositivo que soporte dicho sistema operativo es posible instalar una aplicación para acceder a la terminal de línea de comandos —por ejemplo ConnectBot—, y en ella podemos correr los comandos que mostramos en esta sección.

[52] En la Web se puede obtener acceso a diversos proyectos que ponen a disposición del usuario la documentación de una gran variedad de comandos de Linux, algunos de estos proyectos son:
http://man7.org/linux/man-pages/
https://linux.die.net/man/
https://www.kernel.org/doc/man-pages/

[53] Existe la opción de gestionar ficheros comprimidos, gracias a los comandos *zgrep, zegrep, zless, zmore, zdiff, zcmp, zcat*. Como te puedes dar cuenta la función de cada uno de estos comandos será la misma que su homónimos sin «z» (*grep, egrep, less, more, diff, cmp, cat*) pero para ficheros comprimidos con *gzip* y extensión *.gz*. De forma análoga para archivos comprimidos usando *bz2*, están los comandos *bzgrep, bzegrep, bzless, bzmore, bzdiff, bzcmp, bzcat* y para archivos comprimidos usando *xz*, están los comandos *xzgrep, xzegrep, xzless, xzmore, xzdiff, xzcmp, xzcat*.

Búsqueda

find, grep, locate

Respaldo

tar, gzip, bzip2, zip, unp

Varios

file, stat, type, ps, kill, killall, pgrep, pwdx, awk, sort, sed, md5sum, sleep, watch

Para monitorear el desempeño

lscpu, free, top, whowatch, dstat, vmstat, iotop, iostst, lsof, lsusb, tcpdump, nmcli, ip

A continuación detallamos el uso de varios de estos comandos que se ejecutan en la línea de comandos de GNU/Linux o Terminal[54]. Hay que recalcar que cada comando tiene una gran variedad de opciones, pero la descripción completa de cada comando y opciones de este, se escapa de nuestros fines, por ello si se necesita conocer la referencia completa de dichos comandos hay varias maneras de obtenerla, entre otras haciendo uso de *man, help, info o whatis* aplicado al comando de nuestro interés, por ejemplo:

```
$ man ls
```

[54]Existen varios atajos de teclado que facilitan el navegar en la terminal de comandos, entre los que destacan:
CTRL L Limpia el contenido de la terminal
CTRL C Concluye el programa que está en ejecución
CTRL D Concluye la sesión en la terminal cerrando esta
SHIFT Page Up/Down Navega en la terminal una página arriba o abajo
CTRL A Posiciona el cursor al inicio de la línea
CTRL E Posiciona el cursor al final de la línea
CTRL U Borra lo que está a la izquierda del cursor
CTRL K Borra lo que está a la derecha del cursor
CTRL W Borra la palabra a la derecha del cursor
CTRL Y Pega lo que se quitó con CTRL U, K, W
TAB Autocompleta el nombre de archivo o comando
CTRL R Permite buscar dentro del historial de comandos
!! Permite repetir el último comando
CTRL Z Detiene la ejecución del comando actual (permite continuar la ejecución con *fg* en primer plano o *bg* en segundo plano)

Manipulación de Archivos y Directorios

ls (de listar), permite listar el contenido de un directorio o fichero. La sintaxis es:

```
$ ls /home/directorio
```

el comando ls tiene varias opciones que permiten organizar la salida, lo que resulta particularmente útil cuando es muy grande. Por ejemplo, puedes usar *-a* para mostrar los archivos ocultos, *-i* para mostrar el inodo[55], *-l* para mostrar los usuarios, permisos, tamaño en Bytes y la fecha de los archivos; *-S* ordena por tamaño, *-R* recursivo, *-r* en orden inverso, *-t* ordenados por fecha de modificación, *-h* muestra el tamaño en unidades fáciles de leer -como KB, MB o GB-, *-F* les adiciona una / al final del nombre de los directorios, *-d* sólo muestra directorios, -X ordenar por extensión, *-n* muestra *UID* y *UIG* de los archivos y directorios. Así como para todos los comandos Linux, estas opciones pueden combinarse, terminando en algo como:

```
$ ls -lha /home/directorio
```

por ejemplo para mostrar primero los archivos recientemente modificados, usamos:

```
$ ls -lt
```

o los más recientes al final, usamos:

```
$ ls -ltr
```

podemos pedir que sólo liste los directorios en la trayectoria actual:

```
$ ls -d */
```

que nos muestre aquellos archivos que coincidan con un patrón:

```
$ ls archivo*
$ ls *.txt
$ ls archivo?txt
$ ls archivo-{01,03,05}.txt
$ ls archivo-0[135].txt
```

[55] El inodo es un registro en el disco que contiene la información del archivo como su propietario, tamaño, fecha de creación, ubicación, entre otros.

pwd (de print working directory o imprimir directorio de trabajo), es un comando que imprime nuestra ruta o ubicación al momento de ejecutarlo, así evitamos perdernos si estamos trabajando con múltiples directorios y carpetas. Su sintaxis sería:

```
$ pwd
```

cd (de change directory o cambiar directorio), es como su nombre lo indica el comando que necesitarás para acceder a una ruta distinta de la que te encuentras. Por ejemplo, si estas en el directorio /home y deseas acceder a /home/ejercicios, escribimos:

```
$ cd /home/ejercicios
```

teclear el comando *cd* solo regresa al directorio home del usuario (lo mismo hace al teclear cd ~), teclear el comando *cd* - retorna al último directorio antes de hacer cambio de directorio, si estas en /home/ejercicios y deseas subir un nivel (es decir ir al directorio /home), ejecutas:

```
$ cd ..
```

mkdir (de make directory o crear directorio), crea un directorio nuevo tomando en cuenta la ubicación actual. Por ejemplo, si estas en /home y deseas crear el directorio ejercicios, sería:

```
$ mkdir /home/ejercicios
```

mkdir tiene una opción bastante útil que permite crear un árbol de directorios completo que no existe. Para eso usamos la opción *-p*:

```
$ mkdir -p /home/ejercicios/prueba/uno/dos/tres
```

o podemos pedir que cree múltiples directorios simultáneamente:

```
$ mkdir {uno, dos, tres}
```

rmdir (de remove directory o borrar directorio), borra un directorio vacío:

```
$ rmdir /home/ejercicios
```

o podemos pedir que borre múltiples directorios vacíos simultáneamente:

```
$ rmdir {uno, dos, tres}
```

cp (de copy o copiar), copia un archivo o directorio origen a un archivo o directorio destino. Por ejemplo, para copiar el archivo prueba.txt ubicado en /home a un directorio de respaldo, podemos usar:

```
$ cp /home/prueba.txt /home/respaldo/prueba.txt
```

en la sintaxis siempre se especifica primero el origen y luego el destino. Si indicamos un nombre de destino diferente, *cp* copiará el archivo o directorio con el nuevo nombre.

```
$ cp /home/prueba.txt /home/respaldo/prueba01.txt
```

El comando también cuenta con la opción *-r* que copia no sólo el directorio especificado sino todos sus directorios internos de forma recursiva. Suponiendo que deseamos hacer una copia del directorio */home/ejercicios* que a su vez tiene las carpetas *ejercicio1* y *ejercicio2* en su interior, en lugar de ejecutar un comando para cada carpeta, ejecutamos:

```
$ cp -r /home/ejercicios /home/respaldos/
```

también podemos usar *-u* para copiar aquellos archivos que no existen o son nuevos en el directorio destino:

```
$ cp -u /home/ejercicios/*.dat /home/respaldos/
```

Algunas otras opciones son: *-a* copia el archivo con la misma configuración de permisos y metadatos que el original, *-b* Crea una copia en la memoria intermedia (también llamada buffer) si el archivo original y el destino tienen el mismo nombre, pero diferente contenido, *-d* copia los enlaces simbólicos, *-f* obliga a sobrescribir al copiar, -i pide permiso antes de sobrescribir archivos con el mismo nombre, *-l* crea un enlace duro en lugar de

una copia, *-n* los archivos existentes nunca se sobrescribirán, *-p* los atributos del archivo original se heredan al copiar, *-P* los enlaces simbólicos se guardan como tales al copiar, *-R* los directorios, incluidos los subdirectorios, se copian de forma recursiva, *-s* crea un enlace simbólico para el archivo original, *-S* sobrescribe un sufijo de backup al copiar con –backup, *-u* copia el archivo solamente si el archivo de destino es más antiguo que el original.

mv (de move o mover), mueve un archivo a una ruta específica, y a diferencia de *cp*, lo elimina del origen finalizada la operación. Por ejemplo:

```
$ mv /home/prueba.txt /home/respaldos/prueba2.txt
```

al igual que *cp*, en la sintaxis se especifica primero el origen y luego el destino. Si indicamos un nombre de destino diferente, *mv* moverá el archivo o directorio con el nuevo nombre.

Si queremos solo cambiar la extensión de un archivo podemos usar:

```
$ mv archivo.{old,new}
```

rename este comando[56] permite renombrar un grupo de archivos[57], por ejemplo deseamos renombrar todos los archivos con extensión .htm y reemplazarlos con la extensión .html, entonces usamos:

```
$ rename .htm .html *.htm
```

si tenemos los directorios dir1, dir2, dir3 y los queremos renombrar como dir001, etc., usamos:

```
$ rename -v dir dir00 dir?
```

[56] En caso de no estar instalado, lo instalamos usando:

```
# apt install rename
```

[57] El comando mmv permite copiar, mover, hacer ligas simbólicas de múltiples archivos, se instala usando:

```
# apt install mmv
```

rm (de remove o remover), es el comando necesario para borrar un archivo o directorio. Para borrar el archivo *prueba.txt* ubicado en */home*, ejecutamos:

```
$ rm /home/prueba.txt
```

En algunos casos necesitamos borrar el contenido de un directorio menos un archivo, podemos usar:

```
$ rm -v !("archivo.txt")
```

o borrar todo el contenido de un directorio menos algunos archivos, podemos usar:

```
$ rm -v !("archivo1.txt"|"archivo2.txt")
```

o borrar todo el contenido de un directorio menos, por ejemplo los *.zip, usar:

```
$ rm -v !(*.zip)
```

o borrar todo el contenido de un directorio menos, por ejemplo los *.zip y *.txt, usar:

```
$ rm -v !(*.zip|*.txt)
```

Este comando también presenta varias opciones. La opción *-r* borra todos los archivos y directorios de forma recursiva. Por otra parte, *-f* borra todo sin pedir confirmación. Estas opciones pueden combinarse causando un borrado recursivo y sin confirmación del directorio que se especifique. Para realizar esto en el directorio respaldos ubicado en el */home*, usamos:

```
$ rm -rf /home/respaldos
```

Este comando es muy peligroso, por lo tanto es importante que nos documentemos bien acerca de los efectos de estas opciones en nuestro sistema para así evitar consecuencias nefastas.

ln permite crear enlaces a los archivos, tanto duros (Hard Links) como simbólicos *-s* (Soft Links). En pocas palabras, un enlace simbólico es como un acceso directo en Windows o un alias en Mac OS X mientras que un enlace duro es un nombre diferente para la misma información en disco. Todos los archivos que apuntan a un mismo enlace duro, comparten el *inodo* -este es un registro en el disco que contiene la información del archivo como su propietario, tamaño, fecha de creación, ubicación, entre otros-.

Para crear un enlace duro usamos:

```
$ ln archivo_origen nombre_enlace
```

para conocer el *inodo* de un archivo, usamos:

```
$ ls -li archivo
```

para conocer todos los archivos que apuntan a un mismo enlace duro cuyo *inodo* conozcamos, usamos:

```
$ find / -inum <inodo>
```

y para conocer a todos los archivos que comparten inodo, usamos:

```
$ find . -type f -printf '%10i %p\n' | sort | uniq -w 11 -d -D |
less
```

En caso de que se necesite actualizar un enlace duro con otro archivo podemos usar:

```
$ ln -f nuevo enlace
```

Para crear un enlace simbólico, hacemos:

```
$ ln -s nombre nombre_enlace
```

si bien para conocer a donde apunta un enlace simbólico podemos usar *ls -l*, también podemos usar:

```
$ readlink nombre_enlace
```

en caso de necesitar actualizar un enlace simbólico existente con otro archivo podemos usar:

```
$ ln -fs nuevo ligaexistente
```

En ciertas aplicaciones (como en Dockers), los enlaces simbólicos no son vistos o permitidos, una opción para corregir esto es usar el comando mount para montar:

```
# mount –bind nombre nombre_enlace
```

y el comando umount para desmontar:

```
# umount nombre_enlace
```

Si queremos conocer el número de inodos (usados y libres) del disco podemos usar:

```
$ df -i
```

Si por alguna razón el archivo al que apunta la liga simbólica se borra, esto genera una liga rota, para encontrar las ligas rotas en nuestro árbol de archivos, usamos:

```
$ find . -xtype l
```

y si queremos conocer todas las ligas y a dónde apuntan, usamos:

```
$ find . -type l -print | xargs ls -ld | awk '{print $9 $10 $11}'
```

unlink sirve para remover un archivo, en caso de que el archivo sea un enlace creado por *ln*, sólo borra el enlace no el archivo, por ejemplo:

```
$ unlik nombre_enlace
```

cat (de concatenar) nos permite visualizar el contenido de uno o más archivos de texto sin la necesidad de un editor. Para utilizarlo solo debemos indicarlo junto al archivo(s) que deseamos visualizar:

```
$ cat prueba.txt
```

podemos pedir que enumere cada línea del archivo mediante:

```
cat -n prueba.txt
```

además podemos usar -e para que nos muestre un $ al final de cada línea y -T para que muestre los tabuladores con el carácter ^I.

También está el comando *tac* que muestra primero la última línea, hasta la primera línea del archivo:

```
$ tac prueba.txt
```

touch crea un archivo vacío, si el archivo existe sólo actualiza la hora de modificación. Por ejemplo, para crear el archivo prueba1.txt en */home/antonio/*, sería:

```
$ touch /home/antonio/prueba1.txt
```

cmp compara el contenido de dos archivos y devuelve 0 si los archivos son idénticos ó 1 si los archivos tienen diferencias. En caso de error devuelve -1.

```
$ cmp -s archivo1 archivo2
```

también puede mostrar algo de información sobre las diferencias (otra comando más completo es *comm*) pero para un reporte más detallado tenemos el siguiente comando:

diff al igual que *cmp*, compara el contenido de dos archivos pero en lugar de devolver un valor imprime en pantalla un resumen detallado línea a línea de las diferencias. Ejecutarlo es tan simple como:

```
$ diff archivo1.txt archivo2.txt
```

si necesitamos que no se distingan mayúsculas de minúsculas podemos usar:

```
$ diff archivo1.txt archivo2.txt -i
```

si necesitamos que no se distingan tabuladores de espacios podemos usar:

```
$ diff archivo1.txt archivo2.txt -E
```

si los archivos tienen líneas muy grandes, es posible usar el comando *fold* para cortar las líneas (por omisión a 80 caracteres), por ejemplo:

```
$ diff <(fold -s archivo1.txt) <(fold -s archivo2.txt)
```

También puede usarse con directorios. En este caso comparará los nombres de los archivos correspondientes en cada directorio por orden alfabético e imprimirá en pantalla los archivos que estén en un directorio pero no estén en el otro (otra opción es *sdiff* que permite mostrar las diferencias entre dos archivos y combinar sus contenidos de forma interactiva).

wc imprime en pantalla la cantidad de saltos de línea, palabras y Bytes totales que contenga un archivo. Para usarlo con un archivo cualquiera ejecutamos:

```
$ wc archivo.txt
```

podemos solo pedir que cuente el número de líneas usando *-l*, para el número de caracteres usamos *-c* o para el número de palabras usamos *-w*.

tail muestra en pantalla las últimas 10 líneas de un archivo:

```
$ tail archivo.txt
```

pero podemos indicarle un número diferente de líneas a visualizar usando el parámetro *-n*:

```
$ tail -n 30 archivo.txt
```

si el archivo solo tiene 43 líneas, podemos pedir que solo visualice la 42 y 43, mediante:

```
$ tail -n +42 archivo.txt
```

por último, si el archivo es generado por el sistema de mensajes de Linux(un archivo Log), podemos ver como se van generando sus entradas en tiempo real, usando:

```
$ tail -f /var/log/archivo.log
```

head es el comando opuesto a *tail*, muestra las primeras líneas de un archivo.

```
$ head archivo.txt
```

al igual que *tail*, muestra por defecto las 10 primeras líneas pero podemos indicarle un número diferente usando el parámetro *-n*:

```
$ head -n 50 archivo.txt
```

o que muestre todo excepto las últimas N líneas, usando:

```
$ head -n -15 archivo.txt
```

también es posible visualizar las primeras líneas de múltiples archivos usando:

```
$ head -n 2 archivo1.txt archivo2.txt
```

a la salida de este último comando le podemos quitar el nombre del archivo, mediante:

```
$ head –q n 2 archivo1.txt archivo2.txt
```

Podemos combinarlo con *tail* para mostrar sólo una determinada línea de un archivo (digamos la 13), usando:

```
$ head -n 13 archivo.txt | tail +13
```

y para mostrar un rango de líneas (digamos de la 10 a 15), usamos:

```
$ head -n 15 archivo.txt | tail -n +10
```

por último, podemos indicarle que solo nos muestre los primeros caracteres, mediante -C<número>, ejemplo:

```
$ head -C5 archivo.txt
```

more es un filtro que permite paginar el contenido de un archivo para que se vea a razón de una pantalla a la vez. Para utilizarlo simplemente ejecutamos:

```
$ more archivo.txt
```

para navegar a través del contenido del archivo usamos las flechas direccionales *Arriba* y *Abajo*, *Espacio* o la tecla *Enter* y para buscar una cadena usamos */*. Para salir de more usamos la tecla *q*. Además, podemos indicar que salte hasta donde se encuentra una determinada cadena usando *+/cadena*, o iniciar a partir de la línea *+n* del texto, por ejemplo:

```
$ more +/cadena archivo.txt
$ more +30 archivo.txt
```

less Aunque su nombre es lo opuesto de more es realmente una versión mejorada de éste último. Es otro filtro que permite paginar el contenido de un archivo pero que además de permitir la navegación hacia adelante y hacia atrás, esta optimizado para trabajar con archivos muy grandes. Ejecutarlo es tan simple como escribir:

```
$ less archivo.txt
```

permite navegar a través del contenido del archivo usando las flechas direccionales *Arriba* y *Abajo*, *Espacio* o la tecla *Enter*. Para salir de *less* también usamos la tecla *q*.

paste unifica (en salida estándar) dos o más archivos de texto, fusionando líneas de manera que las entradas en la primera columna pertenecen al primer archivo, las de la segunda columna son para el segundo archivo (separadas por tabuladores), y así sucesivamente, ejemplo:

```
$ paste uno.txt dos.txt tres.txt
```

se puede definir delimitadores entre las entradas de cada fila resultante, usando *-d <separador>*, ejemplo:

```
$ paste -d : uno.txt dos.txt tres.txt
```

además, se puede cambiar la operación de fusión, para que se realice por filas, de manera que el primer renglón son las entradas del primer archivo (separadas por tabuladores), el segundo renglón son las entradas del segundo archivo, y así sucesivamente, para ello se emplea la opción *-s*, ejemplo:

```
$ paste -s uno.txt dos.txt tres.txt
```

cut se encarga de cortar columnas o campos seleccionados de uno o más ficheros (o entrada estándar), por ejemplo para conocer los usuarios del sistema y su directorio de trabajo, usamos:

```
$ cut -d ":" -f 1,6 /etc/passwd
```

para indicar el delimitador usamos -d y para indicarle las columnas a visualizar usamos -f.

tr se encarga de sustituir un delimitador por otro de uno o más ficheros (o entrada estándar), por ejemplo:

Se sustituye el delimitador ":" por un tabulador:

```
$ cut -d ":" -f 1,6 /etc/passwd | tr -s ":" "\t"
```

o elimina las nuevas líneas (\n) de un archivo, usando:

```
$ tr -d \n < entrada.txt > salida.txt
```

entre otras opciones podemos pedir que cambie de mayúsculas a minúsculas o viceversa, usando:

```
$ echo "Esto es una prueba" | tr '[:lower:]' '[:upper:]'
$ echo "Esto es una prueba" | tr '[:upper:]' '[:lower:]'
```

también podemos transformar un conjunto de caracteres (aeiout) en otro ( ), ejemplo:

```
$ echo "Esto es una prueba" | tr '[aeiout]' ' '
```

o hacer el opuesto, es decir, transformar cualquier carácter que no este en conjunto (aeiout) en otro ( ), ejemplo:

```
$ echo "Esto es una prueba" | tr -c '[aeiout]' ' '
```

podemos borrar los caracteres indicados (minúsculas), ejemplo:

```
$ echo "Esto Es Una Prueba" | tr -d '[:lower:]'
```

o cambiar las vocales de mayúsculas a minúsculas, ejemplo:

```
$ echo "Esto Es Una Prueba" | tr 'aeiou' 'AEIOU'
```

es posible el reemplazo de rango de caracteres (a-e) o números(1-4) por otro (x), ejemplos:

```
$ echo "Esto es una prueba" | tr 'a-e' 'x'
$ echo "5uch l337 5p34k" | tr '1-4' 'x'
```

y podemos indicar que reemplace una o múltiples ocurrencias (espacios) por una sola, ejemplo:

```
$ echo "Esto       es          una       prueba" | tr -s ' '
```

fold se usa para que todas las líneas de un archivo se dividan a un ancho especificado, ejemplo:

```
$ cat /etc/services | fold -w 20
```

se puede usar -s para solicitar que si puede corte sobre un espacio, ejemplo:

```
$ cat /etc/services | fold -sw 20
```

nano Es un pequeño editor de texto que esta disponible en casi todas las distribuciones actuales de GNU/Linux[58], funciona con un menú en la parte de inferior que se activa con la tecla *Ctrl* (por ejemplo para grabar se usa *Ctrl-o* y para salir *Ctrl-x*).

```
$ nano archivo.txt
```

Comandos Generales

man muestra la documentación completa de todos los comandos. Por ejemplo, para ver la documentación del comando *clear*:

```
$ man clear
```

help proporciona ayuda de los comandos, con frecuencia puede sustituir al comando *man*, por ejemplo, para conocer la lista de comandos que soporta:

```
$ help
```

info proporciona ayuda de los comandos al igual que *man* y *help*, su uso es similar:

```
$ info mkdir
```

[58] Existe una versión moderna de este editor que no viene instalada por omisión, para instalarla usamos:

```
# apt install micro
```

y su uso es similar:

```
$ micro archivo.txt
```

whatis proporciona una ayuda breve de lo que hacen los comandos, sin mostrar sus opciones, ejemplo:

```
$ whatis ls
```

clear es un sencillo comando que limpiará nuestra terminal por completo dejándola como recién abierta. Para ello escribimos:

```
$ clear
```

w nos proporciona la lista de los usuarios activos en la computadora -recordemos que Linux es un sistema multiusuario-, su uso es:

```
$ w
```

time proporciona el tiempo de ejecución, que es dividido en real, usuario y del sistema, muestra de su uso es la siguiente:

```
$ time ls
```

whoami (del inglés Who Am I o Quien Soy Yo en español) muestra el identificador del usuario actual, para ejecutarlo solo basta con invocarlo:

```
$ whoami
```

date nos muestra la fecha y hora que tiene actualmente la computadora, ejemplo:

```
$ date
```

es posible formatear la salida del comando mediante:

```
$ date + "Weekday: %A Month: %B"[59]
$ date + "Week: %V Year: %y"
$ date -d last-week[60]
```

[59] Para conocer las diferentes banderas de date, usar:

```
$ date - -help
```

[60] podemos usar: tomorrow, last-year, next-year, next-month, entre otras.

cal muestra el calendario[61] del mes actual, con -y nos muestra el calendario del año completo, con -jy nos muestra el calendario con el número de día del año y con -A *n* y -B *m* nos muestra el calendario de *m* meses antes de la fecha actual y *n* meses después, ejemplo:

```
$ cal -A 3 -B 2
```

uptime muestra el tiempo que el equipo de cómputo ha pasado encendido sin ser reiniciado, así como el Load Average (carga promedio del sistema) que es el número de trabajos que se han realizado en los últimos 1, 5 y 15 minutos. Para ver su salida, solo escribimos en la terminal:

```
$ uptime
```

uname es un programa de sistemas operativos de tipo Unix que imprime detalles de la máquina y del sistema operativo que se esta ejecutando. Su salida es diferente dependiendo de las opciones, por ejemplo, uname solo muestra el nombre del sistema operativo pero cuando le pasamos la opción *-r* muestra la versión del Kernel y con *-a* de *all*, su salida es mucho mas completa. Se ejecuta de la siguiente forma:

```
$ uname -a
```

df nos muestra información de los discos y particiones en ellos, además de cuanto esta usado y libre en Bytes en cada una de las particiones, para ver la salida usando información en unidades Gb, Mb y Kb usamos la opción -h, para conocer el numero de inodos disponibles usamos la opción -i, podemos también conocer la información de todos los sistemas de ficheros usando -a y para conocer la información de los sistemas de archivos usamos -T, ejemplo:

```
$ df -hT
```

[61] Otra opción es *ncal*, que se puede instalar usando:

```
# apt install ncal
```

y se usa de forma similar a *cal*.

du nos muestra en Bytes cuanto ocupan los directorios de nuestra trayectoria actual de archivos, usamos la opción *-h* para que muestra el tamaño en unidades fáciles de leer -como KB, MB o GB-, *-a* para conocer el tamaño de archivos y directorios y *-s* para conocer el total de la trayectoria, ejemplo:

```
$ du -sh
```

free nos muestra la cantidad de memoria y Swap usada y libre del sistema, ejemplo:

```
$ free
```

podemos usar los modificadores para ver en Bytes *-b*, ver en Kilobytes *-k*, ver en Megabytes *-m*, ver en Gigabytes *-g*, desplegar una línea con los totales *-t*, ver en intervalos regulares *-s tiempo*, que nos muestre las estadísticas de bajo y alto uso *-l*.

bc es un lenguaje que soporta números de precisión arbitraria con ejecución interactiva, ejemplo:

```
$ bc -l
```

al escribir, por ejemplo:

```
scale = 100
1/3
quit
```

mostrará el resultado con 100 dígitos de precisión.

history muestra el historial de comandos ejecutados en la terminal, ejemplo:

```
$ history
```

para borrar dicho historial se usa la opción *-c*, ejemplo:

```
$ history -c
```

echo sirve para mostar texto en la pantalla, su uso es el siguiente

```
$ echo "Esto es un texto"
```

permite con la opción -e la interpretación de caracteres de escape como retroceso \b, nueva línea \n, tabulador \t, tabulador vertical \v, retorno de carro \r y diagonal invertida \\, ejemplo:

```
$ echo "Esto \ves un \ttexto"
```

Permisos

chmod (del inglés change mode) es un comando que permite cambiar los permisos de acceso de un directorio o archivo. Su sintaxis es:

```
$ chmod [opciones] <modo> <archivo>
```

donde *opciones* nos permite entre otras cosas, cambiar los permisos recursivamente para un directorio con *-R*, el *modo* son los permisos de lectura, escritura y ejecución representados en notación octal[62] y *archivo* es el nombre del directorio o archivo que queremos modificar[63].

Por ejemplo, para asignar permisos de lectura, escritura y ejecución para el dueño, el grupo y remover los permisos para el resto de los usuarios al archivo *prueba.txt* sería:

```
$ chmod 770 prueba.txt
```

[62]

Octal	Binario	Modo Archivo
0	000	- - -
1	001	- - x
2	010	- w -
3	011	- w x
4	100	r - -
5	101	r w -
6	110	r w -
7	111	r w x

[63] También están disponibles los comandos *getfacl* y *setfacl* pertenecientes a *Access Control List* (ACLs) que es un método más flexible para establecer permisos a usuarios y grupos sobre archivos y directorios.

chown (del inglés change owner) nos permite cambiar el propietario de un archivo o directorio. Su sintaxis es:

$ chown [opciones] <nuevo-propietario> <archivo>

donde *opciones* son las opciones del comando, como *-R* para cambiar recursivamente el propietario de un directorio y todo su contenido, *nuevo-propietario* será el nuevo propietario y archivo es el nombre del directorio o archivo que queremos modificar.

Por ejemplo, para cambiarle el propietario del directorio */home/ejercicios* y todo su contenido y asignarlo al usuario *pedro*, hacemos:

$ chown -R pedro /home/ejercicios

También es posible hacer el cambio de propietario y grupo en un solo comando, usando:

$ chown -R pedro:pedro /home/ejercicios

chgrp (del inglés change group) nos permite cambiar el grupo de un archivo o directorio. Su sintaxis es:

$ chgrp [opciones] <nuevo-grupo> <archivo>

donde *opciones* son las opciones del comando, como *-R* para cambiar recursivamente el grupo de un directorio y todo su contenido, *nuevo-grupo* será el nuevo grupo y *archivo* es el nombre del directorio o archivo que queremos modificar.

Por ejemplo, para cambiarle el grupo del directorio */home/ejercicios* y todo su contenido y asignarlo al usuario *pedro*, hacemos:

$ chgrp -R pedro /home/ejercicios

su permite cambiar las credenciales del usuario, es decir ser otro usuario, por ejemplo:

$ su antonio

el usuario del que comúnmente se desea adquirir sus credenciales es el de *root*, para ello usamos:

$ su

useradd (de agregar usuario) se utiliza para crear nuevos usuarios en tu sistema, su sintaxis es:

```
$ useradd [opciones] <nombre-usuario>
```

donde *opciones* nos permite asignar un grupo al usuario con *-g*, asignar el directorio */home* con *-d*, crearlo con *-m* si no existía previamente y *-s* para asignarle un intérprete de comandos o Shell, entre otras.

Así, para crear el usuario andrea cuyo grupo principal será *editores*, ejecutamos:

```
$ useradd -g editores -d /home/andrea -m -s /bin/bash andrea
```

usermod (de modificar usuario) modifica algunos parámetros de un usuario existente, como el nombre, su directorio /home y los grupos a los que pertenece, entre otros. Su sintaxis es:

```
$ usermod [opciones] <nombre-usuario>
```

donde *opciones* cambia el directorio home con *-d*, mueve todo el contenido del directorio anterior con *-m* y cambia el nombre de usuario con *-l*, entre otras. Para cambiar el nombre al usuario *andrea* por *violeta*, sería:

```
$ usermod -l violeta andrea
```

deluser (del inglés delete user) es un sencillo comando para borrar usuarios. Tiene la opción *-r* que adicionalmente borra su directorio */home*. Para borrar el usuario *violeta* con su */home*, ejecutamos:

```
$ deluser -r violeta
```

passwd (del inglés password) es una utilidad que se usa para cambiar o generar la contraseña de un usuario existente. Al invocarlo, pedirá la contraseña actual (si existe) y luego que la contraseña nueva sea introducida dos veces para verificar que fue escrita correctamente. Por ejemplo para asignar una contraseña al usuario *antonio*, sería:

```
$ passwd antonio
```

para conocer la información del Password de mi cuenta puedo usar:

```
$ passwd -S antonio
```

lsattr permite listar los atributos[64] asignados a los ficheros y directorios, para ver los atributos del directorio actual usamos:

```
$ lsattr -a
```

o de forma recursiva

```
$ lsattr -Ra
```

chattr permite cambiar los atributos asignados a los ficheros y directorios, las opciones se agregan con + y se quitan con - y podemos hacerlo de forma recursiva *R*, por ejemplo podemos poner el permiso de inmutable, es decir, que no puede eliminar ni cambiar de nombre, mediante:

```
# chattr +i archivo
```

o quitar el permiso:

```
# chattr -i archivo
```

Podemos deshabilitar la modificación de la fecha de acceso al fichero (atime) mediante:

```
# chattr +A archivo
```

[64] Algunos valores posibles son: *(A)* indica que el valor de la fecha de acceso sobre un archivo no será cambiado en cada lectura, *(a)* el archivo solo puede ser abierto en edición para escritura por redireccionamiento (>>) y no puede ser eliminado, *(c)* el archivo tiene activa la compresión de datos, *(D)* indica que los datos escritos en un directorio se sincronizan en el disco de forma automática, *(d)* elimina el fichero o directorio de las copias de seguridad realizadas con la utilidad *dump*, *(e)* los archivos con este atributo están usando extensiones para mapear los bloques en el disco, (E) el sistema de archivos cifra el archivo, directorio o enlace simbólico que tenga este atributo, *(I)* indica que la carpeta está indexada, *(i)* indica que el fichero o directorio es de solo lectura y no puede ser modificado, borrado, renombrado o ligado simbólicamente ni siquiera por el usuario *root*, *(j)* indica que esta activo el "journaling" de los archivos, (S) indica que el archivo es síncrono, las escrituras en el archivo son inmediatamente efectuadas, *(s)* indica que cuando el archivo sea eliminado, el espacio que ocupaba será rellenado por bloques de ceros, *(T)* indica que el directorio con este atributo se escribirá en las partes más rápidas del disco, *(t)* indica que el fichero no presenta fragmentación en el sistema de ficheros, *(u)* indica que los archivos con este atributo pueden ser recuperados después de ser borrados, *(V)* los archivos con este atributo no se podrá escribir en el y el sistema de archivos verificará automáticamente los datos leídos.

esto permite que cuando se acceda al fichero no se modifique el registro de atime. De este modo no quedarán registrada la fecha de último acceso.

Es posible comprimir automáticamente el fichero en disco por el Kernel y cuando se lea el archivo se verá descomprimido, ejemplo:

```
# chattr +c archivo
```

Además se puede permitir la recuperación de un archivo aunque sea eliminado, usando:

```
# chattr +u archivo
```

y el caso opuesto, al eliminar un archivo, sobrescribir con ceros todos sus bloques, mediante:

```
# chattr +e archivo
```

Podemos establecer que el archivo solo se pueda abrir en modo de adición para escritura, mediante:

```
# chattr +a archivo
```

id es un programa que muestra el identificador del usuario (UID) que ejecuta el programa, el identificador de su grupo (GID), así como todos los grupos a los cuales pertenece el usuario, ejemplo:

```
$ id usuario
```

Búsqueda

find permite buscar[65] dentro del sistema de archivos un directorio o archivos que concuerden con el patrón dado, por ejemplo:

[65] Otra opción al comando *find* es el comando *fdfind*, el cual se puede instalar usando:

```
# apt install fd-find
```

y podemos usarlo mediante:

```
$ fdfind debian
```

```
$ find /home -name *.pdf
```

busca desde la trayectoria */home*, todos los archivos que concluyan con *.pdf* y nos muestra las trayectorias a los archivos que concuerdan con lo solicitado. También podemos pasar la salida de *find* a otro comando como por ejemplo:

```
$ find . -name Portada.pdf | xargs ls -l
```

que nos mostrará la salida larga del comando *ls* de todos los archivos que encuentre *find*.

Además, podemos hacer uso de expresiones regulares como:

```
$ find . -regex "[a-f0-9\-]\{36\}\.jpg"
```

grep permite buscar[66] en archivos un determinado patrón[67], mostrando la línea que lo contiene, por ejemplo:

busca en todos los archivos **.txt* la cadena *chmod:*

```
$ grep chmod *.txt
```

si necesitamos hacer la búsqueda sobre todos los archivos de la carpeta actual y todas sus subcarpetas, podemos usar:

```
$ grep -r chmod *.txt
```

si requerimos seguir sobre todos los enlaces simbólicos, usamos:

[66] El comando grep comparte funcionalidad con egrep y fgrep. ¿Cuál es la diferencia entre grep, egrep y fgrep?, la e en egrep significa extendido (grep -E) y la f en fgrep significa fijo (grep -F). egrep permite el uso de expresiones regulares extendidas y fgrep no permite la expresión regular en absoluto.

[67] Grep es una herramienta muy poderosa, pero existe otra: ripgrep, que es eficiente en búsquedas recursivas, combinando con opciones avanzadas como el filtrado, uso de expresiones regulares, colores, soporte Unicode. Se instala usando:

```
# apt install ripgrep
```

y lo usamos, por ejemplo buscando foobar en los archivos *.py:

```
$ rg -tpy foobar
```

```
$ grep -R chmod *.txt
```

para que nos indice el nombre de archivo en los cuales se encontró la cadena *main*:

```
$ grep -l main *.java
```

busca en todos los archivos **.log* la cadena *error*, ignorando si esta en mayúsculas o minúsculas:

```
$ grep -i error *.log
```

busca en todos los archivos **.log* aquellas que no tengan la cadena *error*:

```
$ grep -v error *.log
```

busca en todos los archivos **.log* aquellos que tengan a la cadena *error* o *fatal:*

```
$ grep 'error\|fatal' *.log
```

busca en todos los archivos **.log* aquellos que tengan a la cadena *error* o *fatal:*

```
$ grep -E 'error\|fatal' *.log
```

busca en todos los archivos **.log* aquellos que tengan a la cadena *error* o *fatal* e indica cuantas coincidencias encontró*:*

```
$ grep -c 'error\|fatal' *.log
```

busca en todos los archivos **.log* aquellos que tengan a la cadena *error* o *fatal*, marcando en color las cadenas encontradas:

```
$ grep –color -E "error|fatal" *.log
```

regresa las líneas que no tengan la cadena indicada:

```
$ grep -v // Ejemplo.java
```

regresa las líneas que no contienen la cadena indicada y no son vacías:

```
$ grep -Ev "^//|^$" Ejmplo.java
```

Podemos usar ^ y $ para forzar a que se encuentre solo al inicio o al final de la línea respectivamente, por ejemplo:

```
$ grep ^antonio /etc/passwd
$ grep sh$ /etc/passwd
```

o buscar una línea que solo contenga una palabra, por ejemplo:

```
$ grep '^texto$' archivo
```

o líneas en blanco usando:

```
$ grep '^$' archivo
```

También podemos indicar que busque usando rango de caracteres mediante[68]:

```
$ grep '[Aa]ntonio' archivo
$ grep '[Aa][Nn]tonio[0-9]' archivo
$ grep '[A-Z][a-z]' archivo
$ grep [[:upper:]] archivo
```

locate permite buscar archivos o directorios cuyo nombre coincida con el patrón dado[69], por ejemplo:

[68] Podemos usar
[[:alnum:]] - Caracteres alfanuméricos
[[:alpha:]] - Caracteres alfabéticos
[[:blank:]] - Espacio o tabulador
[[:digit:]] - Dígitos
[[:lower:]] - Letras minúsculas
[[:space:]] - Espacio, tabulador
[[:upper:]] - Letras mayúsculas

[69] En caso de no estar instalado en el sistema, lo instalamos usando:

```
# apt install mlocate
```

y para actualizar la base de datos del comando, usamos:

```
# updatedb
```

en caso de haber borrado archivos, es necesario actualizar la base de datos para excluirlos.

```
$ locate *dir2*
```

si necesitamos que la búsqueda sea insensible a mayúsculas y minúsculas usamos:

```
$ locate -i desktop.iso
```

podemos limitar la búsqueda a un determinado número de ocurrencias, por ejemplo:

```
$ locate "*.html" -n 20
```

o solo visualizar el número de ocurrencias encontradas, ejemplo:

```
$ locate -c "*.html"
```

Para conocer las estadísticas del comando *locate*, usamos:

Respaldo

tar permite respaldar en un solo archivo un grupo de archivos y/o directorios sin compactarlos, para ello usar:

```
$ tar -cvf nombre.tar directorio
```

para restaurar usar:

```
$ tar -xvf nombre.tar
```

gzip permite respaldar en un solo archivo un grupo de archivos y/o directorios compactándolos usando *gzip*, para ello usamos:

```
$ tar -cvf nombre.tar directorio
$ gzip –best nombre.tar
```

o en un solo paso, usamos:

```
$ tar -zcvf nombre.tar.gz directorio
```

para restaurar se usar:

```
$ tar -zxvf nombre.tar.gz
```

o alternativamente podemos usar:

```
$ gunzip nombre.tar.gz
$ tar -xvf nombre.tar
```

con *gzip* podemos sólo comprimir o descomprimir respectivamente usando:

```
$ gzip fichero
$ gzip -d fichero.gz
```

bzip2 permite respaldar en un solo archivo un grupo de archivos y/o directorios compactándolos usando *bzip2*, para ello escribimos:

```
$ bzip fichero
$ bzip -d fichero.bz2
```

zip permite respaldar en un solo archivo un grupo de archivos y/o directorios compactándolos, para ello usamos:

```
$ zip archivo.zip fichero(s)
$ zip -r archivo.zip directorio/
```

también permite establecer el nivel de compresión usando una escala de 0 a 9 (por omisión es 6), por ejemplo:

```
$ zip -8 archivo.zip fichero(s)
$ zip -8 -r archivo.zip directorio/
```

y podemos borrar los archivos que se comprimirán al terminar el proceso, usando:

```
$ zip -m archivo.zip fichero(s)
$ zip -r -m archivo.zip directorio/
```

Podemos agregar un archivo a un *.zip* existente, usando:

```
$ zip -u archivo.zip nuevo.txt
```

o eliminar un archivo a un .zip existente, usando:

```
$ zip -d archivo.zip porBorrar.txt
```

Podemos crear un *.zip* protegido con clave, mediante:

```
$ zip -e archivo.zip fichero(s)
```

o proteger con clave un *.zip* ya existente, usando:

```
$ zipcloak archivo.zip
```

Podemos ver la información detallada del archivo comprimido, mediante:

```
$ zipdetails archivo.zip
```

y descomprimir mediante:

```
$ unzip archivo.zip
```

también podemos indicar el directorio en en cual descomprimir, usando:

```
$ unzip archivo.zip -d /home/usuario/temp/
```

si al descompactar existe algún error, es posible recuperar parte de los archivos mediante:

```
$ zip -F archive.zip -O archive-fixed.zip
```

o usar *-FF*, después usar:

```
$ jar xvf archive-fixed.zip
```

otra alternativa es usar:

```
$ bsdtar xf archivo.zip
```

Si se requiere descomprimir múltiples archivos empaquetados podemos usar:

```
$ for z in *.zip; do unzip "$z"; done
```

unp permite descomprimir de casi cualquier formato de respaldo, su uso es de lo más sencillo, a saber:

```
$ unp archivo.compactado
```

Varios

file determina el tipo de un archivo y te imprime en pantalla el resultado. No hace falta que el archivo tenga una extensión para que *file* determine su tipo, pues la aplicación ejecuta una serie de pruebas sobre el mismo para tratar de clasificarlo.

```
$ file un_archivo_de_texto.txt
```

stat nos da información de un archivo, datos como: tamaño, blocks usados para almacenarlo, número de ligas, datos del dueño y grupo; fechas de acceso, modificación y cambio.

```
$ stat archivo
```

type permite identificar el comando pasado como parámetro indicando la trayectoria si es comando externo o si es comando interno al Shell, ejemplo:

```
$ type ls
```

ps nos muestra los procesos activos del sistema junto con información de la ejecución de los mismos, para ver todos los procesos en el sistema usar -ef, para conocer los procesos de un usuario usamos -U <usuario>, para conocer los procesos de un determinado grupo usamos -G <grupo>, para conocer todos los procesos de un determinado programa usamos -C <programa>, para ver todos los procesos en forma de árbol y saber que proceso depende de que otros, usamos -ejH, ejemplo:

```
$ ps -ejH
```

Los procesos pueden ser: uninterruptible sleep (D), idle (I), running (R), sleeping (S), stopped by job control signal (T), stopped by debugger during trace (t), zombie (Z), los podemos ver usando:

```
$ ps -au
```

Podemos pedirle al comando que nos muestre algunos datos informativos de los comandos en ejecución y mandar a un archivo los procesos que más consumen memoria, ejemplo:

```
ps -eo cmd,pid,ppid,%mem,%cpu –sort=-%mem | head | tee
topprocs.txt
```

si queremos adicionar datos al archivo previamente creado usamos:

```
$ ps -eo cmd,pid,ppid,%mem,%cpu –sort=-%mem | head | tee
-a topprocs.txt
```

kill es un comando utilizado para enviar mensajes sencillos a los procesos ejecutándose en segundo plano en el sistema. Por defecto el mensaje que se envía es la señal de terminación. Su sintaxis más sencilla es:

```
$ kill [-s] <pid>
```

donde *-s* es la señal a enviar, de no ser especificada ninguna se manda la señal por defecto (*SIGTRM*) y *pid* es el identificador del proceso. Otra de sus opciones es *-9* (*SIGKILL*) que fuerza la terminación de un proceso, para conocer los posibles mensajes de *kill* usar:
$ kill -l.
En Linux cada comando, proceso o ventana gráfica tiene un número de proceso *(PID)*, este se puede obtener mediante el comando *ps* o *top*, y el comando *kill* puede concluir con la ejecución del *PID* indicado y todos sus subprocesos -el usuario sólo puede matar sus propios procesos, *root* puede finanizar (matar) los procesos de cualquier usuario-, ejemplo:
Por ejemplo, para terminar un proceso cuyo *PID* es *3477*, ejecutamos:

```
$ kill 3477
```

Otra señal importante es 1 (SIGHUP) que permite parar y reinicializar el proceso indicado, por ejemplo para detener el proceso 2434, usamos:

```
$ kill -1 2434
```

y lo reiniciamos usando:

```
$ kill -1 2434
```

killall permite finalizar (matar) todas nuestras instancias de ejecución de un comando, por ejemplo:

```
$ killall firefox-esr
```

o podemos finalizar la sesión de un usuario, usando:

```
# killall -u antonio
```

pgrep permite conocer los identificadores de proceso de una determinada aplicación corriendo en el sistema, por ejemplo:

```
$ pgrep firefox
```

pwdx permite conocer el directorio de trabajo de una aplicación a través de su identificador de proceso pasado como parámetro a *pwdx*, ejemplo:

```
$ pwdx 4534
```

o

```
$ pwdx $(pgrep firefox)
```

awk permite procesar, analizar archivos de texto que estén organizados por filas y columnas, ejemplo:

$ awk -F':' '{ print $1 }' /etc/passwd

nos mostrarán todos los usuarios que tiene el sistema, los cuales están dados de alta en el archivo del sistema */etc/passwd.*

Si necesitamos visualizar una determinada línea de un archivo (digamos la 5), podemos usar:

```
$ awk 'NR==5' archivo.txt
```

en el caso de necesitar visualizar un rango de líneas de un archivo (digamos de la 20 a 25), usamos:

```
$ awk 'NR>=20 && NR<=25' archivo.txt
```

sort imprime en pantalla las líneas de un archivo ordenadas alfabéticamente. Para ejecutarlo basta con:

```
$ sort archivo.txt
```

podemos solicitar que lo haga en orden inverso usando -r, que haga el ordenamiento numérico usando -n, podemos omitir los duplicados usando -u, ordenar ignorando mayúscula y minúsculas con -f, ordenamiento tomando en cuenta valores alfanuméricos -h, ordenamiento aleatorio con -R, que ordene por meses -M, que haga el ordenamiento de los renglones tomando como índice cierto renglón, por ejemplo:
$ ls -al | sort -k 4 -n

sed es considerado un editor de texto orientado a "flujo" -en contraposición a los clásicos editores «interactivos»- el cual acepta como entrada un archivo o entrada estándar; cada línea es procesada y el resultado es enviado a la salida estándar. Por ejemplo, borrar las líneas tres a cinco de archivo archivo.txt:

```
$ sed '3,5d' archivo.txt
```

otro ejemplo, borrar todas las líneas en blanco (no las que sólo tengan tabuladores y/o espacios) del archivo fichero.txt

```
$ sed '/^$/d' archivo.txt
```

para quitar las líneas en blanco y las que sólo tengan tabuladores y/o espacios, usamos:

```
$ sed sed '/^[ \t]*$/d' archivo.txt
```

si deseamos visualizar una determinada línea de un archivo (digamos la 13), usamos:

```
$ sed -n '13p' archivo.txt
```

si queremos visualizar un rango de líneas de un archivo (digamos de la 20 a 25), usamos:

```
$ sed -n '20,25p' archivo.txt
```

si necesitamos cambiar las vocales de un archivo de minúsculas a mayúsculas, podemos usar:

```
$ sed 'y/aeiou/AEIOU/' archivo.txt
$ sed 's/[aeiou]/\U&/g' archivo.txt
```

md5sum genera la suma de verificación md5 (Compute and Check MD5 Message Digest) o Hash usada para verificar la integridad de los archivos, esta puede haber cambiado como resultado de una transferencia de archivos defectuosa, un error en disco o una interferencia maliciosa, ejemplo[70]:

```
$ md5sum debian.testing-amd64-netinst.iso
```

sleep se utiliza para temporalizar un intervalo de tiempo determinado, la unidad por defecto es el segundo (s), pero se puede usar minutos (m), horas (h) o días (d), por ejemplo:

```
$ sleep 3m && ls -al
```

watch permite correr un comando de forma repetitiva y a intervalos regulares (2 segundos por omisión) mostrando su salida, por ejemplo:

```
$ watch free
```

de ser necesario, podemos quitarle el título, usando:

```
$ watch -t free
```

podemos indicar el intervalo de ejecución usando:

```
$ watch -n 5 free
```

Podemos solicitarle que nos muestre los cambios sobre la última salida, usando:

```
$ watch -n 5 -d free
```

y podemos indicarle que concluya la ejecución si la salida actual es distinta a la anterior, usando:

```
$ watch -n 5 -g free
```

[70] Otras opciones son:

```
$ shasum debian.testing-amd64-netinst.iso
$ shasum -a256 debian.testing-amd64-netinst.iso
```

Monitorear el Desempeño Existen múltiples herramientas para ser usadas en línea de comandos y ambiente gráfico que permiten monitorear el desempeño y uso de una computadora con GNU/Linux, estas se pueden usar para administrar el sistema y las comunicaciones por red, estos comandos[71] están disponibles en todas las distribuciones de GNU/Linux y son normalmente usados para determinar problemas de desempeño en nuestro sistema de cómputo.

lscpu para conocer el tipo de CPU y sus características usamos:

```
$ lscpu
```

podemos usar también *cat /proc/cpuinfo*, si deseamos un análisis más detallado están los comandos:

```
lscpi, cpuid, dmidecode, inxi, hardinfo, lshw, hwinfo, nproc
```

free despliega la memoria total, usada, compartida, en Cache y libre del sistema:

```
$ free
```

podemos usar también cat /proc/meminfo, si deseamos un análisis más detallado están los comandos:

```
top, vmstat, dstat
```

top muestra el desempeño de nuestro equipo actualizando cada segundo el uso del CPU, memoria, Swap, Cache, Buffer y los procesos que están corriendo en el sistema actualmente y en cada proceso que corre se muestra el identificador, el porcentaje de CPU, prioridad y memoria usada, etc. Para usarlo usamos:

```
$ top
```

[71] Algunos comandos son utilizados por cualquier usuario y otros solo por el administrador.

para salir del programa se debe presionar la tecla q. Los procesos pueden ser: uninterruptible sleep (D), idle (I), running (R), sleeping (S), stopped by job control signal (T), stopped by debugger during trace (t), zombie (Z).

Otras variantes de este comando son:

> bashtop, htop, glances, conky, nmon, atop, vtop, gtop, ps, mpstat, collectl, sar, pstree, pmap, pgrep, pkill, kill, killall, xkill, Linux Process Viewer, etc.

En circunstancias particulares, podemos necesitar que un proceso tenga una distinta prioridad de ejecución (menor o mayor del valor por omisión), para ello podemos usar los comando *nice* y *renice* para cambiar dicho valor. Además podemos pedir que un proceso determinado se restrinja a un procesador particular mediante el uso del comando *taskset* que por lo general mejora el rendimiento.

whowatch es una utilidad simple que muestra de forma interactiva y en tiempo real los usuarios y procesos activos en el sistema:

```
$ whowatch
```

Otras opciones son:

> bashtop, htop, glances, conky, nmon, atop, gtop, ps, mpstat, collectl, sar, pstree, pmap, pgrep, pkill, kill, killall, xkill, Linux Process Viewer, etc.

dstat muestra las estadísticas de recursos de todo el sistema de forma versátil en tiempo real:

```
$ dstat -c –top-cpu -dn –top-mem –mem
```

combina la capacidad de comandos como *iostat, vmstat, netstat* e *ifstat*. Otras opciones son:

> nload, collectl, iptraf, nethogs, iftop, mtr, bmon, slurn, tcptrack, monitorix, nmon, glances

vmstat muestra las estadísticas de la memoria virtual, hilos del Kernel, uso de discos, procesos del sistema, entradas y salidas de bloque, interruptores y actividad del CPU, entre otras opciones, este comando esta contenido en el paquete systat:

$ vmstat 1

otras opciones son:

dstat, sar, vnstat, vnstati, mpstat, iostat, iotop, ioping, atop, top, collectl, nmon, glances

netstat permite monitorizar los paquetes de red que entran y salen, genera estadísticas de su uso, es un paquete que permite encontrar problemas de desempeño en las comunicaciones de red:

$ netstat

podemos solicitar que sólo nos muestre lo referente a un solo puerto, usando:

netstat -ltnp | grep -w ':80'

otras opciones son:

dstat, collectl, iptraf, nethogs, iftop, ifstat, mtr, monitorix, nmon, bwm-ng, cbm, speedometer, pkstat, netwatch, trafshow, netload, glances

iotop permite conocer qué procesos están generando actividades de lectura y grabación en los discos del sistema, así es posible conocer qué procesos están sobrecargando el sistema:

$ iotop

otras opciones son:

iostat, ioping, vmstat, atop, htop, dstat, glances, netdata, netstat, nmon, collectl, glances

iostat este permite conocer estadísticas de uso del sistema de entrada/salida incluyendo dispositivos, discos locales, discos remotos tales como *NFS* y *SAMBA*:

```
$ iostat
```

Otras opciones son:

iotop, ioping, iostat, atop, dstat, nfsstat, ifstat, atop, nmon, collectl, glances

lsof permite conocer la lista de archivos abiertos además de Sockets Network, Pipes, dispositivos y procesos:

```
$ lsof
```

Lista todos los procesos que tiene abierto el archivo:

```
$ lsof /trayectoria/archivo
```

Lista todos los archivos abiertos por el usuario:

```
$ lsof -u usuario
```

también se pueden indicar múltiples usuarios:

```
$ lsof -u usuario1, usuario2
```

o bién por todos los usuarios menos uno, por ejemplo root:

```
$ lsof -u ^root
```

Lista todos los archivos abiertos en un directorio:

```
$ lsof +D /trayectoria/directorio/
```

Lista todos los archivos abiertos por un identificador de proceso:

```
$ lsof -p <pid>
```

también podemos especificar múltiples identificadores de proceso:

```
$ lsof -p pid1, pid2, pid3
```

Lista todos los archivos abiertos por un comando:

```
$ lsof -c <comando>
```

Busca archivos abiertos por un usuario, comando o proceso:

```
$ lsof -a -u usuario -c comando
```

Lista las conexiones y puertos de red abiertos:

```
$ lsof -i
```

si usamos IPV4 o IPV6 podemos ver esos puertos abiertos:

```
$ lsof -i 4
$ lsof -i 6
```

podemos pedirle que nos muestre puertos tcp o udp:

```
$ lsof -i tcp
```

podemos conocer los procesos que usan el puerto TCP:80, mediante:

```
$ lsof -i TCP:80
```

o para los puertos TCP:1-1024, mediante:

```
$ lsof -i TCP:1-1024
```

Podemos pedirle que nos muestre la actividad de un usuario y que archivos están involucrados, usando:

```
$ lsof -i -u usuario
```

Uno de sus principales usos es conocer qué proceso tiene acceso a un disco o partición que no se puede desmontar y manda un error de que un archivo esta siendo usado, pare ello usamos:

```
$ lsof /dev/sda2
```

También podemos matar toda la actividad de un usuario particular:

```
# kill -9 'lsof -t -u antonio'
```

lsusb lista los dispositivos USB del sistema además información del fabricante del mismo, ejemplo:

```
$ lsusb
```

tcpdump es uno de los comando más usados para analizar paquetes de red y es usado para capturar o filtrar paquetes *TCP/IP* que se reciben o se transfieren en una interfaz de red especifica:
tcpdump -i eth0
también permite grabar los paquetes capturados para un análisis posterior. Otras opciones son:

arpwatch, suricata, wireshark, vnstat, vnstati, negios, collectl, glances, ss, iptraf, nethogs, iftop, mtr

ip muestra información y permite manipular los dispositivos de red (interfaces y tuneles), uso:

```
$ ip address
# ip link set ens3 up
# ip link set ens3 down
```

nmcli información sobre los dispositivos de red y su configuración, uso:

```
$ nmcli
```

3.5 GNU Core Utilities

Coreutils (o GNU Core Utilities) es un paquete de Software desarrollado por el proyecto GNU que contiene varias de las herramientas básicas como cat, ls y rm necesarias para sistemas operativos del tipo Unix. Es una combinación de tres paquetes: utilidades de ficheros (fileutils), utilidades de intérpretes de comandos (shellutils) y utilidades de proceso de textos (textutils).

Las utilidades GNU core soportan opciones de cadena larga como parámetros a los comandos, así como cierta permisividad en la convención al especificar opciones antes de los argumentos regulares (siempre que la variable de entorno POSIXLY_CORRECT esté definida, hecho que habilita una diferente funcionalidad en BSD). Adicionalmente, como la filosofía GNU emplea

información desde páginas de manual (y usa herramientas como info), la información proporcionada es mayor. La siguiente lista enumera algunas utilidades importantes:

Salida de archivos completos

- cat: Concatenar y escribir archivos
- tac: Concatenar y escribir archivos al revés
- nl: Rectas numéricas y archivos de escritura.
- od: escribir archivos en formato octal u otros formatos
- base32: Transformar datos en datos imprimibles
- base64: Transformar datos en datos imprimibles
- basenc: Transformar datos en datos imprimibles

Formatear el contenido del archivo

- fmt: reformatear el texto del párrafo
- pr: Paginar o dividir archivos en columnas para imprimir
- fold: Ajustar las líneas de entrada para que quepan en el ancho especificado

Salida de partes de archivos.

- head: Generar la primera parte de los archivos.
- tail: Generar la última parte de los archivos.
- split: Divide un archivo en pedazos.
- csplit: Dividir un archivo en partes determinadas por el contexto

Archivos resumidos

- wc: Imprimir recuentos de nuevas líneas, palabras y bytes
- sum: Imprimir suma de comprobación y recuentos de bloques

- cksum: Imprimir y verificar sumas de verificación de archivos
- md5sum: Imprima o consulte resúmenes de MD5
- b2sum: Imprima o consulte resúmenes de BLAKE2
- sha1sum: Imprima o consulte resúmenes SHA-1
- sha2: imprima o consulte resúmenes SHA-2

Operar con archivos ordenados

- sort: Ordenar archivos de texto
- shuf: Texto aleatorio
- uniq: Unificar archivos
- comm: Compara dos archivos ordenados línea por línea
- ptx: producir índices permutados
- tsort: clasificación topológica

Operando en el campo

- cut: Imprimir partes seleccionadas de líneas
- paste: Fusionar líneas de archivos
- join: Unir líneas en un campo común

Operando con personajes

- tr: Traducir, comprimir y/o eliminar caracteres
- expand: Convertir tabulaciones a espacios
- unexpand: Convertir espacios en pestañas

Listado de directorio

- ls: Listar el contenido del directorio

- dir: enumerar brevemente el contenido del directorio
- vdir: enumerar detalladamente el contenido del directorio
- dircolors: Configuración de color parals

Operaciones básicas

- cp: Copiar archivos y directorios
- dd: Convertir y copiar un archivo
- install: Copiar archivos y establecer atributos
- mv: Mover (cambiar nombre) archivos
- rm: Eliminar archivos o directorios
- shred: Elimina archivos de forma más segura

Tipos de archivos especiales

- link: Realizar un enlace físico a través de la llamada al sistema de enlace
- ln: Hacer enlaces entre archivos
- mkdir: Crear directorios
- mkfifo: Crear FIFO (canalizaciones con nombre)
- mknod: Crear archivos especiales de bloques o caracteres
- readlink: Imprimir valor de un enlace simbólico o nombre de archivo canónico
- rmdir: Eliminar directorios vacíos
- unlink: Eliminar archivos mediante la llamada al sistema de desvinculación

Cambiar los atributos del archivo

- chown: Cambiar propietario y grupo de archivos
- chgrp: Cambiar la propiedad del grupo
- chmod: Cambiar permisos de acceso
- touch: Cambiar marcas de tiempo de archivos

Uso del espacio de archivos

- df: Informar del uso de espacio del sistema de archivos
- du: Estimar el uso del espacio de archivos
- stat: Informe del estado del archivo o del sistema de archivos
- sync: Sincronizar escrituras en caché con almacenamiento persistente
- truncate: Reducir o ampliar el tamaño de un archivo

Imprimir texto

- echo: Imprimir una línea de texto
- printf: Formatear e imprimir datos
- yes: Imprime una cadena hasta que se interrumpa

Condiciones

- false: No hacer nada, sin éxito.
- true: No hacer nada, con éxito
- test: Verifique tipos de archivos y compare valores
- expr: Evaluar expresiones

Redirección

- tee: Redirigir la salida a múltiples archivos o procesos

Manipulación de nombres de archivos

- basename: Eliminar directorio y sufijo de un nombre de archivo
- dirname: Eliminar el componente del último nombre del archivo
- pathchk: Verificar la validez y portabilidad del nombre del archivo
- mktemp: Crear archivo o directorio temporal
- realpath: Imprime el nombre del archivo resuelto.

Contexto de trabajo

- pwd: Imprimir directorio de trabajo
- stty: Imprimir o cambiar características del terminal
- printenv: Imprimir todas o algunas variables de entorno
- tty: Imprimir el nombre del archivo del terminal en la entrada estándar

Información del usuario

- id: Imprimir identidad del usuario
- logname: Imprimir el nombre de inicio de sesión actual
- whoami: Imprimir nombre de usuario efectivo
- groups: Imprimir nombres de grupos en los que se encuentra un usuario
- users: Imprimir nombres de inicio de sesión de los usuarios actualmente conectados
- who: Imprimir quién está conectado actualmente
- pinky: Imprimir información sobre los usuarios

Contexto del sistema

- date: Imprimir o configurar la fecha y hora del sistema
- arch: Imprimir el nombre del hardware de la máquina
- nproc: Imprime el número de procesadores disponibles.

- uname: Imprimir información del sistema
- hostname: Imprimir o configurar el nombre del sistema
- hostid: Imprimir identificador de host numérico
- uptime: Tiempo de actividad y carga del sistema de impresión

Contexto SELinux

- chcon: Cambiar el contexto SELinux del archivo
- runcon: Ejecutar un comando en el contexto SELinux especificado

Invocación de comando modificada

- chroot: Ejecute un comando con un directorio raíz diferente
- env: Ejecutar un comando en un entorno modificado

Expansión de variables ambientales

- nice: Ejecutar un comando con amabilidad modificada
- nohup: Ejecute un comando inmune a los cuelgues
- stdbuf: Ejecute un comando con almacenamiento en búfer de flujo de E/S modificado
- timeout: Ejecutar un comando con un límite de tiempo

Control de procesos

- kill: Enviar una señal a los procesos.

Retraso

- sleep: Retraso por un tiempo específico

Operaciones numéricas

- factor: Imprimir factores primos
- numfmt: Reformatear números
- seq: Imprimir secuencias numéricas

3.6 Desde la Nube

Existen diferentes servicios Web[72] que permiten instalar, configurar y usar cientos de sistemas operativos Linux y Unix -máquinas virtuales usando servicios Web en Debian GNU/Linux y QEMU (véase sección 11)- desde el navegador, esto en aras de que los usuarios que cuenten con algún sistema de acceso a red y un navegador puedan usar, configurar e instalar algún sistema operativo y su respectiva paquetería sin hacer instalación alguna en su equipo de cómputo, tableta o teléfono celular[73].

Una muestra de estos proyectos son: Distrotest (https://distrotest.net), JSLinux (https://bellard.org/jslinux) y OnWorks (https://www.onworks.net).

Algunas versiones listas para usar son:

4mLinux, AbsoluteLinux, Academix, AlpineLinux, Antergos, antiX Linux, Aptosid, ArchBang, ArchLabs, ArchLinux, Archman, ArchStrike, ArcoLinux, ArtixLinux, AryaLinux, AV Linux, BackBoxLinux, BigLinux, Bio-Linux, BlackArch, BlackLab, BlackPantherOS, BlackSlash, blag, BlankOn, Bluestar, Bodhi, BunsenLabs, ByzantineOS, Caine, Calculate Linux Desktop, CentOS, Chakra, ChaletOS, ClearOS, Clonezilla, ConnochaetOS, Cucumber, Damn Small Linux, Damn Small Linux Not, Debian, DebianEdu, deepin, DEFT, Devil-Linux, Devuan, DragonFly BSD, Dragora, DuZeru, Dyne:bolic, Edubuntu, elementaryOS, Elive Linux, Emmabuntüs, Emmi OS, Endless OS, EnsoOS, Exe GNU/Linux, ExTiX, Fatdog64, Fedora Atomic, Fedora Server, Fedora Workstation, FerenOS, FreeBSD, FreeDOS, Frugalware, G4L, GeckoLinux, Gentoo, GNewSense, GoboLinux, Gparted, GreenieLinux, GRML, GuixSD, Haiku, Heads, Kali Linux, Kanotix, KaOS, Knoppix, Kodachi, KolibriOS, Korora, Kubuntu, Kwort, Linux Lite, Linux Mint, LiveRaizo, LMDE, Lubuntu, LXLE OS, Macpup, Mageia, MakuluLinux, Manjaro, Matriux, MauiLinux, MenuetOS, MinerOS, MiniNo, Modicia, Musix, MX Linux, Nas4Free, Neptune, NetBSD, Netrunner, NixOs, NST, NuTyX, OpenIndiana, OpenMandriva, openSUSE, OracleLinux, OSGeo live, OviOS, Parabola CLI, Parabola LXDE, Pardus, Parrot Home, Parrot Security, Parrot Studio, Parsix, PCLinuxOS, PeachOSI, Pentoo, Peppermint, PeppermintOS, Pinguy, PinguyOS, plopLinux, PointLinux, Pop!_OS, PORTEUS, Puppy Linux, PureOS, Q4OS, QubesOS, Quirky, Raspberry Pi Desktop, ReactOS, Redcore, Rescatux, RevengeOS, RoboLinux, Rockstor, ROSA FRESH, Runtu, Sabayon, SalentOS, Salix, ScientificLinux, Siduction, Slackware, Slax, SliTaz, Solus, SolydK, SolydX, SparkyLinux, Springdale, StressLinux, SubgraphOS, SwagArch, Tails, Tanglu, Tiny Core, Trisquel, TrueOS, TurnKey Linux, Ubuntu, Ubuntu Budgie, Ubuntu Studio, UbuntuKylin, Uruk, VectorLinux, VineLinux, VoidLinux, Voyager, VyOS, WattOs, Xubuntu, Zentyal, Zenwalk, Zevenet, Zorin OS

[72] Cuando se trabaja desde la Web es recomendable usar el modo Privado o Incógnito para no guardar el historial de navegación, información introducida en los formularios y borrar al cerrar el navegador los datos de los sitios visitados. Pero recuerda que los sitios Web que visitamos sí guardan información de nuestra visita, nuestro proveedor de internet también guarda constancia de nuestra visita y si descargamos algo, esto no se borra al igual que el historial de descargas, además de las marcas de páginas o favoritos se conservarán al cerrar el navegador.

[73] Estos servicios son conocidos como computación en la nube (Cloud Computing).

Terminales de Linux en la Web

- https://www.tutorialspoint.com/execute_bash_online.php
- http://www.webminal.org/
- https://bellard.org/jslinux/
- https://codeanywhere.com/
- https://copy.sh/v86/
- https://www.masswerk.at/jsuix/
- https://linuxcontainers.org/lxd/try-it/
- http://cb.vu/

Editores BAHS en la Web

- https://www.shellcheck.net/
- https://www.learnshell.org/
- https://www.tutorialspoint.com/execute_bash_online.php
- https://paiza.io/en/projects/new?language=bash
- https://www.jdoodle.com/test-bash-shell-script-online
- http://rextester.com/l/bash_online_compiler

Usar Linux en Dispositivos Android En los dispositivos Android es posible usar un simulador de la línea de comandos del Shell usado en Linux, de forma que podremos introducir todos los comandos habituales para trabajar desde ahí en la comunidad de nuestra terminal Android. Uno de los paquetes más completo es:

https://termux.com

El paquete cuenta con una página *Wiki* en:

https://wiki.termux.com

Usando este paquete, las aplicaciones instaladas disponen de varias mejoras respecto al clásico Android Terminal Emulator, como el hecho de tener acceso a una gran biblioteca de paquetes de Linux para instalar desde la terminal -usando el comando *apt* (véase sección 13.12)-, así como algunos atajos de teclado transformados en combinaciones con los botones físicos de volumen y apagado de la terminal. Igualmente, es compatible con todo tipo de teclados físicos externos. Siendo posible trabajar con lenguajes como *NodeJ, Rubi, Python, C* y paquetes como *Nano, Vi, SSH, Git, Subversion, zsh Shell*, etc.

Usar Linux en Formato Live Linux es uno de los sistemas operativos pioneros en ejecutar de forma autónoma o sin instalar en la computadora, existen diferentes distribuciones Live -descargables para formato CD, DVD, USB- de sistemas operativos y múltiples aplicaciones almacenados en un medio extraíble, que pueden ejecutarse directamente en una computadora, estos se descargan de la Web generalmente en formato ISO[74], una de las listas más completas de versiones Live esta en:

https://livecdlist.com

En el caso de tener un archivo ISO de algún sistema operativo (por ejemplo ubuntu-11.10-desktop-i386.iso) y se quiere ejecutar su contenido desde una máquina virtual con QEMU/KVM sólo es necesario usar:

```
$ kvm -m 512 -cdrom ubuntu-11.10-desktop-i386.iso
```

en este ejemplo usamos en KVM la arquitectura por omisión y memoria de 512 MB (-m 512).

Knoppix es una versión Live ampliamente conocida y completa, esta se puede descargar de:

https://www.knopper.net/knoppix/

y usar mediante:

```
$ kvm -m 1024 -cdrom KNOPPIX_V8.2-2018-05-10-EN.iso
```

aquí se usa la arquitectura por omisión y memoria de 1024 MB.

[74] Una imagen ISO es un archivo informático donde se almacena una copia exacta de un sistema de archivos y de esta se puede generar una imagen para CDROM, DVD o USB.

Usar Máquinas Virtuales de Linux Existen diversos proyectos que permiten descargar decenas de máquinas virtuales listas para ser usadas, para los proyectos VirtualBox y VMWare (y por ende para KVM/QEMU), estas se pueden descargar de múltiples ligas, algunas de ellas son:

https://www.osboxes.org
https://virtualboxes.org/images/

Si desargamos y descomprimimos el archivo lubuntu1210.7z (véase sección 11.9), esto dejará la imagen de VirtualBox de LUBUNTU cuyo nombre es lubuntu1210.vdi. Entonces esta imagen la usaremos directamente en KVM/QEMU, mediante:

```
$ kvm -m 2000 -hda lubuntu1210.vdi
```

Nota: esta imagen usa como usuario y clave de acceso: lubuntu/lubuntu

Distribuciones de Sistemas Operativos Existen diversos sitios Web que están enfocados a explorar detalladamente cada distribución actual o antigua, a un nivel técnico acompañado de grandes y útiles análisis técnicos sobre los mismos, lo que facilita el aprendizaje puntual sobre que distribución usar o empezar a usar sin tanta incertidumbre.

- ArchiveOS https://archiveos.org
- Distro Chooser https://distrochooser.de/es/
- Distro Watch https://distrowatch.com
- Linux Distribution List https://lwn.net/Distributions/

4 Herramientas en Línea de Comandos

En esta sección mostraremos el uso de varios comandos útiles que se pueden usar desde la línea de comandos

4.1 Prompt de la Línea de Comandos

Si acostumbras trabajar en la línea de comandos, muy posiblemente uses el Shell Bash (Bourne Again Shell, derivado del Bourne Shell de Unix), con un simple echo de la variable $SHELL puedes determinarlo: echo $SHELL. Si es el caso, entonces tu Prompt[75] debe de verse parecido a este:

antonio@miMaquina:~$

Que indicaría al usuario (antonio) y el equipo en el que está (miMaquina), ~ indica HOME (en este caso /home/antonio) o directorio de inicio, esta parte cambia cada vez que se ingresa a otro directorio:

antonio@miMaquina:~$ cd /etc

antonio@miMaquina:etc$

Algo útil, pero porque mejor no personalizarlo a nuestro gusto, así que empecemos por partes.

Secuencias de escape para el Prompt El Prompt se establece a través de la variable de entorno PS1:

antonio@miMaquina;~$ echo $PS1

\u@\h:\W\$

[75] En el Bourne Shell y sus derivados como BASH el *Prompt* que nos permite escribir los diferentes comandos, generalmente termina con el carácter:

- $ para usuario sin privilegios
- # para el administrador, conocido como *root*

Se puede apreciar que se utilizan secuencias de escapes para ir construyendo el Prompt, cada secuencia se indica con '\' seguido de un comando como 'u' (user) o 'h' (Host), los demás caracteres como [,],@, espacio, etc. son opcionales y puedes elegirlos tu al acomodo que desees, las secuencias de escape son las siguientes:

- \a un carácter ASCII de ring
- \d la fecha actual en formato "dia_sem mes día", "dom nov 18"
- \e un carácter ASCII de escape
- \h el nombre del equipo hasta el primer ., ejemplo miMaquina de miMaquina.fciencias.unam.mx
- \H el nombre del equipo
- \n nueva línea
- \r retorno de carro, Enter
- \s el nombre del Shell
- \t el tiempo actual en formato de 24 horas HH:MM:SS
- \T el tiempo actual en formato de 12 horas HH:MM:SS
- \@ el tiempo actual en formato de 12 horas con am/pm
- \u el usuario actual
- \v la versión de Bash
- \V el número de release de batch, versión + parche
- \w el directorio de trabajo actual, Path
- \W el nombre del directorio actual
- \! el número en el historial del comando
- \# el número de comando de este comando
- \$ si el usuario es root (UID=0) se indica un '#' o usuario normal '$'

- \\ diagonal
- \[inicio de una secuencia de caracteres no imprimibles
- \] fin de la secuencia de caracteres no imprimibles

Conociendo lo anterior podemos ahora tener un nuevo Prompt:

antonio@miMaquina:~$ PS1='(\t)[\u-\W]\$> '

obteniendo:

(11:26:02)[antonio-etc]$>

Solo se indica el cambio a PS1, con PS1=' ', entre las comillas simples va la nueva secuencia que se desea, asi que personaliza el tuyo.

Añade color a tu Prompt El Shell está lleno comandos muy poco usados, uno de estos es tput, que permite cambiar las características o capacidades disponibles para la terminal, disponibles a través de la base de datos llamada terminfo. Entre las características (hay bastantes) que podemos modificar de una terminal están el color de fondo y de frente (Background y Foreground) del texto a través de las siguientes opciones:

- setaf [0-7] cambia el color de frente
- setab [0-7] cambia el color de fondo
- bold modo negritas
- dim modo de poco brillo
- sgr0 apaga las características o atributos que se hayan indicado previamente

En cuanto a los códigos de color son los siguientes

- 0 Negro
- 1 Rojo
- 2 Verde

- 3 Café
- 4 Azul
- 5 Morado
- 6 Cyan
- 7 Gris

Puedes probar en una terminal escribiendo lo siguiente: tput setaf 1 y el texto se cambiará a rojo y puedes añadir por ejemplo un fondo verde tput setb 2 y te dará un fondo verde para el texto. Así que digamos, en base al Prompt anterior, que se desea la hora en rojo y negritas, esto lo haría:

```
$ PS1='\[$(tput setaf 1)(\t)$(tput sgr0)\][\u-\W]\$> '
(12:06:43)[antonio-~]$>
```

mmmm, un poco complicado, veamos por partes:
\[inicio de secuencia de caracteres no imprimibles
$(tput setaf 1) cambia a color rojo el texto, $(comando) expande el resultado de un comando que se ejecuta
(\t) lo que se ve visible en pantalla (20:06:43)
$(tput sgr0) apagamos los atributos, si no lo hacemos todo quedará en rojo
\] termina la secuencia de caracteres no imprimibles
No es tan complicado una vez que entendemos lo que sucede. Y es posible agregar más características en un sola invocación de $(), $(tput bold; tput setaf 1).

Ahora bien, una vez cambiado tu Prompt, éste no permanecerá así, si cierras la sesión o la terminal y vuelves a ingresar, notarás que sigues con tu mismo y aburrido Prompt de siempre, el cambio a la variable PS1 hay que agregarlo a tu archivo de inicialización de tu sesión, generalmente '.bashrc' o '.bash_profile', para recargar el ambiente hacer:

```
$ source ~/.bashrc
```

Incluso podrías poner los colores en variables, para facilitar el uso de la definición de PS1:

```
# se añade lo siguiente a .bashrc
# colores del texto
rojo=$(tput setaf 1)
verde=$(tput setaf 2)
# colores de fondo
azulF=$(tput setab 4)
grisF=$(tput setab 7)
# sin color
sc=$(tput sgr0)
PS1='\[$rojo(\t)$sc\][\u-\W]\$> '
```

Aquí mostramos otra forma de códigos de color para Bash:

- 0;30 Negro
- 0;34 Azul
- 0;32 Verde
- 0;36 Cyan
- 0;31 Rojo
- 0;35 Púrpura
- 0;33 Café
- 0;37 Gris Claro
- 1;30 Gris Oscuro
- 1;34 Azul Claro
- 1;32 Verde Claro
- 1;36 Cyan Claro
- 1;31 Rojo Claro
- 1;35 Fucsia
- 1;33 Amarillo

- 1;37 Blanco

con ellos podemos personalizar los caracteres especiales de escape usando las siguientes secuencias de caracteres no imprimibles:

\[comienza un secuencia de caracteres no imprimibles

\] termina un secuencia de caracteres no imprimibles

Por ejemplo:

```
PS1='\[\e[0;31m\]\u\[\e[m\] \[\e[1;34m\]\w\[\e[m\] \[\e[0;31m\]\$\[\e[m\]\[\e[0;32m\] '
```

Este indicador tiene las características de que el nombre 'root' está en rojo, el directorio de trabajo en azul claro, un indicador # en rojo y la escritura de texto, verde

Estas son solo algunas secuencias de escape comunes para cambiar el formato del mensaje Bash. En mi caso prefiero tener un Prompt para mi y otro para el usuario root, esto se logra mediante:

```
if [ "$UID" = 0 ]; then
    PS1='\[\033[01;31m\]\u\[\033[01;32m\]@\h\[\033[00m\]:\[\033[01;34m\]\w\[\033[00m\]\n# '
else
    PS1='\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w\[\033[00m\]\n$ '
fi
```

obteniendo para el usuario antonio, algo como:

```
antonio@miMaquina:~
$
```

y para el usuario root, algo como:

```
root@miMaquina:/home/antonio/
#
```

Hay algunas secuencias más disponibles, podemos verlas en la página del manual de Bash.

4.2 Historia de Comandos

Cada vez que se entra en la terminal y se trabaja, esta es guardada en la historia de comandos, podemos acceder al historial usando las flechas para moverse entre los comandos tecleados o las teclas *Ctrl-r* para buscar mediante una cadena al comando tecleado.

El comando que nos permite ver la historia de comandos teclados es *history*, podemos usarlo mediante:

```
$ history
```

esto nos permite ver los comandos tecleados (según la configuración de *history* en el sistema guarda los últimos mil comandos) y podemos reejecutar alguno usando ! y el número de comando en el historial, por ejemplo:

```
$ !20
```

en caso de que necesitemos ejecutar alguno de los últimos comandos ejecutados, podemos usar !-N, por ejemplo el penúltimo usamos:

```
$ history !-2
```

también podemos borrar un determinado comando del historial usando su número en history:

```
$ history -d 20
```

y podemos borrar la historia mediante:

```
$ history -c
```

pero esto no impide que se siga grabando si continuamos trabajando en la terminal, para borrarla y que no guarde nada de lo que hagamos, usamos:

```
$ set +o history
$ history -cw
```

o forzar el borrado del historial y salir de sesión, mediante:

```
$ cat /dev/null > ~/.bash_history && history -wc && exit
```

Por otro lado, podemos cambiar el formato de visualización para conocer la fecha y hora del comando tecleado, mediante:

```
$export HISTTIMEFORMAT="%d/%m/%y %T "
```

otra opción es:

```
$export HISTTIMEFORMAT="%h/%d  - %H:%M:%S "
```

y ahora podemos ver el historial en el formato solicitado usando:

```
$ history
```

si queremos que sea de forma permanente, debemos agregarlo en ~/.bashrc. Además podemos configurar en ~/.bashrc algunos otros aspectos como:

- cambiar el tamaño máximo del archivo de history, mediante:

  HISTFILESIZE=50000

- el número de comando a recordar, usando:

  HISTSIZE=10000

  si lo ponemos en cero, se deshabilita el guardado de la historial:

  HISTSIZE=0

- decirle a history que no guarde duplicados, usando:

  HISTCONTROL=ignoredups

- que no guarde los comandos que inician con espacios, usando:

  HISTCONTROL=ignorespace

- que no guarde duplicados e ignore los que inician con espacio:

  HISTCONTROL=ignoreboth

- cambiar el nombre del archivo en que se guarda la historia, que por omisión es ~/.bash_history, usando:

  HISTFILE=nombre

- además podemos ignorar ciertos patrones o comandos de la historia, para ello usamos:

  HISTIGNORE="history"

de este modo *history* no aparecerá, pero si algo como «*history* | *less*» y similares, así que podemos usar un comodín para evitarlo:

  HISTIGNORE="history*"

o podemos añadir más comando separándolos por «:», por ejemplo:

  HISTIGNORE="history*:echo*:ps*"

- con la configuración por defecto de *history*, usar varias sesiones de forma simultánea supone un problema. Por un lado, el histórico de comandos se guarda cuando finalizamos la sesión, es decir, al hacer un «*exit*» en Bash. Por defecto, al guardar el histórico de sesión, los contenidos del archivo se sobreescriben así que es posible que los comandos ejecutados en una de las sesiones no queden almacenados en el histórico. Para solucionar este problema, podemos decir que añada los comandos en el archivo *history* en lugar de sobreescribirlos:

  shopt -s histappend

- además, podemos especificar que los comandos se almacenen en el archivo de *history* al momento de ser ejecutados en lugar de al finalizar la sesión. De este modo, sesiones simultáneas podrán visualizar en el histórico sus respectivas ejecuciones de comandos[76]:

  PROMPT_COMMAND="history -a"

[76] Debemos verificar antes que la variable de entorno PROMPT_COMMAND no tiene ningún valor asignado, así no lo perderemos al establecer esta configuración. En caso de que lo tenga, podemos concatenar ambos valores separados por «;».

Si queremos ver los cambios se apliquen inmediatamente, usamos:

```
$ source ~/.bashrc
```

Si deseamos conocer cuáles son los comandos usados y cuantas veces los hemos usado, escribimos:

```
$ history | awk '{print $2}' | sort | uniq -c | sort -nr
```

4.3 Grabar y Reproducir Contenido de la Terminal

El comando *history* es una gran utilidad de línea de comando que ayuda a los usuarios a almacenar los comandos utilizados anteriormente, aunque no almacena la salida de estos. En caso de requerirlo, podemos grabar lo que hagamos en la terminal de línea de comandos usando el comando *script* (este comando pertenece al paquete *util-linux*) que nos proporcionará una funcionalidad poderosa que nos ayuda a registrar todo lo que se visualiza en la terminal grabándolo en un archivo de registro *.log* en texto plano y otro archivo de tiempo que temporaliza los comandos que tecleamos y sus respectivas salidas.

Podemos reproducir lo que se grabó usando el comando *scriptreplay* que usa la información auxiliar de tiempo que temporaliza los comandos que tecleamos y sus respectivas salidas.

La sintaxis básica para iniciar a grabar es:

```
$ script –timing=archivo_tiempo.txt archivo_grabacion.log
```

esto nos permite grabar todo lo que hagamos en la terminal hasta que finalicemos la sesión usando:

```
$ exit
```

Si deseamos seguir grabando sobre un archivo ya existente usamos:

```
$ script -a –timing=archivo_tiempo.txt archivo_grabacion.log
```

Para reproducir lo grabado usamos:

```
$ scriptrelay –timing=archivo_tiempo.txt archivo_grabacion.log
```

También está disponible el comando *asciinema*, el cual permite al igual que *script* grabar la sesión, pero además se puede subir lo grabado al sitio Web *asciinema.org*. Para instalarlo usamos:

```
# apt install asciinema
```

Para iniciar la grabación usamos:

```
$ asciinema rec [nombre]
```

para reproducirlo, usamos:

```
$ asciinema play [nombre]
```

4.4 Alias a Comandos

Me gustaría crear un alias para el comando rm para tener un mensaje de confirmación antes de ejecutar este comando. Entonces podemos crear un alias como este:

```
$ alias rm='rm -i'
```

este es un alias temporal y dura hasta que cierras la terminal. Para guardar el alias de forma permanente, es necesario editar el archivo ~/.bashrc y agregar mi alias allí.

La estructura de alias es la siguiente:

```
alias name=value
alias name="command"
alias name="command arg1 arg2"
alias name="/path/to/Script"
alias name="/path/to/Script.pl arg1"
```

y podemos eliminarlos mediante:

```
unalias aliasname
```

Alias útiles

```
alias ls="ls –color=auto"
alias ll="ls -la"
alias l.="ls -d .* –color=auto"
alias la="ls -Ah"
alias l="ls -Cfh"
alias dir="dir –color=auto"
alias vdir="vdir -color=auto"
alias cd..="cd .."
alias ..="cd .."
alias ...="cd ../../../"
alias ....="cd ../../../../"
alias .....="cd ../../../../"
alias .4="cd ../../../../"
alias .5="cd ../../../../.."
alias grep="grep –color=auto"
alias egrep="egrep –color=auto"
alias fgrep="fgrep -color=auto"
alias diff="colordiff"
alias bc="bc -l"
alias mkdir="mkdir -pv"
alias df="df -H"
alias du="du -ch"
```

Confirmación de la acción

```
alias rm="rm -I –preserve-root"
alias mv="mv -i"
alias cp="cp -i"
alias ln="ln -i"
alias chown=2chown –preserve-root"
alias chmod="chmod –preserve-root"
alias chgrp="chgrp –preserve-root"
```

También podemos crear algunos comando nuevos, como este, que es una combinación de cd y ls, escribiendo en ~/.bashrc, lo siguiente:

```
cs() {
     cd "$@" && ls -a;
}
```

Alias a directorios también podemos hacer aliar a directorios como por ejemplo:

```
alias desk="cd ~/Desktop"
```

pero podemos hacer una función en ~/.bashrc. que guarde los directorios que más usamos, en el archivo ~/.directoriosGuardados, mediante:

```
# Permite guardar una trayectoria del árbol de directorios
guarda() {
    printf "$(pwd)\n" >> ~/.directoriosGuardados;
}
```

y en donde nos interese guardar la trayectoria usamos:

```
$ guarda
```

y con la función goto, nos permita movernos entre las diferentes trayectorias guardadas:

```
    # Permite cambiar entre las trayectorias guardadas por la
función: guarda
    function goto
    {
        local foo=$(sort ~/.directoriosGuardados | nl -w1)
        local REPLY
        echo -e "\n0\tSalir\n$foo\n"
        read -p "Introduzca el numero de directorio: "
        if [[ $REPLY =~^[0-"$(echo "$foo" | wc -l)"]$ ]]; then
            [[ $REPLY = "0" ]] && return
            cd "$(grep ^$REPLY <<<$foo | cut -f2)" && echo
"Ahora en: $(pwd)"
        else
            echo "No existe tal numero"
        fi
    }
```

de esta forma, podremos guardar múltiples trayectorias y cuando deseemos un cambio a una nueva trayectoria usamos:

```
$ goto
```

que nos mostrará las diferentes opciones guardadas y seleccionar usando el número que le corresponda o 0 para salir.

4.5 Ayuda de Comandos y Tipo de Archivos

Muchos comandos que podemos llegar a utilizar se pueden clasificar en categorías de acuerdo a su origen. Algunos están incorporados en el Shell, mientras otros provienen de un determinado paquete que hayamos instalado. También existe la posibilidad de que un comando sea en realidad un *alias* de otro comando con sus opciones.

type permite identificar el comando o comandos pasados como parámetro indicando la trayectoria si es comando externo o si es comando interno al Shell. Uso del comando type:

$ type [*opciones*] comando o comandos

Algunas opciones del comando *type* son:

- -P muestra la trayectoria completa del comando
- -p retorna el nombre del archivo en disco al cual pertenece o nada si no hay archivo
- -a muestra la mayor información posible del comando
- -t retorna el tipo de comando, no la trayectoria

Por ejemplo para el comando creado en la función *cs* de la sección anterior

$ type cs

desplegará:

```
cs is a function
cs() {
      cd "$@" && ls -a;
}
```

man muestra la documentación completa de todos los comandos, su sintaxis es:

$ man [*opciones*] [sección] comando

para conocer todas las páginas disponibles del sistema, usamos:

$ man -k .

por ejemplo, para *clear*:

$ man clear

para una versión corta de la ayuda, usamos:

$ man -f clear

también podemos buscar ayuda de algo que tiene que ver con alguna palabra, por ejemplo algo con "sound", usamos:

$ man -k sound

además podemos usar grep en conjunción con man para una ayuda rápida sobre alguna opción del comando buscado, por ejemplo:

$ man ls | grep – –g

El comando *man* tiene diversas secciones donde se puede solicitar que busque la información de un comando para evitar ambigüedades, estas son:

1. *bin*: Binarios esenciales para el funcionamiento del sistema
2. *sys*: Llamadas al sistema
3. *lib*: Funciones de las bibliotecas
4. *dev*: Archivos de dispositivos
5. *etc*: Archivos de configuración
6. *games*: Juegos

7. *mis*: Miscelánea
8. *sbin*: Binarios esenciales para administración y mantenimiento del sistema
9. *boot*: Información del Kernel del sistema

por ejemplo para conocer *man* en la sección 7, usamos:

```
$ man 7 man
```

También es común el uso de *–help* en el comando que necesitamos conocer su uso, por ejemplo:

```
$ clear –help
```

help proporciona ayuda de los comandos, con frecuencia puede sustituir al comando *man*. Por ejemplo, para conocer la lista de comandos que soporta:

```
$ help
```

info proporciona ayuda de los comandos al igual que *man* y *help*, su uso es similar:

```
$ info mkdir
```

pinfo navegador de visualización de información de un comando del sistema, ejemplo:

```
$ pinfo mkdir
```

whatis proporciona una ayuda breve de lo que hacen los comandos, sin mostrar opciones del comando, ejemplo:

```
$ whatis ls
```

apropos busca en las páginas del manual para la palabra clave o expresión regular que le pasemos como parámetro, ejemplo:

```
$ apropos chmod
```

whereis localiza el archivo binario, sus fuentes y las páginas de manual del comando, ejemplo:

```
$ whereis info
```

which sirve para averiguar dónde se encuentra instalado uno o más comandos y para ello busca en los directorios del sistema, ejemplo:

```
$ which chmod ls
```

si se quiere saber todos los lugares en donde esta el comando usamos:

```
$ which -a sync
```

otra opción es:

```
$ command -v sync
```

file determina el tipo de archivo y muestra el resultado. No hace falta que el archivo tenga una extensión para que *file* determine su tipo, pues la aplicación ejecuta una serie de pruebas sobre el mismo para tratar de clasificarlo, ejemplo:

```
$ file un_archivo_de_texto.txt
```

Si se tiene un archivo que contiene una lista de trayectorias a diferentes archivos, podemos pedirle a file que nos de información de dichos archivos, usando:

```
$ file -f listaArchivos.txt
```

Si se tiene un archivo compactado, podemos pedirle a file que nos indique con que fue compactado, usando:

```
$ file -z archivoCompactado
```

4.6 Redireccionando la Entrada y Salida Estándar

Cuando se abre un archivo en Linux, a cada archivo se le asignará un número entero y esta información se almacena en el Kernel. De esta forma el Kernel sabe qué archivos se abren y qué proceso los ha abierto. El número entero asignado es lo que llamamos un descriptor de archivo (en breve FD).

Los procesos pueden abrir archivos a discreción, pero la mayor parte de los procesos esperan a que estén abiertos tres descriptores de archivos (nume-rados 0, 1 y 2) cuando inician. Estos descriptores se conocen como entrada estándar (FD 0 Standar Input "Keyboard"), salida estándar (FD 1 Standard Output "Display Terminal") y error estándar (FD 2 Standard Error "Display Terminal"). Es común que los tres estén abiertos en la terminal del usuario. Así, el programa puede leer lo que el usuario teclea leyendo la entrada estándar, y puede enviar salidas a la pantalla del usuario escribiendo en la salida estándar. El descriptor de archivo de error estándar también esta abierto para escritura, y se usa para los mensajes de error. Podemos ver estos descriptores usando:

```
$ ls -al /dev/std*
```

Standard input la entrada estándar, en inglés *standard input* (mejor conocido como *stdin*) es el mecanismo por el cual un usuario le indica a los programas la información que estos deben procesar. Por omisión, el teclado es la entrada estándar. La entrada estándar representa los datos que necesita una aplicación para funcionar, como por ejemplo un archivo de datos o información ingresada desde la terminal y es representado en la terminal como el tipo 0.

Standard output la salida estándar, en inglés *standard output* (mejor conocido como *stdout*) es el método por el cual el programa puede comunicarse con el usuario. Por omisión, la salida estándar es la pantalla donde se ejecutaron las instrucciones. La salida estándar es la vía que utilizan las aplicaciones para mostrarte información, allí podemos ver el progreso o simplemente los mensajes que la aplicación quiera darte en determinado momento y es representado en la terminal como el tipo 1.

Standard error por último existe un flujo conocido como error estándar, en inglés *standard error output* (mejor conocido como *stderr*) que es

utilizado por las instrucciones para desplegar mensajes de error que surjan durante el transcurso de su ejecución. Al igual que *stdout*, el error estándar será la pantalla donde se procesaron las instrucciones. El error estándar es la forma en que los programas te informan sobre los problemas que pueden encontrarse al momento de la ejecución y es representado en la terminal como el tipo 2.

Operadores de redirección a modo de resumen, indicamos las posibles formas de direccionamiento y los símbolos que se utilizan para lograrlo:

- > Redirecciona stdout hacía un archivo, crea archivo si no existe, sobreescribe si existe.
- >> Redirecciona stdout hacía un archivo, crea archivo si no existe, concatena si existe.
- < Redirecciona stdin desde un archivo. El contenido de un archivo es la entrada del comando.
- 2> Redirecciona stderr hacía un archivo, crea archivo si no existe, sobreescribe si existe.
- 2>> Redirecciona stderr hacía un archivo, crea archivo si no existe, concatena si existe.
- p>&q Fusiona la salida del flujo p con el flujo q ($p, q \in [0, 1, 2]$).
- p<&q Fusiona la entrada del flujo p con el flujo q ($p, q \in [0, 1, 2]$).

Otros redireccionamientos que no utilizan descritopres

- << Conocido como HERE-DOCUMENT o HereDoc
- <<< Conocido como HERE-STRING

Todos estos tipos son representados físicamente como archivos en el sistema, todo en GNU/Linux son archivos. Así, una redirección consiste en trasladar la información de un tipo a otro, por ejemplo de la salida estándar a la entrada estándar o del error estándar a la salida estándar. Esto lo logramos usando el símbolo >. Por ejemplo, para redireccionar la salida de un comando y volcarla a un archivo bastaría con ejecutar:

```
$ ls -la > archivo.txt
```

Sin embargo, cada vez que ejecutemos el comando, el contenido del archivo *archivo.txt* será reemplazado por la salida del comando ls. Si queremos agregar la salida del comando al archivo, en lugar de reemplazarla, entonces ejecutamos:

```
$ ls -la >> archivo.txt
```

Utilizar el comando touch para crear un archivo vació, es una práctica común que también puede realizarse con el operador de redireccionamiento >, mediante:

```
$ > documento1
```

Lo interesante es que, además de la salida estándar, también podemos redireccionar el error estándar y la entrada estándar. Si queremos forzar a que un programa nos imprima en pantalla los errores que consiga durante su ejecución podemos redireccionar el error estándar hacia la salida estándar. Eso lo logramos ejecutando:

```
$ programa 2>&1
```

¿Recuerdan que líneas arriba se comentó que GNU/Linux identifica a cada tipo con un número? Bueno, aquí es donde esos números cobran sentido. El tipo 2 es el error estándar y el tipo 1 es la salida estándar. En los ejemplos previos no tuvimos la necesidad de especificar el tipo 1 porque la terminal lo asume pero pudimos expresarlos explícitamente de la siguiente manera:

```
$ ls -la 1> archivo.txt
$ ls -la 1>> archivo.txt
```

Podemos, por ejemplo, contar las líneas que tiene un archivo redireccionando la entrada estándar de wc hacía un archivo de texto. Así:

```
$ wc < archivo.txt
```

También podemos redireccionar la entrada y salida en el mismo comando, por ejemplo:

```
$ sort < lista_desordenada.txt > lista_ordenada.txt
```

o podemos redireccionar la salida de errores y la salida al mismo archivo, por ejemplo:

```
$ comando > todo.txt 2>&1
```

la otra opción (menos común) es utilizar 1>&2, que indicaría redirecciona la salida del comando stdout hacía donde stderr apunte.

```
$ comando 2> errores.txt 1>&2
```

o redireccionar la salida y salida de error a distintos archivos, por ejemplo::

```
$ comando > salida.txt 2> errores.txt
```

Otra opción para guardar la salida de un comando es usar *logsave*, por ejemplo

```
$ logsave salida.txt ls -al
```

si necesitamos adicionar otras salidas usamos:

```
$ logsave -a salida.txt date
```

siendo posible redireccionar el error, usando:

```
$ logsave salida.txt ls -al /dev/null 2>&1
```

Utilizando << y <<< Es cuando un bloque de texto puede ser redireccionado a un comando o archivo de una manera interactiva. El Here Document funciona indicando un DELIMITADOR que no es más que una palabra o cadena cualquiera que cierra el bloque de texto que se desea redireccionar. Veámoslo con ejemplos:

```
$ wc << fin
> uno dos tres ⟵ de manera interactiva el usuario teclea el bloque de texto, el delimitador es la palabra "fin""
> cuatro cinco
> seis
> fin
3 6 31
```

El comando wc recibe (stdin) un bloque de texto teclado por el usuario. Indicando el fin del bloque de texto con el delimitador fin. Cuando se recibe este, entonces wc ejecuta su función, al recibir por completo su entrada a examinar, en este caso 3 líneas, 6 palabras, 31 caracteres.

```
$ cat << TERMINA > documento1
uno dos
tres
cuatro cinco
TERMINA
$
$ cat documento1
uno dos
tres
cuatro cinco
```

En este caso el comando *cat* recibe el redireccionamiento de hereDoc con del delimitador TERMINA, cuando encuentra el delimitador deja de recibir entradas y direcciona lo ingresado al archivo "documento1". Si se quisiera agregar o concatenar a "documento1" se utilizará cat << TERMINA >> documento1

La diferencia entre << HERE DOCUMENT y <<< HERE STRING es que en HERE STRING no se pasa un delimitador, sino que se pasa una cadena que es interpretada por el comando al que se le redirecciona como un argumento(s). Este argumento puede ser una variable de Shell que puede expandirse. Es decir, HERE STRING se compone de una cadena o string de posibles argumentos o variables:

```
$ bc <<< 5*5
25
$ sed 's/hola/Hola/' <<< "hola mundo"
Hola mundo
```

Redirección Mediante Pipe las tuberías (pipe) unen la salida estándar de un comando con la entrada estándar de otro, es decir, la salida de un comando se emplea como entrada del siguiente. Para ello se emplea el símbolo pipe "|", su sintaxis es:

```
$ comando1 | comando2 | comando3 | ... | comandoN
```

El orden de los comandos en una tubería es crucial, ya que determina el flujo de datos. Cada comando procesa los datos que recibe del comando anterior y pasa su salida al siguiente comando de la cadena. La utilización de tuberías evita la generación constante de archivos intermedios reduciendo el tiempo de procesamiento.

El siguiente ejemplo lista todos los procesos en ejecución en el sistema mediante el comando "ps -ef". Utilizando el pipe, el resultado viaja como entrada hacia el comando "grep" que se quedará solo con aquellas líneas donde aparezca la palabra "antonio". Este nuevo resultado del comando "grep" se envía como entrada del comando "sort" que se encargará de ordenar el resultado y mostrarlo por la pantalla de la terminal. Es decir que el último comando es el que hace uso de la salida estándar.

```
$ ps -ef | grep antonio| sort
```

En este otro ejemplo, el comando "cat" abre el archivo "nombres.dat" y su contenido lo envía, utilizando el pipe, como entrada del comando "cut" el cual recortará las líneas entre los caracteres 10 y 80. Este nuevo resultado, se envía al comando "sort" como entrada de información y haciendo uso de la opción "-r" lo ordenará en orden inverso al orden por defecto y luego muestra el resultado por la pantalla de la terminal.

```
$ cat nombres.dat | cut -c10-80 | sort -r
```

Las tuberías se pueden combinar con la redirección de entrada/salida para crear potentes flujos de trabajo de procesamiento de datos. Los operadores > y < le permiten redirigir la salida de un comando a un archivo o recibir información de un archivo, respectivamente.

Redireccionar la salida a un archivo

```
$ comando1 | comando2 > salida.txt
```

Tomar la entrada de un archivo

```
$ comando1 < entrada.txt | comando2
```

También puede redirigir el error estándar (stderr) 2> si necesita separar los mensajes de error de la salida normal.

```
$ comando1 2> errores.txt | comando2
```

A continuación se muestra un ejemplo que combina tuberias, redirección y procesamiento de texto para extraer y formatear información específica de un archivo de registro:

```
$ grep "error" log.txt | awk '{print $3, $5}' | sort | uniq >
Errores_unicos.txt
```

Este comando:

- Filtra líneas que contienen "error" del log.txt uso grep
- Canaliza la salida a awk, que imprime los campos (columnas) tercero y quinto de cada línea.
- Ordena la salida usando sort
- Elimina líneas duplicadas con uniq
- Redirige la salida final a Errores_unicos.txt

Redirección Mediante el Comando tee existe un comando que permite desviar una copia o bifurcación de la salida de un comando hacia un archivo sin alterar la entrada del siguiente en una tubería. Se trata del comando "tee"

Para entenderlo mejor, en el siguiente ejemplo, se almacena en en el archivo "conectados.dat" el resultado del comando "who" y, sin que este resultado sufra ninguna alteración por parte del comando "tee", se pasa mediante un segundo pipe hacia el comando "wc", quien se encargará de contar las líneas del resultado que produjo el comando "tee".

```
$ who | tee conectados.dat | wc -l
```

otros ejemplos son:

```
$ ps auxww | tee salida.log
$ ls -alh | tee archivo.txt | grep python
$ ls -alh | grep python | tee archivo.txt
$ ls -alh | grep python | tee archivo1.txt archivo2.txt archivo3.txt
$ ls -alh 2>&1 | tee archivo.txt
```

Redirección hacia el dispositivo nulo como se comentó, la salida estándar está asociada al descriptor de archivos 1 y la salida de errores está asociada al descriptor de archivos 2. Si bien para ambas salidas se utiliza el mismo dispositivo físico que es la pantalla de la terminal, ambos se gestionan de manera independiente. Son dos archivos diferentes.

Debes tener en claro que un comando puede generar una salida positiva, que sería aquello que se espera que haga, y eso va a parar a la salida estándar. Pero también puede generar un mensaje de error, y eso va a parar a la salida de errores. En ambos casos, el efecto es verlo reflejado en la pantalla de la terminal, pero internamente están en dos archivos independientes.

Otra cosa a tener en claro es que cuando utilizas las redirecciones de ">" (símbolo mayor) o "|" (pipe), lo que viaja a través de ellas es el resultado de la ejecución de un comando que hubiera ido normalmente a la salida estándar. Los mensajes de errores no viajan a través de las redirecciones a menos que fusiones las salidas.

Pero algo interesante para contar, es que en ocasiones podría no interesarte que los mensajes de errores se muestren en la pantalla de la terminal por lo que es posible reasignar el descriptor de archivos 2 de la salida de errores

estándar hacia un dispositivo que no exista físicamente. A este dispositivo se lo conoce como "dispositivo nulo" y su descriptor está en el directorio /dev.

En el siguiente ejemplo, con el comando "cat" intentaremos abrir el archivo "noexiste.txt" el cual no existe físicamente. En condiciones normales, verías en la pantalla de la terminal el mensaje que indica que "cat" no puede abrir el archivo "noexiste.txt". Este mensaje es un mensaje de error y por consiguiente fue a parar a la salida de errores estándar, en consecuencia lo ves en la pantalla de la terminal.

Pero, mediante la redirección "2>/dev/null", le indicamos al intérprete de comandos que está haciendo la ejecución del comando que todo lo que vaya a parar al descriptor de archivos 2, sea redireccionado hacia el dispositivo nulo "*/dev/nul*l". Y dado que este dispositivo no existe, entonces el mensaje de error se pierde y no se muestra ni almacena en ningún lado.

```
$ cat noexiste.txt 2>/dev/null
```

También podemos hacer algo muy común en la administración de sistemas, descartar el error estándar de un proceso. Para eso ejecutamos:

```
$ programa 2> /dev/null
```

O incluso descartar su salida estándar:

```
$ programa > /dev/null
```

En GNU/Linux, */dev/null* es un archivo especial al que se envía cualquier información que quiera ser descartada. Aunque al principio no lo parezca, el uso del dispositivo nulo es muy útil.

4.7 Metacarácter o Shell Globbing

los metacaracteres o Shell Globbing son caracteres que tienen un significado especial en la línea de comandos, para usar estos caracteres como caracteres ordinarios en la línea de comandos, debes marcarlos de manera especial para que el Shell no los interprete. Por ejemplo:

- La diagonal inversa (\) siempre sirve como carácter de escape para uno o más caracteres que le suceden, lo cual le da a esos caracteres un significado especial. Si un carácter es un metacaracter, el significado

especial es el carácter mismo. Por ejemplo \\ usualmente significa una solo \ y \$ el signo de pesos. En estos casos, la diagonal inversa le quita su significado a los caracteres.

- Cuando un texto se encierra entre comillas (" "), la mayoría de los de los metacaracteres que estén en dicho texto son tratados como caracteres normales, excepto por $, que generalmente indica sustituciones a realizar.

- Otra forma de citar muy similar a la anterior, pero más fuerte es usar comillas sencillas (' ') ya que ni siquiera el carácter $ es interpretado

También existen otros metacaracteres que son comodines que el sistema permite usar para especificar los nombres de archivos que satisfacen el filtro especificado a la hora de buscar, eliminar o filtrar nombres de archivo, estos metacaracteres son: *, ?, [ ] y [^][77].

- ***** Se utiliza para reemplazar cero o más caracteres. Puede ser sustituido por cualquier cadena de caracteres, ejemplos:

  Muestra el contenido de las carpetas que contengan archivos de extensión *txt*:

  ```
  $ ls *.txt
  ```

  Lista todos los archivos que se llamen *archivo* sin importar su extensión:

  ```
  $ ls archivo.*
  ```

  Muestra todos los archivos con extensión *jpg* y que su nombre tenga al final "*chivo*":

  ```
  $ ls *chivo.jpg
  ```

  Muestra todos los archivos que inicien con a y tengan la extensión *png*:

  ```
  $ ls a*.png
  ```

[77] Véase también el uso de las secuencias (véase 4.7).

- **?** Sustituye un carácter cualquiera, ejemplos:

  Muestra todos los archivos empiecen con letras o números pero que luego de ellos tengan los valores "*b4ts.txt*":

  $ ls ?b4ts.txt

  Muestra todos los archivos que inicien con *ab*, siga cualquier letra, número o carácter y finalice con *ts.txt*:

  $ ls ab?ts.txt

  Muestra todos los archivos de tres letras que en medio tenga una letra *i*:

  $ ls ?i?

  Muestra todos los archivos de cuatro letras cualesquiera con extensión *txt*:

  $ ls ????.txt

- [ ] Se usa para definir rangos o conjuntos de caracteres a localizar, para definir los rangos se debe usar el guión -, si son varios caracteres se separan por coma, ejemplos:

  Selecciona el rango de letras mayúsculas del alfabeto [:upper:] o más específicamente [A-Z], muestra todos los archivos que inicien con una letra mayúscula:

  $ ls [A-Z]*

  Selecciona el rango de letras minúsculas del alfabeto [:lower:] o más específicamente [a-z], muestra todos los archivos que inicien con una letra minúscula:

  $ ls [a-z]*

  Selecciona el rango de letras mayúsculas y minúsculas del alfabeto [:alpha:] o más específicamente [a-zA-Z], muestra todos los archivos que inicien con una letra minúscula o mayúscula:

```
$ ls [a-zA-Z]*
```

Selecciona el rango de dígitos [:digit:] o más específicamente [0-9], muestra todos los archivos que inicien con un dígito:

```
$ ls [0-9]*
```

Selecciona el rango de letras mayúsculas, minúsculas y dígitos del alfabeto [:alnum:] o más específicamente [a-zA-Z0-9], muestra todos los archivos que inicien con una letra mayúscula, minúscula o dígito:

```
$ ls [:alnum:]*
```

Selecciona el rango de caracteres de puntuación del alfabeto [:puncht:] o más específicamente " ! # $ % & ' ( ) * + , - . / : ; < = > ? @ [\ ] _ ´ { | } ~ ", muestra todos los archivos que inicien con un carácter de control:

```
$ ls [:puncht:]*
```

Muestra todos los archivos que comiencen por *z* o *v* sin importar la extensión:

```
$ ls [zv]*
```

Muestra todos los archivos que comiencen por z o v y terminen con la extensión *.txt*:

```
$ ls [zv]*.txt´
```

Lista todos los archivos de cualquier extensión que tengan los rangos establecidos entre los corchetes:

```
$ ls archivo[12].*
```

Muestra la lista de todos los archivos que cumplan con el rango de "a-f" sin importar la extensión o el nombre:

```
$ ls [a-f]*
```

Muestra la lista de todos los archivos que inicien con cualquier cosa, pero que terminen con una letra mayúscula:

```
$ ls *[A-Z]
```

Muestra la lista de todos los archivos que inicien con una letra minúscula, tenga después una letra mayúscula, continué con cualquier carácter, después tenga una letra a, b, c-f, z y siga con cualquier cantidad de caracteres:

```
$ ls [a-z][A-Z]?[a,b,c-f,z]*
```

- [^] Este caso es contrario al anterior, este representa que se busque algo exceptuando lo que se encuentra entre los corchetes, también trabaja con rangos.

  Muestra los archivos que no empiecen con una letra minúscula pero que tengan extensión *.txt*:

  ```
  $ ls [^a-z]*.txt
  ```

- [**!**] Este caso igual al anterior, este representa que se busque algo exceptuando lo que se encuentra entre los corchetes, también trabaja con rangos.

  Muestra los archivos que no empiecen con una letra minúscula pero que tengan extensión *.txt*:

  ```
  $ ls [!a-z]*.txt
  ```

Secuencias Como parte del BASH podemos usar el generador de secuencias, como por ejemplo:

```
$ echo {1..10}
```

que muestra la secuencia de los números de 1 a 10, si ahora probamos:

```
$ echo {1..10..2}
```

visualizará los números $1, 3, 5, 7, 9$. Es decir, iniciará con el primer número en la secuencia y terminará en el segundo de la secuencia con incrementos del último número de la secuencia. También podemos hacerlo en orden inverso mediante:

```
$ echo {10..1..2}
```

que nos entregará los números $10, 8, 6, 4, 2$. También podemos usar relleno con ceros, por ejemplo:

```
$ echo {000..121..2}
```

el cual imprimirá los números de 0 a 121 con saltos de dos en dos, como: 000 002 004 006 ... 050 052 054 ... 116 118 120. Y si necesitamos números negativos, podemos usar algo como:

```
$ echo {3..-4}
```

también podemos usarlos para trabajar secuencias con letras, por ejemplo:

```
$ echo {a..z}
```

imprimirá las letras a hasta la z, o este otro ejemplo:

```
$ echo {n..z} {a..m}
```

imprimirá las letras n a la z, y a continuación de a a la m.

Con este generador de secuencias numéricas podemos aplicar a comandos, como por ejemplo:

```
$ mkdir {2009..2019}_Facturas
```

que creará los directorios de 2009_Facturas, hasta 2019_Facturas. También lo podemos usar para borrar archivos, como por ejemplo:

```
$ rm cuadros_{043..61..3}
```

Este generador de secuencias podemos usarlo también en modo texto, por ejemplo:

```
$ touch archivo_{a..z}.txt
```

creará el archivo *archivo_ a.txt* hasta *archivo_ z.txt*. También es posible usar algo como {Z..a} pero generará caracteres no alfanuméricos. Otros usos son:

```
$ touch {blahg, splurg, mmmf}_file.txt
```

creará los archivos *blahg_file.txt, splurg_file.txt* y *mmmf_file.txt*.
Si queremos hacer:

```
$ cp -v file1.txt file1.txt.bak
```

lo podemos lograr, usando:

```
$ cp -v file1.txt{,.bak}
```

Otros ejemplos:

```
$ cp archivo{,.bak}
$ ls {????.txt, *.php}
$ ls {*.txt. *.php}
$ mv proyecto/{nuevo/viejo}/dir/otrodir
$ rm d{01..20}/archivo
$ touch {a,b,c}.{rs,cpp}
$ git add {main,x{1,2}}.rs
$ ls -l ~/{Downloads,Pictures}/*.{pdf,png}
$ mkdir -p ~/test/{etc/x1,lib,usr/{x2,x3},bin,tmp/{Y1,Y2,Y3/z},opt,var}
```

4.8 Nombres de Archivos y Directorios con Caracteres Especiales

Una de las preguntas más obvias aquí es: ¿quién crea/trata con archivos/nombres de directorios que tienen un (#), un punto y coma (;), un guión (-) o cualquier otro carácter especial?

Estoy de acuerdo con usted en que dichos nombres de archivos no son comunes, pero su Shell no debería fallar o darse por vencido cuando tenga que lidiar con dichos nombres. También hablando técnicamente, todo, ya sea una carpeta, un controlador o cualquier otra cosa, se trata como un archivo en Linux.

Nombre de Archivo que Inicia con un Guión en Linux, los nombres de archivos que comienzan con un guión ("-") a veces pueden causar problemas al trabajar con ellos porque el guión inicial puede malinterpretarse como una opción o ser marcado por las utilidades de línea de comandos.

Crear Archivo que Inicia con un Guión si deseamos cree un archivo que comience con un guión (-), digamos -abx.txt usando el comando:

$ touch -abc.txt

el sistema nos mandará un error. El motivo del error es que el Shell interpreta cualquier cosa después de un guión (-) como una opción y, obviamente, no existe tal opción, de ahí el error. Para resolver tal error, tenemos que decirle al Shell que no interprete nada después del carácter especial (aquí guión), como una opción.

Hay dos formas de resolver este error como:

```
$ touch - -    -abc.txt
$ touch ./-abc.txt
```

Podemos verificar que el archivo así creado mediante las dos formas anteriores ejecutando los comandos ls o ls -l para obtener un listado largo.

Editar Archivo que Inicia con un Guión para editar un archivo con un nombre de archivo con guiones, podemos utilizar varios editores de texto disponibles. Sin pérdida de generalidad, aquí hay un ejemplo usando el editor de texto nano:

```
$ nano - -   -abc.txt
$ nano ./-abc.txt
```

Cambiar Nombre Archivo que Inicia con un Guión para cambiar el nombre de un archivo que inicia con guiones, puede usar el comando mv como se muestra. Por ejemplo, para cambiar el nombre de un archivo llamado "-abc.txt" a "-a.txt", usaría:

```
$ mv - -   -abc.txt -a.txt
$ mv ./-abc.txt ./-a.txt
```

Eliminar Archivo que Inicia con un Guión para eliminar un archivo con un nombre de archivo discontinuo, puede usar el comando rm como se muestra.

```
$rm - -   -abc.txt
$ rm ./-abc.txt
```

Si tiene muchos archivos en una carpeta cuyo nombre inicia con guión y desea eliminarlos todos a la vez, podemos hacer lo siguiente:

```
$ rm ./-*
```

Se sigue la misma regla comentada anteriormente para cualquier número de guiones en el nombre del archivo y su aparición. A saber, -a-b-c.txt, ab-c.txt, abc-.txt, etc.

Se sigue la misma regla que se analizó anteriormente para el nombre de la carpeta que tiene cualquier número de guiones y su aparición, excepto el hecho de que para eliminar la carpeta debe usar 'rm -rf' como:

```
$rm -rf - -   -abc
$ rm -rf ./-abc
```

Nombre de Archivo con "#" en Linux, el carácter "#" no está restringido ni reservado para ningún propósito específico en los nombres de archivos. Puede utilizar el carácter "#" como cualquier otro carácter alfanumérico en un nombre de archivo. Sin embargo, vale la pena mencionar que algunos caracteres especiales, incluido el "#", tienen significados especiales en ciertos contextos o cuando se usan con comandos o utilidades específicas.

Por ejemplo, el carácter "#" se usa comúnmente en Scripts de Shell para indicar comentarios. Si está escribiendo un Script de Shell y desea incluir el carácter "#" en un nombre de archivo, es recomendable utilizar caracteres de escape o comillas correctamente para evitar cualquier interpretación no deseada como comentario.

Crear un Archivo con "#" si deseamos cree un archivo que comience con un guión (#), digamos #abx.txt usando el comando:

```
$ touch #abc.txt
```

el sistema nos mandará un error. El motivo del error anterior es que Shell interpreta #abc.txt como un comentario y, por lo tanto, lo ignora. Entonces, el comando touch se pasó sin ningún operando de archivo y, por lo tanto, surge el error.

Para resolver dicho error, puede pedirle al Shell que no interprete # como un comentario.

```
$ touch ./#abc.txt
$ touch '#abc.txt'
```

y podemos verificar el archivo recién creado con ls.

Crear archivo Archivo con "#" ahora cree un archivo cuyo nombre contenga # en cualquier lugar excepto al principio.

```
$ touch ./a#bc.txt
$ touch ./abc#.txt
$ touch 'a#bc.txt'
$ touch 'abc#.txt'
```

¿Qué sucede cuando crea dos archivos (digamos a y #bc) a la vez?

```
$ touch un.txt #bc.txt
```

Obviamente, en el ejemplo anterior solo se creó el archivo 'a' y el archivo '#bc' se ignoró. Para ejecutar la situación anterior con éxito podemos hacer:

```
$ touch a.txt ./#bc.txt
$ touch un.txt '#bc.txt'
```

finalmente, podemos verificar el archivo que acaba de ser creado usando el comando ls.

Cambiar el Nombre o Copiar el Archivo con "#" para cambiar el nombre de un archivo con "#", puede usar el comando mv como se muestra:

```
$ mv ./#bc.txt ./#cd.txt
$ mv '#bc.txt' '#cd.txt'
```

Para copiar un archivo con "#", puede usar el comando cp como se muestra:

```
$ cp ./#cd.txt ./#de.txt
$ cp '#cd.txt' '#de.txt'
```

Editar o eliminar Archivo con "#" para editar un archivo con "#", puede usar el editor nano o vim como se muestra:

```
$ vi ./#cd.txt
$ vi '#cd.txt'
```

Para eliminar un archivo con "#", puede usar el comando rm como se muestra:

```
$ rm ./#bc.txt
$ rm '#bc.txt'
```

Para eliminar todos los archivos que tienen (#) en el nombre del archivo, puede usar:

```
$ rm ./#*
```

Nombre de Archivo con ";" en caso de que no lo sepas, el punto y coma actúa como separador de comandos en BASH y quizás también en otro Shell. El punto y coma le permite ejecutar varios comandos a la vez y actúa como separador. Cree un archivo que tenga un punto y coma:

```
$ toque ./';abc.txt'
$ toque ';abc.txt'
```

Nota: Hemos incluido el nombre del archivo entre comillas simples ' '. Le dice a BASH que; es parte del nombre del archivo y no un separador de comandos. El resto de la acción (es decir, copiar, mover, eliminar) en el archivo y la carpeta que tiene un punto y coma en su nombre se puede realizar directamente encerrando el nombre entre comillas simples.

Caracteres especiales en Nombres de Archivos en Linux, los nombres de archivos pueden contener la mayoría de los caracteres especiales, incluidos espacios, puntos, guiones, guiones bajos e incluso símbolos como @,!, $ y %.

Aquí hay una lista de caracteres especiales que debe usar con precaución o considerar utilizar como escape cuando los use en nombres de archivos:

Signo (+) en el Nombre del Archivo para crear un archivo con signo (+), podemos usar el comando táctil junto con el nombre del archivo entre comillas simples:

```
$ touch '+tony.txt'
```

Signo ($) en Nombre de Archivo para crear un archivo con el signo de ($) , podemos usar el comando táctil junto con el nombre del archivo entre comillas simples:

```
$ touch '$tony.txt'
```

Porcentaje (%) en Nombre de Archivo para crear un archivo de signo (%), podemos utilizar:

```
$ touch '%tony.txt'
```

Asterisco (*) en el Nombre del Archivo para crear un archivo con signo (*) , podemos utilizar.

```
$ touch '*tony.txt'
```

Nota: Cuando tenga que eliminar un archivo que comienza con *, nunca utilice los siguientes comandos para eliminar dichos archivos.

```
$ rm *
$rm -rf *
```

En su lugar, utilizar:

```
$ rm ./*.txt
```

Signo de (!) en el Nombre del Archivo simplemente incluya el nombre del archivo entre comillas simples y el resto de las cosas serán iguales:

```
$ touch '!tony.txt'
```

Signo de (@) en Nombre de Archivo nada adicional, trate un nombre de archivo que tenga el signo @ como un archivo normal:

```
$ touch '@tony.txt'
```

Signo de (^) en Nombre de Archivo no se requiere atención adicional; use un archivo que tenga ^en el nombre del archivo como un archivo normal.

```
$ touch '^tony.txt'
```

Signo (&) en Nombre de Archivo el nombre del archivo debe estar entre comillas simples y estará listo para comenzar:

```
$ touch '&tony.txt'
```

Paréntesis () en Nombre de Archivo si el nombre del archivo tiene paréntesis, debe encerrarlo entre comillas simples:

```
$ touch '(tony.txt)'
```

Signo de llaves ({}) en nombre de archivo no se necesita cuidado adicional. Simplemente trátelo como un archivo más:

```
$ touch {tony.txt}
```

Signo de (<>) en el nombre de archivo un nombre de archivo que tenga (<>) debe estar entre comillas simples:

```
$ touch '<tony.txt>'
```

Signo de ([]) en Nombre de Archivo trate los nombres de archivos que tienen corchetes como archivos normales y no necesitará prestarles especial atención:

```
$ touch [tony.txt]
```

Signo (_) en Nombre de Archivo son muy comunes y no requieren nada extra. Simplemente haga lo que habría hecho con un archivo: normal.

```
$ touch _tony.txt
```

Signo (=) en Nombre de Archivo Tener un signo igual no cambia nada, puedes usarlo como un archivo normal:

```
$ touch =tony.txt
```

Signo (\) en Nombre de Archivo la barra invertida le dice al Shell que ignore el siguiente carácter. Debe encerrar el nombre del archivo entre comillas simples, como hicimos en el caso del punto y coma. El resto de las cosas son sencillas:

```
$ touch '\tony.txt'
```

Signo (/) en Nombre de Archivo no puede crear un archivo cuyo nombre incluya una barra diagonal (/), hará que el sistema de archivos tenga un error. No hay forma de escapar de una barra diagonal.

Signo (?) en Nombre del Archivo nuevamente, un ejemplo en el que no es necesario hacer ningún intento especial. Un nombre de archivo que tiene un signo de interrogación se puede tratar de la forma más general:

```
$ touch '?tony.txt'
```

Signo (.) en Nombre del Archivo los archivos que comienzan con un punto (.) son muy especiales en Linux y se llaman archivos de puntos. Son archivos ocultos generalmente archivos de configuración o de sistema. Debe utilizar el modificador '-a' o '-A' con el comando ls para ver dichos archivos.

Crear, editar, cambiar el nombre y eliminar dichos archivos es sencillo.

```
$ touch '.tony.txt'
```

Nota: En Linux, puede tener tantos puntos (.) como necesite en un nombre de archivo. A diferencia de otros puntos del sistema en el archivo, los nombres no significan separar nombres y extensiones. Puede crear un archivo que tenga varios puntos como:

```
$ touch 1.2.3.4.5.6.7.8.9.10.txt
```

Signo (,) en Nombre de Archivo puedes tener una coma en un nombre de archivo, tantos como quieras y no necesitas nada adicional. Simplemente hágalo de la manera normal, como el nombre de archivo simple:

```
$ touch ',tony.txt'
$ touch ',tony,.txt'
```

Signo (:) en Nombre de Archivo puede tener dos puntos en un nombre de archivo, tantos como desee y no necesita nada adicional. Simplemente hágalo de la manera normal, como el nombre de archivo simple:

```
$ touch :12.txt
$ touch ':tony:.txt'
```

Signo (" ") en Nombre de Archivo para tener comillas en el nombre del archivo, debemos utilizar la regla de intercambio. Es decir, si necesita tener una comilla simple en el nombre del archivo, encierre el nombre del archivo entre comillas dobles y si necesita tener una comilla doble en el nombre del archivo, enciérrela con una comilla simple.

```
$ touch "15'.txt"
$ touch 'tony".txt'
```

Signo (~) en Nombre de Archivo Algunos editores como emacs, crean un archivo de copia de seguridad del archivo que se está editando. El archivo de copia de seguridad tiene el nombre del archivo original más una tilde al final del nombre del archivo. Puedes tener un archivo cuyo nombre incluya una tilde, en cualquier ubicación simplemente como:

```
$ touch '~tony.txt'
$ touch 'anusha~.txt'
```

Espacio en Blanco en el Nombre del Archivo cree un archivo con el nombre que tenga espacios entre caracteres/palabras, diga "hola, mi nombre es tony.txt". No es una buena idea tener nombres de archivos con espacios y si tiene que distinguir un nombre legible, debe usar un guión bajo o un guión. Sin embargo, si tiene que crear un archivo de este tipo, debe utilizar una barra invertida que ignore el siguiente carácter. Para crear el archivo anterior tenemos que hacerlo de esta manera:

```
$ touch hola\ mi\ nombre\ es\ tony.txt
```

He intentado cubrir todos los escenarios con los que te puedas encontrar. La mayoría de las implementaciones anteriores son explícitamente para BASH Shell y es posible que no funcionen en otros Shells.

4.9 Permisos de Archivos y Directorios

GNU/Linux, al ser un sistema diseñado fundamentalmente para trabajo en red, la seguridad de la información que almacenemos en nuestros equipos (y no se diga en los servidores) es fundamental, ya que muchos usuarios tendrán o podrán tener acceso a parte de los recursos de Software (tanto aplicaciones como información) y Hardware que están gestionados en estos equipos de cómputo. ¿Ahora podemos ver por qué la necesidad de un sistema de permisos?

En GNU/Linux, los permisos o derechos que los usuarios pueden tener sobre determinados archivos contenidos en él se establecen en tres niveles claramente diferenciados. Estos tres niveles son los siguientes:

- Permisos del propietario.

- Permisos del grupo.
- Permisos del resto de usuarios (o también llamados "los otros").

Para tener claros estos conceptos, en los sistemas en red siempre existe la figura del administrador, superusuario o root. Este administrador es el encargado de crear y dar de baja a usuarios, así como también, de establecer los privilegios que cada uno de ellos tendrá en el sistema. Estos privilegios se establecen tanto para el directorio de trabajo (Home) de cada usuario como para los directorios y archivos a los que el administrador decida que el usuario pueda acceder.

Permisos del propietario el propietario (user ID, UID) es aquel usuario que genera o crea un archivo/carpeta dentro de su directorio de trabajo, o en algún otro directorio sobre el que tenga derechos. Cada usuario tiene la potestad de crear, por defecto, los archivos que quiera dentro de su directorio de trabajo. En principio, él y solamente él será el que tenga acceso a la información contenida en los archivos y directorios que hay en su directorio trabajo o Home -bueno, no es del todo cierto esto, ya que el usuario *root* siempre tiene acceso a todos los archivos y directorios del sistema-.

Permisos del grupo lo más normal es que cada usuario pertenezca a un grupo de trabajo (group ID, GID). De esta forma, cuando se gestiona un grupo, se gestionan todos los usuarios que pertenecen a éste. Es decir, es más fácil integrar varios usuarios en un grupo al que se le conceden determinados privilegios en el sistema, que asignar los privilegios de forma independiente a cada usuario.

Permisos del resto de usuarios por último, también los privilegios de los archivos contenidos en cualquier directorio, pueden tenerlos otros usuarios que no pertenezcan al grupo de trabajo en el que está integrado el archivo en cuestión. Es decir, a los usuarios que no pertenecen al grupo de trabajo en el que está el archivo, pero que pertenecen a otros grupos de trabajo, se les denomina resto de usuarios del sistema.

¿cómo puedo identificar todo esto? sencillo, abre una terminal y realiza lo siguiente:

```
$ id
```

el comando *id* nos muestra nuestro UID, GID y los grupos a los que pertenecemos, podemos usar:

```
$ id antonio
```

para que nos muestra el UID, GID y los grupos a los que pertenece el usuario antonio.

Para conocer los permisos de nuestros archivos, podemos usar:

```
$ ls -l
```

entregará una salida como esta:

```
-rwxrwxr– 1 antonio ventas 9090 sep 9 14:10 presentacion
-rw-rw-r– 1 antonio antonio 2825990 sep 7 16:36 reporte1
drwxr-xr-x 2 antonio antonio 4096 ago 27 11:41 videos
```

Veamos por partes el listado, tomando como ejemplo la primera línea. La primera columna (-rwxrwxr–) es el tipo de archivo y sus permisos, la siguiente columna (1) es el número de enlaces al archivo, la tercera columna (antonio) representa al propietario del archivo, la cuarta columna (ventas) representa al grupo al que pertenece al archivo y las siguientes son el tamaño, la fecha y hora de última modificación y por último el nombre del archivo o directorio.

El primer carácter al extremo izquierdo, representa el tipo de archivo, los posibles valores para esta posición son los siguientes:

- \- Archivo
- d Directorio
- b Archivo de bloques especiales (Archivos especiales de dispositivo)
- c Archivo de caracteres especiales (Dispositivo tty, impresora...)
- l Archivo de vínculo o enlace (soft/symbolic link)

- p Archivo especial de cauce (pipe o tubería)

Los siguientes 9 restantes, representan los permisos del archivo y deben verse en grupos de 3 y representan:

- \- Sin permiso
- r Permiso de lectura
- w Permiso de escritura
- x Permiso de ejecución

Los tres primeros representan los permisos para el propietario del archivo. Los tres siguientes son los permisos para el grupo del archivo y los tres últimos son los permisos para el resto del mundo u otros.

Las nueve posiciones de permisos son en realidad un bit que o está encendido (mostrado con su letra correspondiente) o está apagado (mostrado con un guión -), así que, por ejemplo, permisos como rwxrw-r–, indicaría que los permisos del propietario (rwx) puede leer, escribir y ejecutar el archivo, el grupo (o sea los usuarios que estén en mismo grupo del archivo) (rw-) podrá leer y escribir pero no ejecutar el archivo, y cualquier otro usuario del sistema (r–), solo podrá leer el archivo, ya que los otros dos bits de lectura y ejecución no se encuentran encendidos o activados.

Permisos para archivos:

- Lectura: permite, fundamentalmente, visualizar el contenido del archivo.
- Escritura: permite modificar el contenido del archivo.
- Ejecución: permite ejecutar el archivo como si de un programa ejecutable se tratase.

Permisos para directorios:

- Lectura: Permite saber qué archivos y directorios contiene el directorio que tiene este permiso.
- Escritura: permite crear archivos en el directorio, bien sean archivos ordinarios o nuevos directorios. Se pueden borrar directorios, copiar archivos en el directorio, mover, cambiar el nombre, etc.

- Ejecución: permite situarse sobre el directorio para poder examinar su contenido, copiar archivos de o hacia él. Si además se dispone de los permisos de escritura y lectura, se podrán realizar todas las operaciones posibles sobre archivos y directorios.

Observación 1 *Si no se dispone del permiso de ejecución, no podremos acceder a dicho directorio (aunque utilicemos el comando "cd"), ya que esta acción será denegada. También permite delimitar el uso de un directorio como parte de una ruta (como cuando pasamos la ruta de un archivo que se encuentra en dicho directorio como referencia. Supongamos que queremos copiar el archivo "X.ogg" el cual se encuentra en la carpeta "/home/antonio/Z" -para lo cual la carpeta "Z" no tiene permiso de ejecución-, haríamos lo siguiente:*

$ cp /home/antonio/Z/X.ogg /home/antonio/Y/

obteniendo con esto un mensaje de error diciéndonos que no tenemos los permisos suficientes para acceder al archivo. Si el permiso de ejecución de un directorio está desactivado, se podrá ver su contenido (si se cuenta con permiso de lectura), pero no se podrá acceder a ninguno de los objetos contenidos en él, pues para ello este directorio es parte del camino necesario para resolver la ubicación de sus objetos.

Permisos en formato numérico octal La combinación de valores de cada grupo de los usuarios forma un número octal, el bit x es 20 es decir 1, el bit w es 21 es decir 2, el bit r es 22 es decir 4, tenemos entonces:

- r = 4
- w = 2
- x = 1

La combinación de bits encendidos o apagados en cada grupo da ocho posibles combinaciones de valores, es decir la suma de los bits encendidos:

$---$	$= 0$	no se tiene ningún permiso
$--x$	$= 1$	solo permiso de ejecución
$-w-$	$= 2$	solo permiso de escritura
$-wx$	$= 3$	permisos de escritura y ejecución
$r--$	$= 4$	solo permiso de lectura
$r-x$	$= 5$	permisos de lectura y ejecución
$rw-$	$= 6$	permisos de lectura y escritura
rwx	$= 7$	todos los permisos establecidos

Cuando se combinan los permisos del usuario, grupo y otros, se obtienen un número de tres cifras que conforman los permisos del archivo o del directorio. Esto es más fácil visualizarlo con algunos ejemplos:

Estableciendo Permisos con el Comando chmod habiendo entendido lo anterior, es ahora fácil cambiar los permisos de cualquier archivo o directorio, usando el comando chmod (change mode), cuya sintaxis es la siguiente:

chmod [opciones] permisos archivo[s], algunos ejemplos:

```
$ chmod 755 reporte1
$ chmod 511 respaldo.sh
$ chmod 700 julio*
$ chmod 644 *
```

Los ejemplos anterior establecen los permisos correspondientes que el usuario propietario desea establecer, el tercer ejemplo (chmod 700 julio*) cambiará los permisos a todos los archivos que empiecen con julio (julio01, julio02, julio_respaldo, etc.) debido al carácter '*' que es parte de las expresiones regulares que el Shell acepta, e indica lo que sea. El último ejemplo por lo tanto cambiará los permisos a los archivos dentro del directorio actual.

Una opción común cuando se desea cambiar todo un árbol de directorios, es decir, varios directorios anidados y sus archivos correspondientes, es usar la opción -R, de recursividad:

```
$ chmod -R 755 respaldos/*
```

Esto cambiará los permisos a 755 (rwxr-xr-x) del directorio respaldos y de todos los subdirectorios y archivos que estén contenidos dentro de este.

Estableciendo Permisos en Modo Simbólico Otra manera popular de establecer los permisos de un archivo o directorio es a través de identificadores del bit (r,w, o x) de los permisos, como ya se vio anteriormente, pero ahora identificando además lo siguiente:

- al usuario con la letra u
- al grupo con la letra g

- a otros usuarios con la letra o
- y cuando nos referimos a todos (usuario, grupo, otros) con la letra a (all, todos en inglés)
- el signo + para establecer el permiso
- el signo - para eliminar o quitar el permiso

La sintaxis es muy simple:

```
$ chmod augo[+|-]rwx[,...] archivo[s]
```

así por ejemplo, si queremos que otros tengan permiso de escritura sería chmod o+w archivo, todos los usuarios con permisos de ejecución chmod a+x archivo.

Si quieres prevenirte de modificar un archivo importante, simplemente quita el permiso de escritura en tu «archivo» con el comando chmod

```
$ chmod -w tuArchivo
```

si quieres hacer un Script ejecutable, escribe

```
$ chmod +x tuScript
```

si quieres remover o agregar todos los atributos a la vez

```
$ chmod -rwx archivo
$ chmod +rwx archivo
```

también puedes usar el signo = (igual) para establecer los permisos en una combinación exacta, este comando remueve los permisos de escritura y ejecución dejando solo el de lectura

```
$ chmod =r archivo
```

En este modo de establecer permisos, solo hay que tomar en cuenta que partiendo de los permisos ya establecidos se agregan o se quitan a los ya existentes

Cambiando Propietario y Grupo Volviendo a mostrar el listado al inicio de esta sección:

```
$> ls -l
-rwxrwxr– 1 antonio ventas 9090 sep 9 14:10 presentacion
-rw-rw-r– 1 antonio antonio 2825990 sep 7 16:36 reporte1
drwxr-xr-x 2 antonio antonio 4096 ago 27 11:41 videos
```

Vemos en la tercera y cuarta columna al usuario propietario del archivo y al grupo al que pertenece, es posible cambiar estos valores a través de los comandos chown (Change Owner, cambiar propietario) y chgrp (Change Group, cambiar grupo). La sintaxis es muy sencilla:

```
$ chown usuario archivo[s]
```

y

```
$ chgrp grupo archivo[s].
```

por ejemplo:

```
# ls -l presentacion
-rwxrwxr– 1 antonio ventas 9090 sep 9 14:10 presentacion

# chown juan presentacion
# ls -l presentacion
-rwxrwxr– 1 juan ventas 9090 sep 9 14:10 presentacion

# chgrp gerentes presentacion
# ls -l presentacion
-rwxrwxr– 1 juan gerentes 9090 sep 9 14:10 presentacion
```

Solo el usuario root puede cambiar usuarios y grupos a su voluntad sobre cualquier usuario, queda claro que habiendo ingresado al sistema como usuario normal, solo podrá hacer cambios de grupos, y eso solo a los que pertenezca.

Una manera rápida para el usuario *root* de cambiar usuario y grupo al mismo tiempo, es con el mismo comando chown de la siguiente manera:

```
# chown juan.gerentes presentacion
```

o en vez de punto, con : puntos

```
# chown juan:gerentes presentacion
```

así, cambiará el usuario.grupo en una sola instrucción.

Además, al igual que con chmod, también es posible utilizar la opción -R para recursividad.

Bits SUID, SGID y de Persistencia (Sticky Bit) Aún hay otro tipo de permisos que hay que considerar. Se trata del Bit de permisos *SUID* (Set User ID), el Bit de permisos *SGID* (Set Group ID) y el Bit de permisos de persistencia (Sticky Bit). Para entender los dos primeros el *SUID* y el *SGID* veamos los permisos para un comando de uso común a todos los usuarios, que es el comando passwd, que como se sabe sirve para cambiar la contraseña del usuario, y puede ser invocado por cualquier usuario para cambiar su propia contraseña, si vemos sus permisos observaremos un nuevo tipo de permiso:

```
# ls -l /usr/bin/passwd
```

```
-r-s–x–x 1 root root 21944 feb 12 2020 /usr/bin/passwd
```

SUID en vez de la 'x' en el grupo del usuario encontramos ahora una 's' (suid). passwd es un comando propiedad de root, pero sin embargo debe de poder ser ejecutado por otros usuarios, no solo por root. Es aquí donde interviene el Bit SUID, donde al activarlo obliga al archivo ejecutable binario a ejecutarse como si lo hubiera lanzado el usuario propietario y no realmente quien lo lanzó o ejecutó. Es decir, es poder invocar un comando propiedad de otro usuario (generalmente de root) como si uno fuera el propietario.

SGID el Bit SGID funciona exactamente igual que el anterior solo que aplica al grupo del archivo. Es decir si el usuario pertenece al grupo 'ventas' y existe un binario llamado 'reporte' que su grupo es 'ventas' y tiene el Bit SGID activado, entonces el usuario que pertenezca al grupo 'ventas' podrá ejecutarlo. También se muestra como una 's' en vez del Bit 'x' en los permisos del grupo.

STICKY BIT (Bit de persistencia) este Bit se aplica para directorios como en el caso de /tmp y se indica con una 't':

```
# ls -ld /tmp

drwxrwxrwt 24 root root 4096 feb 25 18:14 /tmp
```

Puede apreciarse la 't' en vez de la 'x' en los permisos de otros. Lo que hace el Bit de persistencia en directorios compartidos por varios usuarios, es que el sólo el propietario del archivo pueda eliminarlo del directorio. Es decir cualquier otro usuario va a poder leer el contenido de un archivo o ejecutarlo si fuera un binario, pero sólo el propietario original podrá eliminarlo o modificarlo. Si no se tuviera el Sticky Bit activado, entonces en estas carpetas públicas, cualquiera podría eliminar o modificar los archivos de cualquier otro usuario.

Estableciendo Permisos Especiales Para cambiar este tipo de Bit se utiliza el mismo comando *chmod* pero agregando un número octal (1 al 7) extra al principio de los permisos, ejemplo:

```
# ls -l /usr/prog

-r-x–x–x 24 root root 4096 sep 25 18:14 prog

# chmod 4511 /usr/prog
# ls -l /usr/prog

-r-s–x–x 24 root root 4096 sep 25 18:14 prog
```

Nótese que el valor extra es el '4' y los demás permisos se dejan como se quieran los permisos para el archivo. Es decir, los permisos originales en este ejemplo eran 511 (r-x–x–x), y al cambiarlos a 4511, se cambió el Bit SUID reemplazando el Bit 'x' del usuario por 's'.

Los posibles valores serían los siguientes:

- - - - - - - - - = 0 Predeterminado, sin permisos especiales. No se requiere indicar.
- - - - - - - - t = 1 Bit de persistencia, Sticky Bit
- - - - - s - - - = 2 Bit Sgid de grupo
- - - - - s - - t = 3 Bit Sgid y Sticky

- - s - - - - - -	= 4	Bit Suid
- - s - - - - - t	= 5	Bit Suid y Sticky
- - s - - s - - -	= 6	Bit Suid y Sgid
- - s - - s - - t	= 7	Bit Suid, Sgid y Sticky

Observación 2 *Algo sumamente delicado y que se tiene que tomar muy en cuenta es lo que decidas establecer con permisos de Bit SUID y SGID, ya que recuerda que al establecerlos de esta manera, cualquier usuario podrá ejecutarlos como si fueran el propietario original de ese programa. Y esto puede tener consecuencias de seguridad severas en tu sistema. Siempre considera y reconsidera si conviene que un usuario normal ejecute aplicaciones propias de root a través del cambio de Bits SUID o SGID. Mejores alternativas pueden ser los comandos sudo y su.*

Permisos preestablecidos con umask El comando umask establece un permiso de archivo y directorio para todos los archivos y directorios que se creen, para conocer su valor podemos teclear:

```
$ umask
```

y nos regresa su valor, por ejemplo: 0022

En formato simbólico usamos la opción -S:

```
$ umask -S
```

y nos regresa su valor, por ejemplo: u=rwx,g=rx,o=rx

lo anterior indica que un directorio se creará con los permisos 755 y los archivos con los permisos 644. Esto se logra restando de 777 el valor de umask (777-022) y (666-022) respectivamente. El primer valor de umask corresponde para valores de Sticky Bit, *GUID* o *SUID*, que por omisión es 0.

Para establecer el valor de la máscara, simplemente se usa el mismo comando *umask* seguido del valor de máscara que se desee:

```
$ umask 0002
```

para dejarlo fijo en la sesión, entonces conviene agregarlo a *.bash_rc* o *.bash_profile* de nuestro directorio de inicio. Algunos ejemplos son:

umask	*Permisos*	*en Archivos*	*Permisos*	*en Directorios*
000	666	(rw-rw-rw-)	777	(rwxrwxrwx)
002	664	(rw-rw-r- -)	775	(rwxrwxr-x)
022	644	(rw-r- -r- -)	755	(rwxr-xr-x)
027	640	(rw-r- - - - -)	750	(rwxr-x- - -)
077	600	(rw- - - - - - -)	700	(rwx- - - - - -)
277	400	(r- - - - - - - -)	500	(r-x- - - - - -)

Archivos y Fechas En sistemas similares a UNIX como GNU/Linux, todo se considera un archivo, y toda la información sobre un archivo (metadatos o atributos del archivo como la hora de creación, la última modificación, etc) excepto en contenido del archivo real se almacena en un inodo y Linux identifica todos y cada uno de los archivos por su número de inodo que no sea el nombre de archivo legible por humanos.

Además, el programa *stat* de GNU/Linux es una utilidad útil para mostrar archivos o el estado del sistema de archivos. Muestra información como el número de inodo, la hora de creación del archivo, la última modificación de datos, el último acceso, el último cambio de estado y mucho más. Podemos combinar *stat* y *debugfs* para obtener toda la información de un archivo, mediante:

```
$ stat archivo
```

no regresa el inodo del archivo en cuestión (supongamos 12), y usando *df* encontramos la partición en la que reside el archivo (supongamos /dev/sda1), combinando esto entonces:

```
# debugfs -R 'stat <12>' /dev/sda2
```

nos mostrará toda la información de dicho archivo.

4.10 Procesos en Primer y Segundo Plano

La ejecución de una orden se efectúa con una o más llamadas al sistema operativo. Por lo regular, el Shell ejecuta una pausa cuando se le pide ejecutar una orden, queda en espera a que ésta termine se de ejecutar. Existe una sintaxis sencilla (un signo & al final de la línea de órdenes) para indicar que el Shell no debe esperar hasta que termine de ejecutarse la orden. Una orden

que deja de ejecutándose de esta manera mientras el Shell sigue interpretando órdenes subsecuentes es una orden en segundo plano (Background), o que se ejecuta en segundo plano. Los procesos para los cuales el Shell sí espera se ejecutan en primer plano (Foreground).

El Shell del sistema GNU/Linux ofrece un recurso llamado control de trabajos (y visualizarlos con los comandos *ps* o *top*) implementado especialmente en el núcleo. El control de trabajos permite transferir procesos entre el primer y segundo plano. Los procesos pueden detenerse y reiniciarse según diversas condiciones, como que un trabajo en segundo plano requiera entradas desde la terminal del usuario. Este esquema hace posible la mayor parte del control de procesos que proporcionan las interfaces de ventanas, pero no requiere Hardware especial. Cada ventana se trata como una terminal, y permite a múltiples procesos estar en primer plano (uno por ventana) en cualquier momento. Desde luego pueden haber procesos de segundo plano en cualquiera de las ventanas.

En GNU/Linux podemos ejecutar procesos en primer plano (Foreground) o bien en segundo plano (Background). Un programa en primer plano lanzado desde un terminal monopoliza dicho terminal, por lo que en principio, no podremos ejecutar ningún otro programa a la vez (veremos más adelante como se puede hacer). Por el contrario un programa en segundo plano una vez iniciado, deja de monopolizar el terminal desde el que se lanzó, y este nos vuelve a mostrar el Prompt[78].

El comando *bg* que es la abreviatura de la palabra background por tanto recibe como parámetro un número de proceso en estado *Stopped* y hace que ésta reanude su ejecución pero en segundo plano es decir, sin bloquear el teclado para el Shell.

De igual forma que hemos pasado un proceso a ejecutar a un segundo plano, el Shell nos permite recuperar la ejecución en primer plano de un proceso que estaba en segundo plano. Para ello se utiliza el comando *fg* (abreviatura de la palabra Foreground). Como parámetro recibe al igual que *bg* el número de un proceso prefijado con %.

¿Como podemos lanzar otro programa desde un terminal con otro pro-

[78] En el Bourne Shell y sus derivados como BASH el *Prompt* que nos permite escribir los diferentes comandos, generalmente termina con el carácter:

- $ para usuario sin privilegios
- # para el administrador, conocido como *root*

grama en ejecución en Foreground?

Pulsamos *CTRL-z* con lo que pausamos el programa en ejecución y Foreground, ojo lo pausamos con lo cual dejará de funcionar, y ya podremos lanzar otro programa, por ejemplo *ls* o podemos hacer una prueba lanzamos *gimp* y comprobamos que podemos operar con él, luego pulsamos *CTRL-z* y vemos cómo dejamos de poder trabajar con *gimp*).

Ahora queremos volver a poner en funcionamiento a *gimp* y así poder volver a utilizar gimp o si queremos devolverlo a Foreground escribiremos *fg*, o si queremos devolverlo a background escribiremos *bg* (esta sería la opción más lógica).

En el caso de que tengamos más de un programa detenido deberemos indicarle tanto a *fg* como a *bg* el identificador de tarea sobre el que actuarán, este ID podemos obtenerlo con el comando *jobs*.

Tenemos a la mano muchas herramientas en el Shell que nos permiten trabajar con los procesos, saber sus PID, sus estamos, sus nombre, sus usuarios, la cantidad de recursos que consumen. Podemos mencionar algunas:

- ps
- pstree
- pmap
- top
- kill
- jobs
- disown

Para interrumpir un proceso que se está ejecutando en segundo plano podemos hacerlo de dos formas. Mediante el comando *fg* la pasamos a ejecutar al primer plano y luego con *Ctrl-c*. Pero también podemos conseguir el mismo efecto sin utilizar el comando *fg* sino simplemente con el comando *kill* seguido del número de proceso prefijado con el símbolo % que nos da el comando *jobs* y podemos usar *disown* para que el comando continúe activo después de cerrar la terminal.

Nohup y & Cuando se cierra una sesión en GNU/Linux, el sistema manda una señal para matar todos los procesos que esté lanzando nuestro usuario (Señal *SIGHUP*). Por lo tanto, aunque les pongamos el & al final, morirán. Para evitar que esto suceda, hay que ejecutar el Script o comando poniendo el comando *nohup* delante y lo que va a hacer es ignorar la señal de *SIGHUP* y redirigir toda la información correspondiente a un fichero llamado nohup.out

```
$ nohup ./Script.sh &
```

esta opción es muy útil cuando no hemos de interactuar con el proceso, puesto que al quedar el fichero de log, se puede ver el resultado con toda la información al respecto.

Los Scripts son indispensable en Linux y en algunas ocasiones puede ser necesario ejecutarlo en segundo plano, ya sea porque tarde mucho en finalizar o porque el programa tiene que ejecutarse de forma indefinida y al mismo tiempo se quieren analizar sus entradas/salidas en tiempo real, o cuando, en el caso de conexiones remotas, por el motivo que sea, se pueda producir una desconexión.

Una buena práctica sería redireccionar stdin, stdout y stderr. Básicamente, por dos razones: rastrear la salida de nuestro Script en caso de producirse algún error, y evitar problemas al terminar nuestra sesión *ssh*, si es que la ejecutamos en un servidor remoto, por ejemplo:

```
$ nohup ./Script.sh > foo.out 2> foo.err < /dev/null &
```

esta opción es muy útil cuando necesitamos ejecutar un proceso largo y no nos interesa saber nada de él hasta que finalice. Podemos desconectarnos irnos, regresar al día siguiente y analizar el resultado del proceso, como alternativa podemos también usar:

```
$ nohup ./Script.sh > salida.txt 2>&1 &
```

Limitar el Tiempo de Ejecución Algunas veces es necesario limitar el tiempo máximo de ejecución de un comando o Script, para ello disponemos de *timeout*, su formato es el siguiente:

```
$ timeout [opciones] duración comando [argumentos]
```

la duración la podemos especificar en segundos (s), minutos (m), horas (h) o días (d), ejemplos:

```
$ timeout 8s ping 127.0.0.1
$ timeout 2h comando argumentos
```

Podemos darle un período de gracia antes de forzar su terminación, ejemplo:

```
$ timeout -k=5 2m comando argumentos
$ timeout -k=5 -s SIGKILL 2m comando argumentos
```

en este caso se envía una señal *KILL* si el comando todavía se está ejecutando tanto tiempo después de que se envió la señal inicial.

También podemos pedir que corra en segundo plano el comando, mediante:

```
$ timeout –foreground 30s ssh -t usuario@servidor top
```

Limitar uso de CPU a un programa Algunas veces es necesario limitar el uso de la CPU (expresado en porcentaje de uso) a un proceso o programa para ello usamos a *cpulimit*, para instalarlo usamos:

```
# apt install cpulimit
```

Si queremos limitar el uso del CPU de un programa en ejecución usamos:

```
# cpulimit -l 50 -e apache2
```

Si conocemos el PID de nuestro proceso, podemos usarlo para limitar el uso del CPU (digamos al 30%), usando:

```
$ cpulimit -l 30 -p 4546 &
```

o bien, podemos lanzar el proceso limitando el uso del CPU, mediante:

```
$ cpulimit -l 30 ./miprograma &
```

4.11 GNU Parallel

¿Alguna vez tuviste la extraña sensación de que tu computadora no es tan rápida como debería ser? Solía sentirme así, y luego encontré GNU Parallel. GNU Parallel es una herramienta de Shell que permite la ejecución de trabajos en paralelo.

Un trabajo puede ser un único comando o entrada de un archivo que contiene elementos tales como una lista de comandos, una lista de archivos, una lista de Hosts, una lista de usuarios, una lista de URL o una lista de tablas. GNU Parallel también puede tomar información de un comando con pipes (|).

Si te encuentras en una situación en la que necesitas ejecutar varios comandos al mismo tiempo (por ejemplo, en los servidores Linux de un centro de datos), ¿qué haces? GNU Parallel es una alternativa interesante que debes conocer.

Cuando se ejecutan comandos en Linux, ya sea uno a la vez en la línea de comandos o desde un Script de Bash, los comandos se ejecutan en secuencia. El primer comando se ejecuta, seguido por el segundo, seguido por el tercero. Es cierto, el tiempo entre los comandos es tan minúsculo, que el ojo humano no se daría cuenta. Pero para algunos casos, puede que no sea el medio más eficiente para ejecutar comandos.

GNU Parallel se puede instalar en casi cualquier distribución de Linux. Dado que GNU Parallel se encuentra en el repositorio estándar, se instala mediante:

```
# apt install parallel
```

Por ejemplo, para cambiar el formato de los .jpg a .png podríamos hacer lo siguiente:

```
$ find .  -name "*.jpg" | parallel -I% –max-args 1 convert %
%.png
```

o

```
$ ls -l *.jpg | parallel convert '{}' '{.}.png'
```

Con eso conseguimos que los comandos *find* o *ls* busquen todos los ficheros .jpg en el directorio actual con cualquier nombre y le pase todos los resultados

a *parallel* mediante la pipe, que luego transmitirá de uno a uno al comando convert para convertirlos a png. Es decir, va a realizar convert nombre1.jpg nombre1.png, convert nombre2.jpg nombre2.png, y así sucesivamente...

O directamente usamos *parallel* para ello:

```
$ parallel convert '{}' '{.}.png' ::: *.jpg
```

ahora, utilizaremos zenity para presentar una barra de progreso de manera gráfica. ¡Porque, que sea terminal no significa que no sea visual!

```
$ find -maxdepth 1 -name '*.jpg' | parallel –bar convert {}
convertidas/{/.}.png 2> >(zenity –progress –auto-kill)
```

5 Trabajando en Línea de Comandos

La mayoría de los usuarios de ordenadores de hoy sólo están familiarizados con la interfaz gráfica de usuario o GUI (del inglés Graphical User Interface) y los vendedores y los expertos les han enseñado que la interfaz de línea de comandos o CLI (del inglés Command Line Interface) es una cosa espantosa del pasado. Es una pena, porque una buena interfaz de línea de comandos es una maravillosa y expresiva forma de comunicarse con el ordenador, muy parecida a lo que el lenguaje escrito es para los seres humanos. Se ha dicho que "las interfaces gráficas de usuario hacen fáciles las tareas fáciles, mientras que las interfaces de línea de comandos hacen posibles las tareas difíciles" y eso es muy cierto aún hoy.

Dado que Linux fue desarrollado desde la familia de sistemas operativos Unix, comparte la misma rica herencia de herramientas de línea de comandos que Unix. Unix saltó a la fama en los primeros años ochenta -aunque fue desarrollado una década antes-, antes de que se extendiera la adopción de las interfaces gráficas de usuario, y por eso, se desarrolló una amplia interfaz de línea de comandos en su lugar. De hecho, una de las razones más potentes para que los primeros que utilizaron Linux lo eligieran sobre, digamos, Windows NT, era la poderosa interfaz de línea de comandos que hacía las "tareas difíciles posibles".

Emuladores de Terminal cuando utilizamos una interfaz gráfica de usuario, necesitamos otro programa llamado emulador de terminal para interactuar con el Shell. Si buscamos en nuestros menús de escritorio, probablemente encontraremos uno. KDE usa Konsole y GNOME usa Gnome-terminal, aunque es probable que se llame simplemente "Terminal" en nuestro menú. Hay muchos otros emuladores de terminal disponibles para Linux, pero todos hacen básicamente lo mismo; nos dan acceso al Shell.

Trabajar con la línea de comandos no es una tarea tan desalentadora como muchos pudieran pensar. No se requieren conocimientos especiales para usar la línea de comandos, pues es un programa como otro cualquiera. La mayoría de las acciones realizadas en Linux pueden llevarse a cabo usando la línea de comandos. Aunque existen herramientas gráficas para la mayoría de programas, a veces esto no es suficiente. Entonces es cuando la línea de comandos cobra su utilidad.

La terminal es llamada a menudo la línea de comandos, el "Prompt", o el "Shell". Antes, éste era el único método por el cual el usuario interactuaba con lel ordenador; sin embargo, muchos usuarios de Linux encuentran más rápido el uso de la terminal que un método gráfico e incluso hoy día tiene algunas ventajas. Aquí aprenderemos cómo usar la terminal.

El Intérprete de Órdenes de Consola o Shell es un programa informático, cuya función consiste en interpretar órdenes, y un lenguaje de consola. El más conocido es Bash[79]. Es una Shell de Unix compatible con POSIX y el intérprete de comandos por defecto en la mayoría de las distribuciones GNU/Linux, además de Mac OS. También se ha llevado a otros sistemas como Windows y Android.

Algunas alternativas a Bash son:

- SH (Bourne Shell) fue desarrollado por Stephen Bourne y es un Shell que se encuentra dentro de la jerarquía de archivos de Unix en /bin/sh.
- TCSH/CSH (C Shell) ha sido desarrollado para proporcionar una interfaz de usuario. Gracias a este Shell podremos ejecutar comandos y ejecutar múltiples programas desde la consola del sistema.
- KSH (Korn Shell) su desarrollo principal fue la interpretación de órdenes a través de línea de comandos.
- Fish (Friendly Interactive Shell) no está basado en sh y posee una sintaxis de línea de comandos única que está diseñada para ser más amigable con los usuarios que están iniciando en el mundo Shell.
- ZSH (Z Shell) ha sido un Shell diseñado en 1990 influenciado por Bash, Ksh y Tcsh. Zsh es un Shell popular gracias a sus características de desempeño y funcionalidades a la hora de ejecutar comandos.
- TSCH (TS Shell) es una versión mejorada de CSH , la cual ofrece múltiples usos ya que es un lenguaje de comandos que puede ser usado tanto como un Shell de inicio de sesión interactivo como un procesador de comandos Shell.

[79] Su nombre es un acrónimo de Bourne-again Shell ("Shell Bourne otra vez"), haciendo un juego de palabras (Born-again significa "nacido de nuevo") sobre la Bourne Shell (sh), que fue uno de los primeros intérpretes importantes de Unix.

- DaSH (Debian Almquist Shell) es un intérprete de comandos de Unix compatible con el estándar sh de POSIX, mucho más ligero y rápido que otros como Bash pero con menos características.
- Busybox, integra más de doscientas utilidades en un solo ejecutable, usa Almquist Shell y es ideal para dispositivos Linux integrados.
- DSH (Distributed Shell) es un pequeño programa que nos permitirá ejecutar un comando en varias máquinas de una sola vez.

Podemos conocer que interprete de comandos usamos, para ello escribimos:

```
$ ps -p $$
```

Si deseamos cambiar de intérprete a por ejemplo *zsh*, podemos instalarlo usando:

```
# apt install zsh
```

y debemos indicarle al sistema que queremos utilizar *zsh* como Shell por defecto usando:

```
chsh -s $(which zsh)
```

La sintaxis de órdenes de Bash es un superconjunto de instrucciones basadas en la sintaxis del intérprete Bourne. La especificación definitiva de la sintaxis de órdenes de Bash, puede encontrarse en el Bash Reference Manual distribuido por el proyecto GNU. Esta sección destaca algunas de sus características únicas.

5.1 Buscar Archivos

La búsqueda de archivos de GNU/Linux se puede hacer con por lo menos dos comandos diferentes para buscar archivos: find (que traduce encontrar) y locate (que traduce ubicar).

Usando el comando find El comando más común utilizado para encontrar y filtrar archivos es a través del comando *find*. El diseño básico de este comando es el siguiente:

find <directorio inicio> <opciones> <termino búsqueda>

El argumento <*directorio inicio*> es el punto de origen de donde deseas iniciar la búsqueda. Esto es útil si tienes una idea aproximada de dónde podría estar ubicado el archivo deseado, ya que hace más específica la búsqueda. La mayoría de las veces, sin embargo, querrás buscar el archivo en todo el sistema. Puedes hacer esto reemplazando tu ruta con una barra *" / "*, que es el símbolo del directorio raíz. A veces es posible que quieras iniciar la búsqueda desde el directorio de trabajo actual, es decir, el directorio donde está abierto el terminal. Esto se puede hacer con el argumento punto *" . "*. Para averiguar tu directorio actual, usa el comando pwd. Finalmente, para comenzar la búsqueda de archivos desde tu carpeta de inicio, usa el símbolo *" ~"*.

El segundo argumento es el filtro que deseas usar para buscar tu archivo. Este podría ser el nombre, tipo, fecha de creación o de modificación del archivo, etc. El tercer argumento es una continuación del segundo, donde se especificará el término de búsqueda relevante.

Búsqueda por Nombre por supuesto, el método más común y obvio para buscar un archivo es usar su nombre. Para ejecutar una consulta de búsqueda simple usando el nombre del archivo, usa el comando find de la siguiente manera:

```
$ find . -name "archivo"
```

En el comando anterior, usamos la opción *-name* y buscamos un archivo llamado *archivo*. Ten presente que comenzamos la búsqueda en nuestro directorio actual y nos dará todas las coincidencias que encuentre. Pero si sólo deseamos la primera coincidencia podemos usar:

```
$ find . -name "archivo" -print -quit
```

Una cosa importante para recordar cuando se utiliza el estándar es el argumento *-name*, que busca términos distinguiendo entre mayúsculas y minúsculas en GNU/Linux. Por lo tanto, si conoces el nombre del archivo, pero no estás seguro de su las mayúsculas y minúsculas, usa el comando find de esta manera:

```
$ find . -iname "archivo"
```

o bien para buscar todos los archivos con extensión .zip, .tar, .tar.gz, entre otros, usamos:

```
$ find . -type f \( -name "*.zip" -o -name "*.tar*" \)
```

Otra forma de utilizar la opción name es buscar todos los archivos sin una palabra clave en el nombre. En GNU/Linux, puedes hacer esto de dos maneras. El primer método implica el uso de la palabra clave *-not* de la siguiente manera:

```
$ find . -not -name "archivo"
```

También podemos usar *" ! "*[80], aunque debe estar precedido por el identificador de escape para que GNU/Linux sepa que esta es parte del comando de búsqueda y no una independiente.

```
$ find . \! -name "archivo"
```

También puede buscar varios archivos con un formato común como *.txt*, lo cual podría ser útil en algunos casos:

```
$ find . -name "*.txt"
```

esto listará todos los archivos de texto comenzando con la carpeta actual.

Algunas veces es necesario acotar la profundidad de la búsqueda y limitarla, digamos al actual directorio, por ejemplo:

```
$ find . -maxdepth 1 archivo
```

Si deseas buscar un determinado archivo por nombre y eliminarlo, usamos el argumento *-delete* después del nombre del archivo:

```
find . -name "archivo" -delete
```

otra forma es usar:

[80] Otras opciones son -not (!), -or (-o), y -and (-a) aunque este último está implícito cuando se enumeran dos requisitos.

```
$ find . -name "archivo" | xargs rm -f
```

también podemos tomar la salida del comando y hacer algo con ella, como:

```
$ find . -name 'log*' | xargs wc
```

Si al realizar una búsqueda se generan errores (por no tener acceso a archivos o directorios por ejemplo), estos pueden ser redireccionados, mediante:

```
$ find / -type f 2> /dev/null
```

Búsqueda por Tipo para la mayoría de los usuarios, basta con saber cómo encontrar archivos por sus nombres. Sin embargo, siempre es útil conocer todas las herramientas que se ofrecen para aprovechar GNU/Linux al máximo.

Aquí es donde entra en juego el argumento -type. GNU/Linux ofrece a los usuarios las siguientes opciones para buscar archivos por tipo:

- f - Archivo normal
- d - Directorio o carpeta
- l - Enlace simbólico
- c - Dispositivos de caracteres
- b - Dispositivos de bloque

Un ejemplo simple del uso del tipo de archivo para la búsqueda se puede ver a continuación:

```
$ find / -type d
```

Esto mostrará una lista de todos los directorios presentes en el sistema de archivos, al haber comenzado la búsqueda desde nuestro directorio raíz con el símbolo de barra inclinada. Para encontrar las ligas simbólicas que hay desde nuestra actual trayectoria, usamos:

```
$ find . -type l
```

También puedes concatenar las opciones -type y -name para hacer tus búsquedas aún más específicas:

```
$ find / -type f -name "archivo"
```

esto buscará archivos llamados *archivo*, excluyendo directorios o enlaces.
Otro ejemplo es buscar archivos, directorios y enlaces simbólicos, mediante:

```
$ find /tmp -type f,d,l
```

o

```
$ find /tmp \( -type f -o -type d -o -type l \)
```

Para busca todas las ligas simbólicas y saber a donde apuntan, usamos:

```
$ find . -type l -print | xargs ls -ld | awk '{print $9 $10 $11}'
```

y si queremos conocer las ligas rotas, usamos:

```
$ find . -xtype l
```

Algunas veces es necesario acotar la profundidad de la búsqueda y limitarla, digamos al actual directorio, por ejemplo:

```
$ find . -maxdepth 1 -type f -newer archivo
```

Búsqueda por Fecha si quieres buscar archivos en función de su fecha de acceso y las registros de fecha de modificación, GNU/Linux te ofrece las herramientas para hacerlo. Hay 3 registros de tiempo de los cuales Linux realiza seguimiento en los archivos:

- Tiempo de acceso (-atime) - Fecha más reciente en que el archivo fue leído o escrito.
- Tiempo de modificación (-mtime) - Fecha más reciente en que se modificó el archivo.
- Hora de cambio (-ctime) - Fecha más reciente en que se actualizaron los metadatos del archivo.

Esta opción debe usarse con un número. Este número especifica el número de días desde que se accedió, modificó o cambió el archivo. La forma más sencilla de buscar archivos por tiempo es:

```
$ find / -atime 1
```

Esto encontrará todos los archivos a los que se accedió hace un día desde el momento actual. Esto significa que se listarán todos los archivos que fueron leídos y/o escritos desde el día anterior.

Podemos hacer que nuestras consultas sean más precisas con los signos más (+) y menos (-) precediendo al número de días. Por ejemplo:

```
$ find / -mtime +2
```

Esto listará todos los archivos que tienen un tiempo de modificación de más de dos días.

Para buscar todos los archivos cuyos metadatos de *inodo* se actualizaron hace menos de un día, ejecuta lo siguiente:

```
$find / -ctime -1
```

Finalmente, hay algunos argumentos adicionales además de estos 3 que también están relacionados con las búsquedas por fecha. El argumento -*<time-deScriptor>min* busca cuántos minutos han pasado. Se puede usar así:

```
$find / -mmin -1
```

Esto buscará archivos que se modificaron hace menos de un minuto. Además, el argumento *-newer* se puede usar para comparar la antigüedad de dos archivos y encontrar el más reciente.

Para buscar archivos **.log* de más de 90 días, podemos usar:

```
$ find / -name "*.log" -type f -mtime +90 -ls
```

y los podemos borrar usando:

```
$ find /app/logs/ -name "*.log" -type f -mtime +90 -delete
```

Podemos buscar archivos no vacíos de más de 4 días en el directorio actual sin extensión *.gz* y comprimirlos:

```
$ find . -mtime +4 ! -name '*.gz' ! -empty -execdir gzip -v9
{} +
```

Búsqueda por tamaño al igual que GNU/Linux te brinda la opción de buscar archivos basados en registros de tiempo, también te permite hacer lo mismo con los tamaños. La sintaxis básica para buscar archivos por tamaño es:

$ find <directorio inicio > -size <tamaño> <unidad tamaño>

Puedes especificar las siguientes unidades de tamaño:

- c - Bytes
- k - kiloBytes
- M - MegaBytes
- G - GigaBytes
- b - Trozos de 512 Bytes

Un ejemplo simple de cómo usar el comando find para los tamaños de archivo es el siguiente:

$ find / -size 10M

Esto buscará en tu sistema archivos que tengan exactamente 10 MegaBytes de tamaño. Al igual que cuando buscaste en función del tiempo, puedes filtrar aún más tus búsquedas con los signos más y menos:

$ find / -size +5G

El comando anterior listará todos los archivos de tu disco que tengan más de 5 GigaBytes de tamaño. De manera similar, puede lograr esto con el signo menos para indicar "menor que" en tus consultas.

Además podemos pedir que la búsqueda nos entregue los datos de los archivos encontrados usando el comando *ls -l*, por ejemplo:

$ find . -size +1000c -exec ls -l {} \;

o si queremos buscar y borrar, usamos:

$ find / -type f -mtime +30 -sise +1M -exec rm -rf {} \;

o bien:

```
$ find / type f -mtime +30 -size +1M -delete
```

Por último, podemos buscar ficheros vacíos, mediante:

```
$ find . -type f -empty
$ find . -size 0c
```

o directorios vacíos, usamos:

```
$ find . -type d -empty
```

Algunos ejemplos útiles son:

```
$ find . -printf '%s %p\n' | sort -nr | head -10
$ find . -type d -printf '%s %p\n' | sort -nr | head -10
$ find . -type f -printf '%s %p\n' | sort -nr | head -10
$ find .  -iname "*.mp4" -printf '%s %p\n' | sort -nr | head
-10
$ find . -type d -empty -print0 | xargs -0 rmdir
$ find . -type f -empty -print0 | xargs -0 rm
```

Búsqueda por Propiedad La jerarquía de privilegios de Linux también se puede utilizar al buscar archivos. GNU/Linux te da la capacidad de especificar tus búsquedas según la propiedad del archivo, así como los permisos otorgados a diferentes usuarios.

Para buscar archivos de un determinado propietario, se debe ejecutar el siguiente comando:

```
$ find / -user usr
```

Esto devolverá una lista de todos los archivos que posee el usuario llamado *usr*. Similar a los nombres de usuario, también podemos buscar archivos a través de nombres de grupo:

```
$ find / -group otro
```

Esto buscará aquellos archivos que son propiedad del grupo llamado *otro*.

Búsqueda por permisos Los usuarios que desean buscar archivos basados en los permisos de los archivos[81] pueden hacerlo usando la opción *-perm* del comando *find*. Por ejemplo:

```
$ find / -perm 644
```

En GNU/Linux, 644 corresponde a permisos de lectura y escritura. Lo que significa que este comando buscará todos los archivos que solo tienen permisos de lectura y escritura. Otras opciones son:

```
$ find . -perm 1551
$ find . -perm /u=r
$ find . -perm /a=x
```

o si deseamos buscar los que no tienen esos permisos, para ello usamos *-not*, por ejemplo:

```
$ find . -not -perm 777
```

También podemos buscar ficheros/directorios que tengan activos los bits *SUID*, *SGID* y *Sticky Bit* que también podrían representar problemas de seguridad. Pero para que busque cualquiera sin importar el resto de los permisos, debemos usar / para obviar los permisos estándar, para ello usamos respectivamente:

```
$ find . -perm /4000
$ find . -perm /2000
$ find . -perm /1000
```

[81]

Octal	Binario	Modo Archivo
0	000	- - -
1	001	- - x
2	010	- w -
3	011	- w x
4	100	r - -
5	101	r w -
6	110	r w -
7	111	r w x

El siguiente ejemplo busca archivos que sean de lectura para todos (-perm -444 o -perm -a+r) y que tenga al menos un conjunto de bits de escritura (-perm /222 o -perm /a+w) pero no son ejecutables para cualquiera (\! -perm /111 o \! -perm /a+x):

```
$ find . -perm -444 -perm /222 \! -perm /111
```

o

```
$ find . -perm -a+r -perm /a+w \! -perm /a+x
```

Por último, podemos cambiar todos los permisos digamos 744 encontrados a 644, usando:

```
$ find . -perm -744 -exec chmod -R 644 {} \;
```

Búsqueda por inodo el comando *ln* permite crear enlaces a los archivos, tanto duros (Hard Links) como simbólicos *-s* (Soft Links). Todos los archivos que apuntan a un mismo enlace duro, comparten el *inodo* -este es un registro en el disco que contiene la información del archivo como su propietario, tamaño, fecha de creación y ubicación-.

Para conocer el *inodo* de un archivo, usamos:

```
$ ls -li archivo
```

y para conocer todos los archivos que apuntan a un mismo enlace duro cuyo *inodo* conozcamos, usamos:.

```
$ find / -inum <inodo>
```

y para conocer a todos los archivos que comparten inodo, usamos:

```
$ find . -type f -printf '%10i %p\n' | sort | uniq -w 11 -d -D |
less
```

Búsqueda usando expresiones regulares esto nos ofrece una potencia considerable. Por ejemplo si deseamos buscar un nombre como:

archivo01_01.txt, archivo02_03.txt, etc.

podemos utilizar:

```
$ find . -regex './archivo0[1-2]_0[1-3].*'
```

Otras opciones útiles Además de todos estos métodos de búsqueda de archivos, hay otras opciones útiles que uno debería recordar. Por ejemplo, para buscar archivos y carpetas vacíos en el sistema, usamos:

```
$ find / -empty
```

del mismo modo, para buscar todos los ejecutables guardados en tu disco, utiliza la opción *-exec:*

```
$ find / -exec
```

para buscar archivos leíbles, usamos:

```
$ find / -read
```

Hay veces que necesitamos quitar espacios en el nombre de los archivo, por ejemplo:

```
$ find . -name "* *mp3" -exec rename 's/\ /-/g' {} \;
```

remplazará en todos los archivos **.mp3* los espacios en el nombre.
Si necesitamos buscar en todos nuestros archivos *.java* aquellos que contengan por ejemplo la palabra *interface*, podemos usar:

```
$ find . -name "*.java" - type f -exec grep -l interface {} \;
```

Si necesitamos buscar archivos y filtrarlos por nombre de archivo, además de canalizar el resultado a algún otro comando (por ejemplo *wc* para contar palabras y usamos *tr* para sustituir el carácter nulo por nuevas líneas), el comando queda como:

```
$ find . -name ".md" | grep "notas/2020" | \
tr '\n' '\0' | xargs -0 wc -w
```

El siguiente ejemplo copia el contenido de la actual trayectoria a /dest-dir, pero omite archivos y directorios llamados .snapshot (y cualquier cosa dentro de ellos). También omite archivos o directorios cuyo nombre termina en ~, pero no sus contenidos.

```
    $ find . -name .snapshot -prune -o \( \! -name '*~' -print0 \)
\
    | cpio -pmd0 /dest-dir
```

la construcción -prune -o \(\! -name '*~' -print0 \)| es bastante común. La idea aquí es que la expresión antes de -prune coincide con las cosas que se deben podar, sin embargo, el -prune acción en si misma se devuelve verdadero, por lo que el siguiente -o asegura que el lado derecho es evaluado solo para aquellos directorios que no fueron podados. La expresión en el lado derecho de -o está entre paréntesis solo por claridad. Hace hincapié en que la acción -print0 tiene lugar solo para las cosas que no tenían -prune aplicado a ellos. Otro ejemplo es:

```
$ find repo/ \( -exec test -d '{}'/.svn \; -or \
-exec test -d {}/.git \; -or -exec test -d {}/CVS \; \) \
-print -prune
```

en este ejemplo -prune evita el descenso innecesario a los directorios que ya han sido descubiertos.

Los siguientes ejemplos permiten copiar los archivos (por ejemplo *.mp3) pero preservan la estructura de directorios que existía en su lugar de origen, veamos:

```
$ find . -name '*.mp3' -exec cp –parents \{\} ~/OtroDirectorio \;
$ find . -name '*.mp3' | cpio -pdm ~/OtrDirectorio
$ find ~/miDirectorio -name '*.mp3' -exec cp –parents \{\} ~/OtrDirectorio \;
$ rsync -a -m –include '*/' –include '*.mp3' –exclude '*' ~/miDirectorio/ ~/OtrDirectorio
$ rsync -a –prune-empty-dirs –include '*/' –include '*.mp3' –exclude '*' ~/miDirectorio/ ~/OtrDirectorio
```

el primer y segundo ejemplo busca a partir de la actual trayectoria y los copia en *~/OtroDirectorio*, en los demás ejemplos, inicia la búsqueda en *~/midirectorio/* y los copia en *~/OtroDirectorio*.

Para eliminar múltiples archivos del tipo que buscamos, tenemos dos opciones:

```
$ find . -name "archivo" -exec rm -rf {} \;
```

o

```
$ find . -type f -name "archivo" -exec rm -rf {} \;
```

la diferencia entre ambos comandos indicados arriba es que el primero localiza y borra archivos y directorios, y el segundo sólo archivos:

Usando el comando Locate En este punto, podrías cuestionar la necesidad de tener una alternativa para el comando *find*. Después de todo, hace todo lo necesario para buscar archivos. Sin embargo, locate es una alternativa útil, ya que es más rápido que find para buscar en todo el sistema.

Por defecto, GNU/Linux no viene con el comando locate preinstalado. Para obtener el paquete, ejecuta los siguientes comandos en tu terminal:

```
# apt install mlocate
```

Ahora que has adquirido locate, puedes usarlo para buscar archivos así:

```
$ locate archivo
```

Puedes usar el argumento *-b* para una búsqueda más específica. Solo buscará el "nombre base" del archivo, enumerando efectivamente solo los archivos que tienen el término de búsqueda en lugar de devolver los directorios que conducen a los archivos.

```
$ locate -b archivo
```

Otros argumentos disponibles incluyen:

-e muestra entradas de archivos existentes en el momento en que se ejecuta el comando locate.

-q inhabilita la visualización de errores encontrados en el proceso de búsqueda.

-c muestra la cantidad de archivos que coinciden, en lugar de los nombres de los archivos

El comando locate estándar a veces puede mostrar archivos que han sido eliminados. Esto se debe a que el comando locate busca en la base de datos principal del sistema operativo GNU/Linux. Si esa base de datos no se actualiza, incluso los archivos eliminados pueden aparecer en los resultados de búsqueda. Puedes actualizar manualmente la base de datos ejecutando lo siguiente:

```
# updatedb
```

5.2 Empaquetadores, Compresores y Descompresores

La herramienta de empaquetado por excelencia en Linux y Unix es *tar.* Simplemente se trata de meter o guardar todos los archivos o directorios que necesitemos en un mismo archivo. Una forma sencilla de transportar archivos de un sitio a otro, ya sea en nuestra máquina o entre diferentes máquinas. Además *tar*, no solo te permite empaquetar esos archivos y directorios, sino que además te da la posibilidad de comprimirlos para de esta manera que ocupen menos espacio, y sean más fáciles de transportar.

Existen otros programas para manejar de manera óptima y fácil la agrupación y la compactación de un conjunto de archivos, como *gzip, bz2, xz, zip, lha, arj, rar, etc.* La mayoría de los programas para comprimir y descomprimir son secuenciales (solo usan un core), pero hay otros que trabajan en paralelo, que permiten usar dos o más cores logrando un alto desempeño como: pigz, plzip, pbzip2, lrzip, lbzip2, pixz.

Para instalar los programas más comunes en Debian GNU/Linux usar:

```
# apt install arj bzip2 clzip cpio dtrx gzip kgb \
lbzip2 lhasa lrzip lzip lziprecover lzop ncompress \
p7zip-full p7zip-rar pax pbzip2 pigz pixz plzip \
rar rarcrack tarlz unace unace-nonfree unar unp \
unrar unzip xz-utils zip zpaq zstd zutils \
archivemount
```

Podemos conocer los programas que disponemos en nuestro equipo para compresión usando:

```
$ apropos compress
```

Lo Básico Empaquetar y Desempaquetar Empecemos por lo más básico, que es empaquetar y desempaquetar usando el comando *tar.* Lo distingo de comprimir y descomprimir, porque esto lo veremos más adelante, ya que se trata de dos conceptos distintos.

Para que sea más sencillo, creamos un conjunto de archivos con los que trabajaré a continuación. Para ello ejecuta lo siguiente:

```
$ touch archivo{1..20}.txt
```

Empaquetando Archivos se han creado 20 archivos que vamos a empaquetar a continuación. Para ello, ejecutamos lo siguiente[82]:

```
$ tar -cvf empaquetado.tar *.txt
```

ahora tenemos todos los archivos en un mismo paquete.

¿Que son esas opciones que he puesto? -c es para crear un archivo
-v muestra el progreso del comando
-f para indicar el archivo que se va a crear en este caso[83]

Desempaquetando Archivos ahora que ya tenemos claro como empaquetar archivos, es necesario que se puedan desempaquetar. Es tan importante una operación como otra. Así, para desempaquetar, tenemos que utilizar la opción *-x*, de extraer. Así, para desempaquetar, ejecuta la siguiente instrucción:

```
$ tar -xvf empaquetado.tar
```

Extraer un Archivo de un Paquete para extraer un o algunos archivos del paquete ejecutamos:

```
$ tar -xvf empaquetado.tar archivo1 archivo2
```

Extraer un grupo de Archivos de un Paquete si necesitamos extraer un grupo de archivos podemos usar comodines, por ejemplo:

```
$ tar -xvf empaquetado.tar –wildcards '*.php'
```

[82] Esto también se puede poner por separado, como:
$ tar -c -v -f empaquetado.tar *.txt

[83] Tienes que tener en cuenta que -f tiene que preceder al nombre del archivo que vas a crear, ya sea que lo utilices por separado o junto con otras opciones. Quiero decir, que -cvf empaquetado.tar funcionará, pero en cambio -fcv empaquetado.tar arrojará un error, esto es algo que debemos tener en cuenta.

Listando el Contenido para conocer el contenido del archivo *.tar* usamos:

```
$ tar -tvf empaquetado.tar
```

Si no indicamos la opción -v solo mostrará los archivos empaquetados. Utilizando esta opción, además muestra otra información como el propietario, los permisos o la fecha.

Verificar un Archivo Empaquetado para verificar el contenido de un archivo empaquetado, usamos:

```
$ tar -tvfW empaquetado.tar
```

Quitando un Archivo del Paquete Una vez creado un determinado paquete, es posible que necesitemos quitar un archivo, es decir, que el archivo no pertenezca a ese nuevo paquete. Para hacer esto, debemos tener en cuenta que el paquete no puede estar comprimido. En caso de que esté comprimido no funcionará. Así, para borrar un archivo de un paquete ejecutamos:,

```
$ tar -f empaquetado.tar –delete archivo12.txt
```

Añadir un Archivo a un Paquete otra situación común que nos encontramos con más o menos frecuencia es cuando ya creamos el paquete, y dejamos un archivo fuera. En este caso, hay varias opciones, desde empaquetar de nuevo, a simplemente añadir ese archivo al paquete. Así, por ejemplo, si lo que queremos es simplemente añadir el archivo al paquete recurrimos a esta opción:

```
$ tar -f empaquetado.tar –append archivo12.txt
```

pero, ¿y si el archivo ya existe y simplemente lo queremos modificar? En este caso, usamos la opción:

```
$ tar -f empaquetado.tar –update archivo12.md
```

No solo esto, sino que también podemos comparar si un archivo que tenemos empaquetado es distinto del que no está empaquetado. Lo cierto es que lo vamos a comparar por fecha y por tamaño, únicamente, pero por lo menos ya sabemos si se ha modificado. Para hacer esto, ejecutamos:

```
$ tar -f empaquetado.tar –diff file12.txt
```

Esto devolverá un resultado como:

```
archivo12.txt: La fecha de modificación es distinta
archivo12.txt: El tamaño es distinto
```

Excluir Archivos o Directorios es común al respaldar excluir algunos archivos o directorios de nuestro respaldo, para ello podemos usar la opción *–exclude* en el que podemos indicar archivo que deseamos excluir del empaquetado, por ejemplo:

```
$ tar -cvf empaquetado.tar *.txt –exclude='archivo.txt'
```

o podemos indicar el directorio a excluir:

```
$ tar -cvf empaquetado.tar *.txt –exclude=carpeta/ignoraEsta
```

Cifrar y Descifrar es posible cifrar[84] el contenido que deseamos empaquetar usando OpenSSL para generar un archivo con extensión *.tar.gz*, por ejemplo, para cifrar el contenido del actual directorio usamos:

```
$ tar -czf - * | openssl enc -e -aes256 -out archivoSeguro.tar.gz
```

y para hacer el proceso de descifrado usamos:

```
$ openssl enc -d -aes256 -in archivoSeguro.tar.gz | tar xz -C
prueba
```

el cual será extraído en el subdirectorio *prueba*.

[84] El cifrado consiste en ocultar y mantener en secreto la información cifrandola, el cifrado garantiza que la información sin formato cuando se almacena o envía por la red sea ilegible. El descifrado permite volver hacer legible la información. Existen distintas formas de cifrado: cifrado simétrico y asimétrico.

Comprimir y Descomprimir Hasta el momento todo lo memos hecho empaquetando archivos, pero si lo que requerimos es reducir el tamaño total ocupado, tenemos que comprimir esos paquetes. Respecto a la compresión hay distintas opciones:

- gzip es la compresión por defecto, y la puedes declarar utilizando la opción -z.
- bzip2 en este caso, tienes que utilizar la opción -j.

Ahora necesitamos combinar estas dos opciones con todo lo que hemos visto hasta el momento. Así para empaquetar y comprimir utilizamos:

```
$ tar -cvzf empaquetado.tar.gz *.txt
```

Para el caso de descomprimir:

```
$ tar -xvzf empaquetado.tar.gz
```

Para ver el contenido del archivo:

```
$ tar -tvf empaquetado.tar.gz
```

Para revisar la integridad de un archivo usamos:

```
$ gunzip -t empaquetado.tar.gz
```

también podemos integrar lo visto anteriormente en un programa Bash como el mostrado en (véase 5.8).

Ahora bien, existe una diferencia en cuanto a las posibilidades que ofrece un archivo empaquetado y otro comprimido. Esta diferencia radica en las operaciones que podemos efectuar. Así, con un archivo comprimido, no podremos actualizarlo, al menos en primera instancia. Es decir, no podemos utilizar las opciones –delete, –append o –update.

Una solución es desempaquetar, operar y comprimir, como por ejemplo:

```
$ gunzip empaquetado.tar.gz; tar -f empaquetado.tar –delete
archivo5.txt; gzip empaquetado.tar
```

Estas son algunas de las opciones que permite hacer *tar*. Y digo bien, simplemente algunas de las opciones, y sin lugar a dudas las más habituales. Y es que *tar* es toda una navaja suiza que permite empaquetar archivos y directorios de decenas de formas posibles, no solo lo que hemos visto hasta el momento. También, se puede especificar directorios, excluir archivos por nombre o incluso por patrón, y mucho más.

Cifrar y Descifrar Archivos es posible cifrar archivos usando criptografía de clave pública o sólo una clave de cifrado simétrico. La clave que se usa para el cifrado simétrico deriva de la contraseña dada en el momento de cifrar el documento.

GnuPG es un paquete completo que permite generar un par de claves públicas, intercambiar y comprobar la autenticidad de claves, cifrar y descifrar archivos, etc. Para instalar el paquete GnuPG, usamos:

```
# apt install gnupg
```

Cifrar un Archivo para cifrar un archivo se usa el programa *gpg* con la opción *-c* la cual nos permitirá indicar la clave de cifrado y su confirmación, por ejemplo:

```
$ gpg -c archivo.tar
```

el archivo resultante será: archivo.tar.pgp. Es necesario escoger claves de cifrado robustas, una opción para generarlas es GnuPG, para generar una clave aleatoria de 32 caracteres, usamos:

```
$ gpg –gen-random –armor 1 32
```

Descifrar un Archivo para descifrar un archivo cifrado con GnuPG, usamos el comando *gpg* al cual se le indica el nombre del archivo cifrado, por ejemplo:

```
$ gpg archivo.tar.pgp
```

el archivo resultante será: archivo.tar.

Cifrar y Descifrar Usando tar podemos generar un archivo compactado y cifrarlo en una mismo comando mediante:

```
$ tar -cvzf - directorio | gpg -c > archivo.tar.gz.gpg
```

también podemos indicar la clave de cifrado -no es seguro pues otro usuario podría verla- mediante:

```
$ tar -cvzf - directorio | gpg -c –passphrase miclave > archivo.tar.gz.gpg
```

y para descifrar usamos:

```
$ gpg -d archivo.tar.gz.gpg | tar -xvzf -
```

Cifrar y Descifrar Usando ccrypt este comando permite cifrar usando *AES256*, no viene instalado por omisión en el sistema, pero lo podemos instalar usando:

```
# apt install ccrypt
```

Para cifrar un archivo *archivo.tar* usamos:

```
$ ccrypt archivo.tar
```

esto nos generará un archivo *archivo.tar.cpt*, para descifrarlo usamos:

```
$ ccrypt -d archivo.tar.cpt
```

que nos retornará a nuestro archivo original.

Cifrar y Descifrar Usando age este comando permite cifrar usando una clave pública o frase, no viene instalado por omisión en el sistema, pero lo podemos instalar usando:

```
# apt install age
```

Para generar una clave pública en el archivo llave.txt usamos:

```
$ age-keygen -o llave.txt
```

Para cifrar un archivo *archivo.tar* con clave pública usamos:

```
$ tar cvz archivos | age -r llave.txt > archivo.tar.gz.age
```

para descifrarlo usamos:

```
$ age –decrypt -i llave.txt > archivo.tar.gz
```

que nos retornará a nuestro archivo original.
Para cifrar un archivo *archivo.tar* con frase usamos:

```
$ age –passphrase –output archivo.tar.gz.age archivo.tar.gz
```

para descifrarlo usamos:

```
$ age –decrypt –output archivo.tar.gz archivo.tar.gz.age
```

que nos retornará a nuestro archivo original.

Respaldos Incrementales Un respaldo incremental es aquel que almacena solamente las diferencias con respecto al respaldo anterior, tar posee dos opciones que nos van a servir para trabajar con respaldos incrementales:

- «-listed-incremental=file», o «-g file», que permite especificar un archivo de respaldo incremental.
- «-incremental» o «-G», que permite analizar un respaldo incremental y saber qué archivos del total están presentes en dicho respaldo.

El archivo de respaldo incremental almacenará metadatos (.snar) que ayudará a la herramienta tar a determinar qué archivos han cambiado desde el último incremental, cuáles se han agregado, o cuales se han eliminado, de modo que tar podrá generar el siguiente respaldo incremental solamente de los cambios del anterior. Por ejemplo:
Creamos el respaldo completo (la base para los respaldos incrementales)[85]:

```
$ tar -cvzf Respaldos/bkp0.tgz -g Respaldos/incremental.snar
datos/
```

Supongamos ahora que realizamos el respaldo incremental del siguiente día, usando;

```
$ tar -cvzf Respaldos/bkp1.tgz -g Respaldos/incremental.snar
datos/
```

de esta manera realizaremos todos los respaldos incrementales necesarios.
Veamos, antes de proceder a restaurar todo, cómo podemos analizar qué contiene cada respaldo incremental. Esto podemos llevarlo a cabo mediante la opción «-incremental» del comando tar, a la que debemos también agregar dos modificadores «-verbose» para ver la información completa. En mi caso he usado «-G» en vez de «-incremental», y «-vv» en vez de «-verbose -verbose»:

```
$ tar -tvvGf Respaldos/bkp1.tgz
```

[85] Aquí usamos para los ejemplos de los respaldos comprimidos con gzip, pero el comportamiento es exactamente el mismo al utilizar bzip2, xz, lmza o cualquier otro formato soportado.

como puede verse en la salida, delante de cada nombre de archivo tenemos un «Y» o un «N». «Y» significa que el archivo está presente en dicho respaldo, y «N» que no lo está. El resultado es consistente con los cambios que hemos realizado sobre los archivos.

Supongamos ahora que perdemos todos los datos y que necesitamos recuperarlos desde el respaldo completo y los incrementales. Para extraer todos los archivos en el mismo estado exacto en el que estaban a la hora de realizar el último respaldo incremental, debemos usar la opción «-extract» o «-x» junto con la opción «-listed-incremental». Como la información de respaldo incremental está almacenada dentro del Tarball, es decir, el Tarball contiene solo los cambios respecto del último incremental, puede pasarse cualquier argumento al «-listed-incremental» o «-g». Usualmente se utiliza /dev/null, o se usa directamente «-incremental» o «-G».

Suponiendo que el directorio datos no existe, vamos a proceder a restaurar todos los respaldo, desde el primero, completo, uno a uno hasta el último incremental disponible, para regenerar el directorio original y su contenido:

```
$ tar xvGf Respaldos/bkp0.tgz
$ tar xvGf Respaldos/bkp1.tgz
$ tar xvGf Respaldos/bkpn.tgz
```

de esta forma se puede restaurar paso por paso todos los respaldo que antes hicimos, y ahora tenemos dentro del directorio *datos/* toda la información de nuevo. Y sí, los archivos tienen el contenido original modificado tal y como quedaron en el último respaldo incremental.

Zips de la Muerte Cuando descomprimimos un archivo debemos tener cuidado, ya que el resultado suele ocupar bastante más y puede dejarnos sin espacio en el ordenador. El ratio máximo de compresión de la mayoría de zips está marcado en 1032 a uno, aunque en muchas ocasiones tampoco se alcanza. Sin embargo, desde 1996 se tiene constancia de la existencia de las llamadas "bombas zip" o lo que se conoce popularmente como "el zip de la muerte". Un archivo que sobrepasa este límite de descompresión e inunda nuestro ordenador con miles de millones de Bytes.

Una bomba zip es un archivo comprimido como cualquier otro. Su potencial peligro está en que es muy fácil de esconder y compartir. Pero al descomprimir es cuando colapsa el ordenador al liberar una gran cantidad de datos.

El zip de la muerte más conocido es 42.zip (https://unforgettable.dk), un archivo con un récord de compresión de 106.000 millones a uno. Su autor es desconocido, pero el ratio es impresionante. Estamos hablando de un archivo que pesa únicamente 42 KiloBytes (de ahí el nombre) y que al descomprimirse puede alcanzar los 4.3 GigaBytes. Pero la clave está en que el archivo contiene cinco capas, cada una con 16 archivos. Lo que en total una vez liberados todos ocupan un total de 4.5 PetaBytes.

David Fifield (https://www.bamSoftware.com/hacks/zipbomb/) nos ofrece tres archivos de ejemplo (uno de ellos no recursivo que establece el ratio en 28 millones a uno). El primero donde pasamos de 42kB a 5.5GB, un segundo de 10MB a 281TB y un tercero de 46MB a 4.5 PetaBytes, este último únicamente compatible con el formato Zip64.

Los antivirus sí las detectan afortunadamente para la seguridad de los usuarios, esta técnica de la bomba zip es bastante conocida desde hace muchos años. Pese a que el archivo comprimido pasa desapercibido en el sistema, una vez vamos a descomprimirlo es cuando los antivirus lo detectan como un posible peligro y nos advierten antes de iniciar el proceso.

También los hay en formato TAR sin tratarse de ningún gusano o virus, otro ataque similar es la 'bomba fork' o 'wabbit'. En este caso, es un archivo que crea copias de sí mismo. Una técnica de la que los usuarios de Linux tampoco se libran, ya que también funciona con archivos TAR.

Trabajando con Archivos Compactados Existe la opción de gestionar ficheros comprimidos, gracias a los comandos *zgrep, zegrep, zless, zmore, zdiff, zcmp, zcat* entre otros. Como te puedes dar cuenta la función de cada uno de estos comandos será la misma que su homónimos sin «z» (*grep, egrep, less, more, diff, cmp, cat*) pero para ficheros comprimidos con *gzip* y extensión *.gz*. A saber:

- zcat: Mostar por pantalla el contenido de un fichero comprimido.

  $ zcat archivo.gz | more

  en el caso de necesitar conocer las propiedades del archivo compactado, usamos:

```
$ zcat -l archivo.gz
```

y en el caso de que el comando *zcat* nos muestre avisos, los podemos callar usando:

```
$ zcat -q archivo.gz
```

- zless y zmore: Comandos para examinar ficheros de texto plano o comprimidos de forma paginada por la pantalla.

```
$ zless archivo.gz
```

- zgrep y zegrep: Comando para realizar búsquedas de expresiones regulares en ficheros comprimidos.

```
$ zgrep -i "cadena" archivo.gz
```

- zdiff y zcmp: Comando para comparar ficheros comprimidos.

```
$ zdiff archivo1.gz archivo2.gz
```

De forma análoga para archivos comprimidos usando *bz2*, están los comandos *bzgrep, bzegrep, bzless, bzmore, bzdiff, bzcmp, bzcat* y para archivos comprimidos usando *xz*, están los comandos *xzgrep, xzegrep, xzless, xzmore, xzdiff, xzcmp, xzcat*.

Desempaquetar o Descomprimir El uso básico para desempaquetar o descomprimir según la extensión del archivo es el siguiente:

- *.tar Desempaquetar usando: tar xf archivo.tar
- *.tar.bz2 Descomprimir usando: tar xjf archivo.tar.bz
- *.tbz Descomprimir usando: tar xjf archivo.tbz
- *.tbz2 Descomprimir usando: tar xjf archivo.tbz2
- *.tgz Descomprimir usando: tar xzf archivo.tgz
- *.tar.gz Descomprimir usando: tar xzf archivo.tar.gz

- *.tar.lz Descomprimir usando: tar -xf archivo.lz
- *.tar.xz Descomprimir usando: tar -xf archivo.tar.xz
- *.xz Descomprimir usando: tar Jxf archivo.xz
- *.gz Descomprimir usando: gunzip archivo.gz
- *.bz2 Descomprimir usando: bunzip2 archivo.bz2
- *.zip Descomprimir usando: unzip archivo.zip
- *.Z Descomprimir usando: uncompress archivo.Z
- *.7z Descomprimir usando: 7z e archivo.7z
- *.zst Descomprimir usando: zstd -d archivo.zst
- *.lha Descomprimir usando: lha x archivo.lha
- *.arj Descomprimir usando: arj x archivo.arj
- *.zoo Descomprimir usando: zoo x archivo.zoo
- *.lz Descomprimir usando: lzip -d archivo.lz
- *.cpio Desempaquetar usando: cpio -idv < archivo.cpio
- *.rar Descompactar usando: rar x archivo.rar
- * Descompactar[86] usando: unp archivoCompactado
- * Descompactar usando: dtrx archivoCompactado
- * Descompactar usando: unar archivoCompactado

[86] El programa unp permite descomprimir un archivo de casi cualquier formato, para instalar el paquete usamos:

```
# apt install unp
```

para descompactar, usamos:

```
$ unp archivoCompactado
```

El proceso de instalación y uso es similar para los paquetes: dtrx y unar.

Formato *tar.gz* para respaldar y compactar un grupo de archivos y/o directorios en formato *tar.gz*, usamos:

```
$ tar -cvf nombre.tar.gz directorio
$ gzip –best nombre.tar
```

o usamos:

```
$ tar -zcvf nombre.tar.gz directorio
```

o usamos:

```
$ tar -jcvf nombre.tar.bz2 directorio
```

para restaurar, usamos:

```
$ gunzip nombre.tar.gz
$ tar -xvf nombre.tar
```

o usamos:

```
$ tar -zxvf nombre.tar.gz
```

o usamos:

```
$ tar -jxvf nombre.tar.bz2
```

Si se requiere descomprimir múltiples archivos empaquetados podemos usar:

```
$ for z in *.tar.gz; do tar -zcvf "$z"; done
```

Formato *gz* para compactar archivos y/o directorios al formato *gzip*, usamos:

```
$ gzip ficheros
```

y para descompactarlo usamos:

```
$ gzip -dk fichero.gz
```

si solo nos interesa descompactar y no preservar el archivo .gz:

```
$ gzip -d fichero.gz
```

o

```
$gunzip fichero.gz
```

Si se requiere descomprimir múltiples archivos empaquetados podemos usar:

```
$ for z in *.gz; do gunzip "$z"; done
```

También podemos hacer la compactación en paralelo usando múltiples cores, mediante el paquete *pigz* con formato *gzip* (-0 sin compresión, -1 compresión más rápida, -6 compresión por omisión y -9 mejor compresión), para compactar manteniendo el archivo original, usamos:

```
$ pigz -k archivo.iso
```

generara un archivo.izo.gz. Podemos listar el contenido de un .gz usando:

```
$ pigz -l archivo.iso.gz
```

Para comprimir un directorio, es necesario usar tar, mediante:

```
$ tar cf - Directorio/ | pigz > archivo.tar.gz
```

o

```
$ tar –use-compress-program=pigz -cf archivo.tar.gz Directo-
rio/
```

Para descomprimir usamos:

```
$ pigz -d archivo.iso.gz
```

o

```
$ unpigz archivo.iso.gz
```

Por omisión *pigz* usa todos los cores de la máquina, si necesitamos limitar el número de cores usamos la bandera -p y el número de cores a usar, por ejemplo:

```
$ pigz -9 -k -p3 archivo.iso
```

Podemos usar otros formatos de compresión, usando la opción -k -z para zlib (.zz), usando:

```
$ pzip -k -k archivo.iso
```

o usando la opción -k -K para zip (.zip):

```
$ pzip -k -K archivo.iso
```

Formato *bz2* para compactar archivos y/o directorios al formato *bz2, bz, tbz2 tbz o bzip2*, usamos:

```
$ bzip2 ficheros
```

y para descompactarlo usamos:

```
$ bzip2 -d fichero.bz2
```

podemos descomprimir usando el comando **bzcat** mediante:

```
$ bzcat -dc fichero.bz2
```

o bien descomprimir usando el comando **bunzip2** mediante:

```
$ bunzip2 fichero.bz2
```

Si se requiere descomprimir múltiples archivos empaquetados podemos usar:

```
$ for z in *.bz2; do bunzip2 "$z"; done
```

Formato *xz* para respaldar y compactar un grupo de archivos y/o directorios en formato *tar.xz*, usamos:

```
$ tar -cvJf nombre.tar.xz directorio
$ xz nombre.tar
$ xz -k archivo.tar
$ tar –xz -cf - /trayectoria | ssh usuario@maquina "cat > archivo.txz"
```

podemos controlar el nivel de compresión usando valores de 0 (sin compresión) al 9 (mejor compresión), mediante:

```
$ xz -9 archivo.tar
$ xz –k9 archivo.tar
```

podemos descomprimir usando:

```
$ tar -xf nombre.tar.xz
$ tar -xvf nombre.tar.xz
$ tar -Jxvf nombre.tar.xz
$ tar -xf nombre.tar.xz
$ tar –xz -xf archivo.txz
$ xz -v -d nombre.tar.xz
$ xz –decompress archivo.tar.xz
$ unxz archivo.xz
```

y podemos ver el contenido de un archivo *.tar.xz* si usamos:

```
$ tar ft archivo.tar.xz
```

Si se requiere descomprimir múltiples archivos empaquetados podemos usar:

```
$ for z in *.tar.xz; do tar-xf "$z"; done
```

Formato *zip* permite respaldar en un solo archivo un grupo de archivos y/o directorios compactándolos, para ello usar:

```
$ zip archivo.zip fichero(s)
$ zip -r archivo.zip directorio/
```

también permite establecer los niveles de compresión usando una escala de 0 a 9 (por omisión es 6), por ejemplo:

```
$ zip -8 archivo.zip fichero(s)
$ zip -8 -r archivo.zip directorio/
```

y podemos borrar los archivos que se comprimirán al terminar el proceso, usando:

```
$ zip -m archivo.zip fichero(s)
$ zip -rm archivo.zip directorio/
```

Podemos agregar un archivo a un *.zip* existente, usando:

```
$ zip -u archivo.zip nuevo.txt
```

o eliminar un archivo a un .zip existente, usando:

```
$ zip -d archivo.zip porBorrar.txt
```

Podemos crear un *.zip* protegido con clave, usando:

```
$ zip -e archivo.zip fichero(s)
```

si necesitamos poner una clave de acceso (digamos 2GAXTW), usamos

```
$ zip -P 2GAXTW -r archivo.zip ficheros
```

o proteger con clave un *.zip* ya existente, usando:

```
$ zipcloak archivo.zip
```

Podemos ver la información detallada del archivo comprimido, usando:

```
$ zipdetails archivo.zip
```

Y descomprimir usando:

```
$ unzip archivo.zip
```

para descompactar un solo directorio usamos:

```
$ unzip -d directorio archivo.zip
```

también podemos indicar el directorio en en cual descomprimir, usando:

```
$ unzip archivo.zip -d /home/usuario/temp/
```

para descomprimir un *.zip* protegido con clave (digamos 2GAXTW), usamos:

```
$ unzip -P 2GAXTW archivo.zip
```

para extraer sin mostrar salida de los archivos extraídos, usamos:

```
$ unzip -q archivo.zip
```

para indicar que no se extraigan algunos archivos, usamos:

```
$ unzip archivo.zip -x *.eps
```

para sobreescribir los archivos al descomprimir sin pedir confirmación, usamos:

```
$ unzip -o archivo.zip
```

para no sobreescribir los archivos al descomprimir, usamos:

```
$ unzip -n arvhivo.zip
```

para actualizar archivos y en su caso crearlos si es necesario, usamos:

```
$ unzip -u archivo.zip
```

para actualizar archivos pero no crear nuevos, usamos:

```
$ unzip -f archivo.zip
```

para listar el contenido de un *.zip*, usamos:

```
$ unzip -l archivo.zip
```

para obtener información detallada de un *.zip*, usamos:

```
$ unzip -v archivo.zip
```

para revisar la integridad de un *.zip*, usamos:

```
$ unzip -t archivo.zip
```

Si al descompactar existe algún error, es posible recuperar parte de los archivos mediante:

```
$ zip -F archive.zip -O archive-fixed.zip
```

o usar *-FF*, después usar:

```
$ jar xvf archive-fixed.zip
```

otra alternativa es usar:

```
$ bsdtar xf archivo.zip
```

Si se requiere descomprimir múltiples archivos empaquetados podemos usar:

```
$ for z in *.zip; do unzip "$z"; done
```

Formato *7z* para respaldar y compactar un grupo de archivos y/o directorios en formato *7-zip*, usamos:

```
$ 7z a archivo.7z ficheros
```

también podemos pedir que use una clave (password$$) para cifrar el archivo:

```
$ 7z a archivo.7z ficheros -ppassword$$
```

para restaurar usamos:

```
$ 7z e archivo.7z
```

Si se requiere descomprimir múltiples archivos empaquetados podemos usar:

```
$ for z in *.7z; do 7z e "$z"; done
```

Formato *zst* para respaldar y compactar un grupo de archivos y/o directorios en formato *zst*, usamos:

```
$ zstd ficheros -o archivo.zst
```

para restaurar usamos:

```
$ zstd -d archivo.zst
```

o usamos:

```
$ unzstd archivo.zst
```

Si se requiere descomprimir múltiples archivos empaquetados podemos usar:

```
$ for z in *.zst; do unzstd "$z"; done
```

Formato *lha* para respaldar y compactar un grupo de archivos y/o directorios en formato *lha*, usamos:

```
$ lha a archivo.lha ficheros
```

para restaurar usamos:

```
$ lha x archivo.lha
```

Si se requiere descomprimir múltiples archivos empaquetados podemos usar:

```
$ for z in *.lha; do lha x "$z"; done
```

Formato *arj* para respaldar y compactar un grupo de archivos y/o directorios en formato *arj*, usamos:

```
$ arj a archivo.arj ficheros
```

para restaurar usamos:

```
$ arj x archivo.arj
$ unarj archivo.arj
```

Si se requiere descomprimir múltiples archivos empaquetados podemos usar:

```
$ for z in *.arj; do arj x "$z"; done
```

Formato *zoo* para respaldar y compactar un grupo de archivos y/o directorios en formato *zoo*, usamos:

```
$ zoo a archivo.zoo ficheros
```

para restaurar usamos:

```
$ zoo x archivo.zoo
```

Si se requiere descomprimir múltiples archivos empaquetados podemos usar:

```
$ for z in *.zoo; do zoo x "$z"; done
```

Formato lz y tlz para compactar un *archivo* y remplazarlo por *archivo.lz*, usamos:

```
$ lzip -v archivo
```

para compactar todo un dispositivo (por ejemplo */dev/sdc*), usamos:

```
$ lzip -c /dev/sdc > archivo.lz
```

para restaurar usamos:

```
$ tar -xf archivo.lz
$ lzip -d archivo.lz
```

o extraer un archivo multivolumen de un archivo tar

```
$ lzip -cd archivo.tar.lz | tar -xf -
```

Podemos usar *tarlz* que es una combinación de *tar* y *lzip* que trabaja en paralelo para hacer la compresión, por ejemplo de los archivos *a*, *b*, y *c*:

```
$ tarlz -cf archivo.tar.lz a b c
```

y podemos adicionarle los archivos *d* y *e*, usando:

```
$ tarlz -rf archivo.tar.lz d e
```

Para extraer los archivos podemos usar:

```
$ tarlz -xf archivo.tar.lz
```

También podemos usar la versión plzip que trabaja en paralelo. Para compactar un *archivo* y remplazarlo por *archivo.lz*, usamos:

```
$ plzip -v archivo
```

para compactar todo un dispositivo (por ejemplo */dev/sdc*), usamos:

```
$ plzip -c /dev/sdc > archivo.lz
```

y para restaurar usamos:

```
$ plzip -d archivo.lz
```

Formato cpio para empaquetar archivos (por ejemplo*.txt) en un archivo .cpio, usamos:

```
$ ls *.txt | cpio -ov > archivo.cpio
```

para desempaquetar, usamos:

```
$ cpio -idv < archivo.cpio
```

Formato *rar* para respaldar y compactar un grupo de archivos y/o directorios al formato *rar*, usamos:

```
$ rar a archivo.rar ficheros
```

para restaurar usamos:

```
$ rar x archivo.rar
$ unrar archivo.rar
```

En algunos casos, archivos *rar* de Windows no es posible descomprimirlos correctamente en Linux, para descomprimirlos podemos descargar utilerías GNU para Win32 de:

http://unxutils.sourceforge.net/

entre ellas, descargar *unrar* (es un sólo archivo zip) de la dirección:

http://sourceforge.net/projects/unxutils/

ahora usando Wine es posible descomprimir los archivos desde Linux mediante:

```
$ wine unrar.exe e archivo.rar
```

Para descomprimir archivos *rar* o *zip* rotos, usar:

```
$ unrar e -kb -y nombreArchivo.rar
```

o usamos:

```
$ bsdtar xf nombreArchivo.zip
```

Si se requiere descomprimir múltiples archivos empaquetados podemos usar:

```
$ for z in *.rar; do unrar e "$z"; done
```

UNP para descompactar casi de cualquier formato existe un programa llamado *unp* (basado en un Scrip de Perl) que permite descomprimir de casi cualquier formato, para instalarlo hacemos:

```
# apt install unp
```

Es un extractor universal, su uso es de lo más sencillo, a saber:

```
$ unp archivoCompactado
```

Parte de los formatos soportados y por que programas se muestra en la siguiente lista (generada usando: *unp -s*):

7z (p7zip or p7zip-full), ace (unace), ar,deb (binutils), arj (arj), bz2 (bzip2), cab (cabextract), chm (libchm-bin or archmage), cpio,afio (cpio or afio), dat (tnef), dms (xdms), exe (orange or unzip or unrar or unarj or lha), gz (gzip), cpio,afio (cpio or afio), dat (tnef), dms (xdms), exe (orange or unzip or unrar or unarj or lha), gz (gzip), hqx (macutils), lha,lzh (lha), lz(lzip), lzma (xz-utils or lzma), lzo (lzop), lzx (unlzx), mbox (formail and mpack), pmd (ppmd), rar (rar or unrar or unrar-free), rpm (rpm2cpio and cpio), sea,sea.bin (macutils), shar (sharutils), tar (tar), tar.bz2,tbz2 (tar with bzip2), tar.lzip (tar with lzip), tar.lzop,tzo (tar with lzop), tar.xz,txz (tar with xz-utils), tar.z (tar with compress), tgz,tar.gz (tar with gzip), uu (sharutils), xz (xz-utils), zip,cbz,cbr,jar,war,ear,xpi,adf (unzip), zoo (zoo).

DTRX para descompactar de múltiples formatos esta herramienta llamada *dtrx* (Do The Right Extraction), es una aplicación de código abierto que simplifica el proceso de extracción de archivos comprimidos. Para su instalación hacemos:

```
# apt install dtrx
```

Y ya estamos listos para comenzar a usarlo en la terminal. Dtrx soporta los formatos comunes de compresión como: tar, bz2, zip, cpio, deb, rpm, gem, 7z, cab, lzh, rar, gz, bz2, lzma, xz, además de binarios deb, gem (ruby), archivos de Installshield e incluso algunos tipos de exe. También descomprime archivos comprimidos con gzip, bzip2, lzma, xz, y otros compresores.

Para descomprimir usamos:

```
$ dtrx archivo.zip
```

Podemos hacer un preview del contenido de un archivo lanzandolo con el parámetro -l:

```
$ dtrx -l archivo.zip
```

Si tenemos un archivo comprimido que a su vez incluye múltiples ficheros, al ejecutar *dtrx* nos va a dar 5 opciones diferentes, para hacer la cosa más o menos automática:

-a siempre extraer los archivos incluidos durante esta sesión

-o extraer los archivos incluidos esta vez.

-n elegir no extraer los archivos incluidos por esta vez.

-v nunca extraer los archivos incluidos durante esta sesión.

-l listar los archivos incluidos.

Si queremos que trabaje de forma recursiva, lo podemos hacer añadiendo la opción -r:

```
$ dtrx -r archivo.tar.gz
```

Además con el parámetro -m podemos extraer los metadatos de archivos *deb* y *gem*:

```
$ dtrx -m archivo
```

UNAR para descompactar de múltiples formatos esta herramienta llamada *unar*, es una aplicación de código abierto que simplifica el proceso de extracción de archivos comprimidos. Para su instalación hacemos:

```
# apt install unar
```

Y ya estamos listos para comenzar a usarlo en la terminal, *unar* soporta los formatos comunes de compresión como: zip, rar, 7z, tar, gzip, bzip2, lzma, xz, cab, msi, nsis, exe, iso, bin, arj, arc, pakm, acem, lzh, adf, entre otros.

Para descomprimir usamos:

```
$ unar archivo.zip
```

Otras opciones cuando se tiene una lista de archivos de distintas trayectorias o es resultado de una búsqueda, para compactar es preferible usar *afio*. Podemos instalarlo usando:

```
# apt install afio
```

Para compactar, digamos todos los archivos **.?pp* (programas de C++) usar:

```
$ find . -name *.?pp | afio -o -Z fuentes
```

para descompactarlos, usar:

```
$ afio -i -Z fuentes
```

Si se desea compactar usando GZIP, usar:

```
$ cat lista | afio -o -Z -G 9 fuentes
```

Si se desea ver el listado de archivos que contiene fuentes, usar:

```
$ afio -t fuentes
```

Si se desea compactar y mandar a otra máquina usar:

```
$ find . -name *.?pp | afio -o -Z user@servidor%ssh:fuentes
```

Como el uso de *afio* no necesita extensión en el archivo, para descompactarlo de cualquier formato es recomendable usar *unp*, este escoge el mejor método para el archivo en cuestión:

```
$ unp archivoCompactado
```

Respaldo y/o Restauración Remoto es posible usar *ssh*[87] en conjunción con *tar o dd*[88] para hacer respaldos y/o restauraciones de forma remota, por ejemplo:

```
$ ssh user@maq tar czf - /dir1/ > /dest/arch.tar.gz
$ ssh user@maq 'cd /dir1/ && tar -cf - file | gzip -9' >arch.tar.gz
$ tar zcvf - /dir | ssh user@maquina "cat > /dest/arch.tar.gz"
$ tar zcf - /dir/ | gpg -e | ssh user@maq 'cat - > arch.tar.gz.gpg'
$ ssh user@maq 'tar zcf - /some/dir' | tar zxf -
$ ssh user@maq 'tar czf - /home/user' | tar xvzf - -C /home/user
$ cat arch.tar.gz | ssh user@maq "tar zxvf -"
$ cat arch.tar.gz | ssh user@maq "cd /dest/; tar zxvf -"
# dd if=/dev/sdvf | ssh user@maq 'dd of=arch.img'
$ ssh root@maq 'dd if=arch.img' | dd of=/dev/sdvf
$ tar cvzf - /dir | ssh root@maq "dd of=/dest/arch.tar.gz"
$ tar cvjf - * | ssh user@maq "(cd /dest/; tar xjf -)"
$ tar cvzf - dir/ | ssh user@maq "cat > /dest/arch.tgz"
$ tar cvzf - /dir | ssh user@maq "dd of=/dest/arch.tar.gz"
$ ssh user@maq "cat /dest/arch.tar.gz" | tar xvzf -
$ tar cvjf - * | ssh root@maq "(cd /dest/; tar xjf - )"
```

Es posible usar *nc*[89] en conjunción con *tar* y *pv* para hacer respaldos y/o restauraciones de forma remota, supongamos que estamos en el IP 192.168.13.230. podemos mandar un archivo grande (digamos un *.iso)* y usar el comando *pv* para ver el progreso de la transferencia usando:

$ tar -zcf - Archivo.iso | pv | nc -l -p 5555 -q 5

y para recibirlo en el otro equipo usamos:

```
$ nc 192.168.13.230 5555 | pv | tar -zxf -
```

Además, ponemos también usar *nc* en conjunción con *tar* y *gpg* para hacer respaldos y/o restauraciones de forma remota, supongamos que estamos en el IP 192.168.13.230, podemos generar un archivo compactado, cifrarlo y enviarlo a otro equipo en una mismo comando mediante:

[87] ssh o Secure Shell, es un protocolo de administración remota que le permite al usuario controlar y modificar servidores remotos a través de un mecanismo de autentificación.

[88] dd o Data Duplicator permite copiar y convertir datos -en Linux cualquier dispositivo y partición se trata y gestiona como un archivo- de archivos a bajo nivel.

[89] El comando *netcat* (*nc*) es una utilidad que permite escribir y leer datos a través de conexiones de red usando los protocolos TCP y UDP

```
$ tar -cvzf - directorio | gpg -c | nc -l 6666
```

y para recibirlo en el otro equipo usamos:

```
$ nc 192.168.13.230 6666 | gpg -d | tar -xvzf -
```

Siempre es mejor opción usar *scp*, pero *nc* cumplirá con su cometido.

Montar Archivos Compactados con Archivemount ¿Alguna vez haz necesitado mirar el contenido de un fichero .tar, .tar.gz, entre otros, pero sin tener que descomprimirlo? Tal vez te gustaría extraer solo unos pocos ficheros, puede que lo que te interese es trabajar con una carpeta a la que le modificamos archivos, sin tener que archivar esta carpeta cada cierto tiempo.

En este caso, tenemos un sistema de ficheros bastante interesante que se llama *archivemount*, y que nos permite ver un fichero .tar, .tar.gz, entre otros. Como si de una carpeta local más se tratara. Al desmontar este sistema de fichero, se crea el mismo fichero de nuevo con los cambios hechos, de forma automática, y con previa copia del anterior.

Archivemount es un sistema de archivos basado en FUSE para variantes de Unix, incluido Linux. Su propósito es montar archivos (es decir, tar, tar.gz, etc.) en un punto de montaje donde se puede leer o escribir como con cualquier otro sistema de archivos. Esto hace que el acceso al contenido del archivo, que puede estar comprimido, sea transparente para otros programas, sin descomprimirlos.

Para instalarlo usamos:

```
# apt install archivemount
```

Para usarlo, necesitamos un directorio cualquiera donde montar nuestro *archivo.tar.gz*, por ejemplo *mnt*, entonces hacemos:

```
$ archivemount archivo.tar.gz mnt
```

ahora podemos ingresar al directorio *mnt* y trabajar con los archivos que tenia nuestro *archivo.tar.gz*, podemos visualizar el contenido, agregar o borrar archivos y al concluir para guardar los cambios necesitamos desmontarlo usando:

```
$ fusermount -u mnt
```

Esto generará un nuevo *archivo.tar.gz* con los cambios y el archivo anterior se guardará en *archivo.tar.gz.orig*.

Nota: si el archivo es de varios GigaBytes, entonces el montado y acceso a los archivos puede ser un proceso lento, para ello se desarrollo *ratarmount* (aplicación en Python), que permite montar un archivo *.tar, tar.gz*, entre otros, en modo de solo lectura, pero es notoriamente rápido para archivos compactados de varios GigaBytes de tamaño.

Montar Archivos Compactados con Ratarmount permite montar para su lectura un archivos .tar, tar.gz, entre otros. Esta es una aplicación de Python que se puede instalar para todos los usuario usando *pip3*, para ello usamos:

```
# pip3 install ratarmount
```

o instalar a nivel de usuario usando *pip3*, para ello usamos:

```
$ pip3 install ratarmount
```

usando instalación a nivel de usuario, para montar (la primera vez creará el índice que permitirá el acceso rápido a los archivos), usamos:

```
$ ~/.local/bin/ratarmount archivo.tar.gz mnt
```

una vez generado el índice (este proceso es algo lento), podemos ingresar al directorio *mnt* y trabajar con los archivos que tiene nuestro *archivo.tar.gz*, podemos visualizar el contenido y copiar archivos con una velocidad de acceso alta. Al concluir su uso, necesitamos desmontarlo usando:

```
$ fusermount -u mnt
```

como parte del proceso de acceso a nuestros archivo, se genera un archivo tmpXXXX, el cual podemos borrar usando:

```
$ rm tmp*
```

para más información del proyecto, podemos consultar:

```
https://github.com/mxmlnkn/ratarmount
```

5.3 Copiar Archivos Entre Equipos

scp permite transferir archivos y/o directorios de una máquina a otra de forma cifrada usando *SSH* (Secure Shell) que es un protocolo de administración remota que le permite al usuario controlar y modificar servidores remotos a través de un mecanismo de autentificación[90]. El comando *scp* (Secure Copy Protocol) tiene una sintaxis similar al del comando *cp*, con la salvedad que es necesario indicar el usuario, la máquina y el subdirectorio de trabajo del archivo y/o directorio para el destino, fuente o ambos.

Por ejemplo, si se desea transmitir un archivo a una máquina *192.168.13.230* con usuario *antonio*, en el directorio *~/Datos/* estando en sesión en otra máquina, se usa la siguiente sintaxis:

```
$ scp archivo.dat antonio@192.168.13.230:~/Datos/
```

Si se desea transmitir un subdirectorio a la máquina *192.168.13.230*, en el directorio *home* del usuario (denotado con .), se usa la siguiente sintaxis:

```
$ scp -r Directorio antonio@192.168.13.230:.
```

también podemos decirle que excluya algunos archivos (*.mp3) de la copia, usando:

```
$ scp -r !(*.mp3) Directorio antonio@192.168.13.230:.
```

Si se desea copiar un archivo de una máquina remota a nuestra máquina, usamos:

```
$ scp antonio@192.168.13.230:~/archivo ~/destino/
```

o de forma alternativa usamos (. indica el directorio donde el usuario se encuentra):

```
$ scp antonio@192.168.13.230:~/archivo .
```

Si se desea copiar de una máquina remota a otra máquina remota, usamos:

[90] En Android podemos hacer uso de SSH/SFTP mediante aplicaciones como: Termius, JuiceSSH, Mobile SSH, Advanced Client app, ConnectBot.

```
$ scp user1@HOST1:~/archivo user2@HOST2:~/
```

Si se desea transferir múltiples archivos podemos usar:

```
$ scp file1.txt file2.txt user@HOST:/home/user/
```

o de forma alternativa usamos (. indica el directorio donde el usuario se encuentra):

```
$ scp user@Host:/home/user/\{file1.txt,file2.txt\} .
```

En el caso que se quiera limitar el ancho de banda en la transmisión de archivos por *scp*, usar:

```
$ scp -l 400 user@server:/home/user/* .
```

En el caso de que se desee usar otro puerto distinto al de omisión (22) usar:

```
$ scp -P 4455 file.txt user@HOST:/home/user/file.txt
```

En el caso de querer incrementar la velocidad de transferencia en el uso de *scp*, la opción más viable es el cambiar el cifrado usada por omisión por otras como *3des-cbc, aes128-cbc, aes192-cbc, aes256-cbc, aes-128-ctr, aes192-ctr, eas256-ctr, arcfour256, arcfour, blowfish-cbc y cast128-cbc* mediante:

```
$ scp -c blowfish user@server:/home/user/file .
```

o de forma alternativa usamos:

```
$ scp -c arcfour256 user@HOST:/home/user/file .
```

Si adicionalmente se quiere compactar para reducir el tiempo de transferencia, usamos:

```
$ scp -C SourceFile user@HOST:/home/user/TargetFile
```

Si se desea que no se muestre información de la transferencia de los archivos al usar *scp* usar:

```
$ scp -q SourceFile user@HOST:/home/user/TargetFile
```

o si desea ver más información en la transferencia usar:

```
$ scp -v SourceFile user@HOST:/home/user/TargetFile
```

Si se instala *sshpass*, entonces hacemos:

```
$ sshpass -p "your_password" scp -r backup_user@target_ip:/home/
/backup/$name
```

rsync es una aplicación libre y multiplataforma que ofrece transmisión eficiente de datos incrementales, que opera también con datos comprimidos y cifrados. Mediante una técnica de Delta Encoding, que permite sincronizar archivos y directorios entre dos máquinas de una red o entre dos ubicaciones en una misma máquina, minimizando el volumen de datos transferidos. Es ideal para trabaja en varias máquinas en las que se desea tener sincronizada una o más carpetas, para instalar usamos:

```
# apt install rsync
```

La sintaxis básica es:

```
$ rsync [Opciones[91]] Origen [Origen]... Destino
```

por ejemplo, para hacer la sincronización de la carpeta *~/Datos* a *~/Respaldo* usamos:

```
$ rsync ~/Datos/ ~/Respaldo/
```

[91] Algunas opciones son: -r recursivo, -b backups, -R relativo, -u actualiza, -p muestra el progreso, -c comprime, -p preserva permisos, -v muestra lo que hace, -q trabaja en modo silencioso, -l preserva ligas simbólicas, -H preserva enlaces duros, -t preserva tiempos de modificación, etc.

Si queremos saber que hará el comando pero sin hacer la operación indicada, podemos usar la opción –dry-run, por ejemplo:

```
$ rsync –dry-run ~/Datos/ ~/Respaldo/
```

hay varias opciones que podemos usar en *rsync*, ejemplo de ellas es:

```
$ rsync –verbose –recursive –links –hard-links –times –delete
–stats ~/Datos/ ~/Respaldo/
```

en este caso se sincronizaría el contenido de ~/Datos con *~/Respaldo*, pero lo hará mostrando lo que transfiere, de forma recursiva, conservará enlaces simbólicos y sus tiempos, borrará los archivos que estén en el destino pero no en el origen y al terminar mostrará las estadísticas de la transferencia.

Para hacer la transmisión cifrada entre equipos, podemos usar *rsync* conjuntamente con *ssh*, supongamos que se esta en la máquina y quiere tener sincronizada la carpeta ~/Datos con mas de un equipo, mediante *ssh* usando un puerto *343*, en la máquina *192.168.13.230* y usuario *antonio*, usar:

```
$ rsync –partial –recursive –links –hard-links –times –verbose
–delete –stats ~/Datos/ -e 'ssh -p 343' antonio@192.168.13.230: ~/Respaldo/
```

por supuesto esto puede hacerse en cualquier dirección, i.e. de la máquina remota a nuestra máquina o viceversa, ejemplo:

```
$ rsync –partial –recursive –links –hard-links –times –verbose
–delete –stats -e 'ssh -p 343' antonio@192.168.13.230: ~/Respaldo/
~/Datos/
```

Los siguientes ejemplos permiten copiar los archivos (por ejemplo *.mp3) pero preservan la estructura de directorios que existía en su lugar de origen:

```
$ rsync -a -m –include '*/' –include '*.mp3' –exclude '*' ~/miDi-
rectorio/ ~/OtrDirectorio
$ rsync -a –prune-empty-dirs –include '*/' –include '*.mp3'
–exclude '*' ~/miDirectorio/ ~/OtrDirectorio
```

inicia la búsqueda en *~/midirectorio/* y los copia en *~/OtroDirectorio*.

Algunas veces necesitamos cambiar la prioridad de ejecución de un proceso *rsync* para evitar saturar el sistema, para ello podemos usar *ionice* que cambiara la prioridad de los comandos en ejecución, en este caso para todos los procesos de rsync se ejecutarán solo cuando la carga en el sistema sea ligera, mediante:

```
$ pgrep rsync | xargs ionice -c3 -p
```

Entre las opciones para excluir archivos o directorios en el uso de rsync, tenemos:

```
$ rsync -av –exclude 'archivo.txt' /fuente/ /destino/
$ rsync -av –exclude 'miDirectorio' /fuente/ /destino/
$ rsync -av –exclude '*.mp3' /fuente/ /destino/
$ rsync -av –exclude '*.mp3' –exclude '*.txt' /fuente/ /destino/
$ rsync -av –exclude={'*.mp3', '*.txt'} /fuente/ /destino/
```

si tenemos un *archivo.txt* con el contenido a excluir, podemos usar:

```
$ rsync -av –exclude-form={'archivo.txt'} /fuente/ /destino/
```

podemos excluir por el tamaño de los archivos, ejemplos:

```
$ rsync -av –max-size=500m /fuente/ /destino/
$ rsync -av –min-size=1m /fuente/ /destino/
```

pssh permite transferir o copiar archivos a múltiples servidores en Linux con un mismo comando:

- pscp - es una utilería para copiar archivos en paralelo a múltiples equipos
- prsync - es una utilería para transferir de forma eficiente archivos entre múltiples equipos en paralelo

- pnuke - permite concluir procesos en múltiples equipos en paralelo
- pslurp - permite copiar archivos de múltiples equipos a un equipo central en paralelo

Si creamos un archivo *Hosts.txt*, con los *IPs* como el siguiente:

```
192.168.0.3:22
192.168.0.9:22
```

podemos usar para copiar un archivo a múltiples servidores:

```
$ pscp -h Hosts.txt -l USR -Av wine-1.7.55.tar.bz2 /tmp/
```

o de forma alternativa usamos:

```
$ pscp.pssh -h Hosts.txt -l USR -Av wine-1.7.55.tar.bz2 /tmp/
```

donde:

-h indica que se lean los *IPs* del archivo indicado
-l se indica el usuario a usar en todos los equipos.
-A solicita el password para ser enviado a *ssh*
-v visualiza las operaciones y mensajes que genera el comando

Podemos copiar directorios a múltiples servidores, usando:

```
$ pscp -h myscpHosts.txt -l USR -Av -r Android\ Games/
/tmp/
```

o de forma alternativa:

```
$ pscp.pssh -h myscpHosts.txt -l USR -Av -r Android\ Games/
/tmp/
```

nc el comando *netcat* (*nc*) es una utilidad que permite escribir y leer datos a través de conexiones de red usando los protocolos TCP y UDP. Supongamos que estamos en el IP 192.168.13.230. Podemos generar un archivo compactado, cifrarlo y enviarlo a otro equipo en una mismo comando mediante:

```
$ tar -cvzf - directorio | gpg -c | nc -l 6666
```

y para recibirlo en el otro equipo usamos:

```
$ nc 192.168.13.230 6666 | gpg -d | tar -xvzf -
```

Siempre es mejor opción usar *scp*, pero *nc* cumplirá con su cometido.

5.4 Respaldo y Restauración

Una copia de seguridad (respaldo, copia de respaldo en inglés backup y data backup) es una copia de los datos originales que se realiza con el fin de disponer de un medio para recuperarlos en caso de su pérdida. Las copias de seguridad son útiles ante distintos eventos y usos: recuperar los sistemas informáticos y los datos de una catástrofe informática, natural o ataque; restaurar una pequeña cantidad de archivos que pueden haberse eliminado accidentalmente, corrompido, infectado por un virus informático u otras causas; guardar información histórica de forma más económica que los discos duros y además permitiendo el traslado a ubicaciones distintas de la de los datos originales; etc.

El proceso de copia de seguridad se complementa con otro conocido como restauración de los datos, que es la acción de leer y grabar en la ubicación original u otra alternativa los datos requeridos. La pérdida de datos es muy común en algún momento.

Ya que los sistemas de respaldo contienen por lo menos una copia de todos los datos que vale la pena salvar, se deben de tenerse en cuenta los requerimientos de almacenamiento. La organización del espacio de almacenamiento y la administración del proceso de efectuar la copia de seguridad son tareas a las que debemos dedicar tiempo y tener la certeza que están bien hechas. Para brindar una estructura de almacenamiento adecuada y segura existen muchos tipos diferentes de dispositivos físicos y en la nube para almacenar datos que son útiles para hacer copias de seguridad, cada uno con

sus ventajas y desventajas a tener en cuenta para elegirlos, como duplicidad, seguridad en los datos y facilidad de traslado.

Antes de que los datos sean enviados a su lugar de almacenamiento se lo debe seleccionar, extraer y manipular. Se han desarrollado muchas técnicas diferentes para optimizar el procedimiento de efectuar los respaldos. Estos procedimientos incluyen entre otras optimizaciones para trabajar con archivos abiertos y fuentes de datos en uso y también incluyen procesos de compresión, cifrado, y procesos de deduplicación, entendiéndose por esto último a una forma específica de compresión donde los datos superfluos son eliminados.

Tipos de Respaldos Existen distintos tipos de respaldo, entre los que destacan:

- Respaldo Completo: Se copian todos los archivos que pueda haber en una trayectoria.
- Respaldo Diferencial: Es una copia intermedia entre la completa y la incremental. Es decir, copiara tanto los archivos que se han creado nuevos como los que se han modificado.
- Respaldo Incremental: Solo copiara los archivos que hayan sido modificados tras haber hecho una copia de seguridad previa de tipo completo o diferencial. Para ello compara las fechas de modificación de los archivos de la fuente y los de la copia previa y si hay diferencias el Software tomara la decisión de copiar solo aquellos que se hayan modificado. Lo bueno de esta copia es que no es tan pesada como la completa y permite actualizar solo lo que te interesa.
- Respaldo Espejo: Este modo de respaldo es similar al respaldo completo, con la salvedad de que los archivos no se pueden comprimir para tratar de garantizar su restauración, por lo que además de ser menos segura también ocupa más espacio de almacenamiento.

Software de Copias de Seguridad Existe una gran gama de programas en el mercado para realizar copias de seguridad, es importante definir previamente los requerimientos específicos de respaldo y restauración para determinar el Software adecuado a nuestras necesidades. Existe una gran

cantidad de programas adaptados a cada necesidad, por ejemplo para respaldar toda la máquina podemos usar:

- Clonezilla
- Mondo Rescue
- Partimage
- Ping
- Redo Backup

Para hacer respaldos desde el ambiente gráfico:

- Bacula
- Amanda
- Backupninja
- Areca Backup
- BackupPC
- Keep
- UrBackup
- Back in Time
- Timeshift
- Simple Backup Solution
- KBackup
- Kup Backup System
- Grsync

Para hacer respaldos desde la línea de comandos:

- tar (véase 5.2)

- scp (véase 5.3)
- rsync (véase 5.3)
- rsnapshot
- bup
- restic
- rdiff-backup
- duplicity
- rclone

Para el adecuado respaldo de ficheros con datos de carácter personal es común que las copias de seguridad de dichos datos se almacenen cifrados y en una ubicación diferente al lugar de origen.

La copia de seguridad es el mejor método de protección de datos de importancia, pero siempre existe la posibilidad de que la copia de datos no haya funcionado correctamente y en caso de necesidad de restauración de los datos no podamos realizarlo ya que la información de la copia de seguridad puede encontrarse corrupta por diversos motivos:

- el medio en el que se realizaba la copia se encuentra dañado.
- los automatismos de copia no se han ejecutado correctamente.
- y otros muchos motivos que pueden causar que nuestras copias de seguridad sean incorrectas, y por lo tanto inútiles.

Para evitar este problema es muy importante que nos cercioremos de que hacemos las copias correctamente y comprobemos que somos capaces de restaurar la copia de seguridad a su ubicación original, comprobando así que la copia sea correcta y que somos capaces de restaurarla y conocemos el método de restauración, ya que en caso de necesidad crítica los nervios afloran y nos pueden echar por tierra nuestra labor de copia al realizar algún paso erróneo a la hora de restaurar los datos.

Siempre que podamos debemos cifrar los respaldos, esto los mantiene más seguros. Si alguien llegara a tener acceso a datos de respaldo sin el debido cifrado, este se podría restaurar y con ello podría utilizar toda la información del mismo.

rdiff-backup es capaz de realizar una copia casi exacta del directorio origen (sin cifrar), preservando toda la información de los recursos: permisos, usuarios y grupos, fecha de modificación, enlaces simbólicos, ficheros fifos, etc., manteniéndola independientemente de la plataforma y sistema de ficheros donde se almacene, gracias a que toda esa información se guarda en ficheros independientes de metadatos. Además seremos capaces de recuperar una instantánea exacta de un determinado día, restaurando el estado que en ese momento tenía nuestro directorio (fichero eliminados, modificados, etc.).

Otra característica importante es la eficiencia del espacio usado. Los respaldos incrementales que utilizan herramientas como *backup-manager*, realizan una copia completa de los recursos modificados, en cambio, *rdiff-backup* sólo almacena las fracciones de datos que realmente han cambiado.

El respaldo de recursos remotos es otro de sus puntos fuertes y que considero importante destacar. La transferencia de recursos remotos esta optimizada; sólo se transfiere por la red la información que realmente ha cambiado, haciendo un uso eficiente del ancho de banda. El único requisito es que en la máquina remota también se encuentre instalado *rdiff-backup*. Para su instalación hacemos:

```
# apt install rdiff-backup
```

Realizar copias de seguridad con *rdiff-backup* es realmente sencillo, es como si estuviéramos usando el comando *cp* de Linux. Únicamente tendremos que indicar el directorio origen (del que se quiere realizar el respaldo) y el directorio destino (donde se almacenará el respaldo).

```
$ rdiff-backup dir_ori dir_dest
```

Podemos encontrarnos varios escenarios posibles dependiendo de donde se encuentren situados los directorios origen y destino.

- directorio origen local y destino local
- directorio origen remoto y destino local
- directorio origen local y destino remoto

Independientemente de cada caso su uso es idéntico, sólo cambia la forma de acceder a los directorios remotos.

Por ejemplo, el comando que deberíamos lanzar para realizar un respaldo de nuestro directorio local /home/autentia al directorio destino /mnt/backup situado en la misma máquina sería:

```
$ rdiff-backup /home/autentia /mnt/backup
```

Si alguno de los dos directorios estuviese en una máquina remota deberíamos hacer preceder al directorio del usuario y el nombre de la máquina. Por ejemplo, vamos a imaginar que queremos realizar el respaldo del directorio /etc de la maquina mac1 (utilizando el usuario backup para acceder a la máquina) al directorio /mnt/backup/mac1 de nuestra máquina local. El comando que deberíamos ejecutar sería:

```
$ rdiff-backup backup@mac1::/etc /mnt/backup/mac1
```

Obviamente, para que este comando funcione en modo Script, deberemos tener configurada correctamente nuestra máquina para acceder al otro equipo sin necesidad de contraseña.

Para restaura el directorio home de la copia realizada el 10 de Enero de 2021 en /tmp/home.

```
$ rdiff-backup -r 2021-01-10 /mnt/backup/home /tmp/home
```

Otro ejemplo, restauramos la copia actual situada en la máquina remota mac1 en el directorio /temp.

```
$ rdiff-backup -r now backup@mac1::/mnt/backup /temp
```

podremos obtener el listado con los cambios incrementales de un fichero:

```
$ rdiff-backup -l /mnt/backup/file
```

Con -list-changed-since podremos saber cuántos ficheros han cambiado desde una marca de tiempo en concreto. Por ejemplo con el siguiente comando estamos viendo que ficheros han cambiado en los últimos 10 días.

```
$ rdiff-backup -list-changed-since 10D /mnt/backup/etc
```

-list-at-time lista todos los ficheros que estuvieron presentes en un determinado momento. Esto incluye tanto fichero eliminados como aquellos que no han sido modificados.

```
$ rdiff-backup -list-at-time 5D /mnt/backup/home
```

Y por último podremos comparar los cambios producidos entre el resplado y un directorio en concreto. rdiff-backup proporciona dos opciones -compare y -compare-at-time; este último para poder comparar con una marca de tiempo determinada. Por ejemplo:

```
$ rdiff-backup -compare /home /mnt/backup/home
$ rdiff-backup -compare-at-time 2W /home /mnt/backup/home
```

duplicity es una herramienta avanzada de copias de seguridad cifradas, disponible para la línea de comandos. Está escrita en *Python*, usando herramientas como *librsync* y *GnuPG*.

Los archivos obtenidos se encuentran en formato *tar* y, si lo creemos oportuno, pueden estar o no cifrados. Además, permite realizar copias incrementales que nos permite ahorrar espacio. Para su instalación hacemos:

```
# apt install duplicity
```

Para hacer una copia de seguridad de los datos que queramos, solo hay que utilizar el nombre de la herramienta, seguido de la ruta de los datos que queremos copiar y, a continuación, del destino donde queremos almacenarla. Es decir, algo como esto:

```
$ duplicity ~/documentos file:///backup
```

la recuperación usando esa carpeta como destino. Para lograrlo, escribimos algo como esto:

```
$ duplicity file:///backup recupera
```

Como cabía esperar, la herramienta nos pide la frase de paso para cifrar antes de iniciar la recuperación.

A continuación, vamos a repetir la copia anterior, pero evitando que se copie el directorio borradores y todo su contenido. Lo logramos añadiendo la opción -exclude, seguida de la ruta a excluir

```
$ duplicity ~/documentos file:///backup –exclude ~/documentos/borradores
```

Podemos usar duplicity en un Scrip, por ejemplo:

```
export PASSPHRASE='ClaveCifrado'
export FTP_PASSWORD='ClaveServidor'
## Respalda localmente
duplicity /home/antonio/trabajo file:/home/antonio/Respaldos
## Verifica
duplicity verify file:/home/antonio/Respaldos /home/antonio/trabajo
## Respalda en el servidor
duplicity /home/antonio/trabajo scp://antonio@\
192.168.13.230//home/antonio/Respaldos
unset PASSPHRASE
unset FTP_PASSWORD
```

rclone es una aplicación libre para sistemas de tipo Linux, Unix y Microsoft Windows, se trata de una herramienta en línea de comando para sincronizar archivos y directorios desde la computadora con los proveedores más importantes de alojamiento de contenidos en la nube[92]. También permite efectuar copias dentro de nuestro propio sistema de archivos. Está escrito en lenguaje de programación *Go* y se basa en la utilidad *rsync*. Para instalarlo hacemos:

```
# apt install rclone
```

Para crear, editar o eliminar un dispositivo remoto, es necesario utilizar la siguiente orden

```
$ rclone config
```

Se entra a una sesión interactiva, en la que configuraremos a *rclone* con los datos de nuestra cuenta en la nube, especificando el nombre del recurso mediante el cual podremos identificar nuestro servicio de la nube (por ejemplo antonio_google) en nuestra máquina.

Por ejemplo, podemos conocer el monto de espacio disponible en el servidor, usando:

```
$ rclone about antonio_google:
```

[92] Algunos de los servicios en la nube más importantes soportados por Rclone son: Amazon Drive, Amazon S3, Box, Dropbox, Google Cloud Storage, Google Drive, Hubic, Mega, Microsoft OneDrive, Nextcloud, OpenDrive, Oracle Cloud Storage, ownClowd, pCloud, QingCloud Object Storage, Webdav, Yandex Disk.

listamos el contenido de carpetas y contenedores del servidor en la nube:

```
$ rclone lsd antonio_google:
```

podemos crear un directorio para hacer la sincronización:

```
$ rclone mkdir antonio_google:Respaldos
```

y realizamos la sincronización del contenido de nuestro equipo con el de la nube, usamos:

```
$ rclone sync /home/antonio/trabajo/ antonio_google:Respaldos
-P
```

si necesitamos sincronizar el contenido de la nube con nuestro equipo, usamos:

```
$ rclone sync antonio_google:Respaldos /home/antonio/trabajo/
-P
```

Podemos sincronizar el contenido de una unidad o directorio, por ejemplo en el directorio *nube*, usando:

```
$ rclone mount antonio_google:Respaldos ./nube
```

este permanecerá montado hasta que usemos Ctrl-C para terminar el comando rclone, en caso de ser necesario desmontar el punto de montaje manualmente, usamos:

```
$ fusermount -u ./nube
```

Algunas otras opciones de trabajo son:

- rclone config - crear, editar o eliminar un dispositivo remoto.
- rclone copy - Copiar archivos del origen al destino.
- rclone sync - Sincronizar archivos, modificando el destino únicamente.
- rclone move - Copiar archivos del origen al destino.

- rclone delete - Borrar archivos.
- rclone purge - Borrar todo el contenido de la carpeta.
- rclone mkdir - Crear una carpeta si no existe.
- rclone rmdir - Borrar una carpeta.
- rclone rmdirs - Borrar carpetas vacías.
- rclone check - Comprueba si los archivos en el origen y destino coinciden.
- rclone ls - Lista todos los archivos.
- rclone lsd - Lista todas las carpetas y contenedores.
- rclone size - Devuelve el tamaño y el número de objetos.
- rclone version - Muestra el número de versión.
- rclone authorize - Autorización remota.
- rclone cat - Concatena archivos y los envía a la salida estándar.
- rclone copyto - Copiar archivos del origen al destino, exceptuando los ya copiados.
- rclone listremotes - Lista todos los dispositivos remotos.
- rclone moveto - Mueve archivos o carpetas desde el origen al destino.
- rclone about - Devuelve información sobre el dispositivo remoto.

5.5 Incrementando la Seguridad a Nivel Usuario

La mejor opción, es elegir una distribución de GNU/Linux que nos permita mantener el sistema actualizado, instalar sólo los paquetes que necesitamos y que estos provengan de una fuente confiable, además de cifrar las particiones del sistema operativo y de datos del usuario.

Borrado de Archivos o Particiones de Forma Segura La herramienta *shred* ("ha-cer trizas" en español) -provista por el paquete *coreutils* instalado por omisión-, permite sobrescribir un archivo o dispositivo para ocultar y eliminar todo su contenido, sin dejar rastro alguno. A fin de que sea imposible de recuperar para herramientas de Hardware que hacen análisis magnético de sectores[93], *shred* sobrescribe repetidamente el contenido con valores aleatorios, haciendo un uso intensivo de disco[94]. A su vez permite eliminar el archivo al finalizar.

Por ejemplo, se procede a sobreescribir y borrar el archivo con *shred*:

```
$ shred -n 5 -u -v -z miArchivo.dat
```

las opciones utilizadas son las siguientes:

-n 5: 5 iteraciones de sobrescritura.

-u: borrar el archivo al finalizar.

-v: muestre el progreso.

-f: fuerza el cambio de permisos para permitir la escritura de ser necesario.

-z: rellenar con ceros al final para ocultar el propio Shredding.

Se observa que en cada pasada se alterna entre rellenado con ceros, con datos aleatorios y con unos (ffffff). Esto se hace para maximizar la efectividad del borrado o Shredding, de forma que todos los Bits resulten modificados al menos una vez. Finalmente se renombra y se borra. Esto ocurre para hacer el Shredding del propio nombre del archivo en la entrada de directorio.

Lo podemos usar en múltiples archivos a la vez, mediante:

[93] Es importante aclarar que esta herramienta se basa en la suposición importante, de que el sistema de archivos sobrescribe los datos en el lugar. Generalmente es así, pero muchos sistemas de archivos modernos no lo respetan. Por ejemplo los sistemas de archivos con Journal o estructurados en Log (por ejemplo *ext4*); sistemas de archivos que escriben datos redundantes como los esquemas *RAID*; sistemas de archivos que soportan Snapshots (*LVM* o *ZFS*); sistemas de archivos que hacen un uso intensivo de Cache en locaciones temporales (*NFS*); sistemas de archivos comprimidos; etc.

En tales casos *shred* no es efectivo, o no se garantiza que lo sea.

Para el caso de los sistemas de archivos *ext3* y *ext4* esta advertencia aplica sólo si el Journal está habilitado tanto para los datos como metadatos (modo data=journal). En los modos data=ordered (por defecto) y data=writeback, *shred* funciona correctamente.

[94] No es recomendado su uso en discos SSD por el desgaste acumulativo que provoca al mismo.

```
$ shred -u -v *.txt
```

o en una partición del disco (/dev/sda2/), mediante:

```
# shred -vfz /dev/sda2
```

También es posible usar el paquete *wipe*, para instalarlo usamos

```
# apt install wipe
```

y lo usamos mediante:

```
$ wipe archivo
$ wipe -r -q directorio/*
# wipe /dev/sda2
```

o el paquete *secure-delete*, para instalarlo usamos:

```
# apt install secure-delete
```

este paquete consta de cuatro herramientas, a saber:

- srm - utilizado para borrar archivos o directorios que se encuentren en disco

  ```
  $ srm archivo
  $ srm -r directorio/*
  ```

- sdmem - utilizado para limpiar restos de información en la memoria (RAM) de la máquina

  ```
  # sdmem -f -v
  ```

- sfill - utilizado para limpiar los restos de información del espacio vacío del disco

  ```
  # sfill -v /home/usuario
  ```

- sswap - utilizado para limpiar los restos de información de la partición Swap, para conocerla usamos:

  ```
  $ swapon
  ```

  supongamos que la partición Swap es: /dev/sda2, entonces:

  ```
  # swapoff /dev/sda2
  # sswap /dev/sd2
  # swapon /dev/sda2
  ```

Cifrado de Discos y Particiones Al momento de instalar el sistema operativo una buena práctica de seguridad es solicitar que use particiones cifradas para nuestros datos y el sistema operativo. Esto permite mantener a nuestros datos lejos de miradas indiscretas. Además cada disco externo o unidad Flash se recomienda cifrar, esto es posible al formatear el dispositivo -casi todos los escritorios de GNU/Linux tienen integrada una opción para ello-, así en caso de perdida o robo del dispositivo nuestros datos permanecen ilegibles para cualquiera.

En caso que nuestro escritorio no tenga instalada una opción gráfica para formatear y cifrar dispositivos, podemos usar GNOME Disks (también conocida como *gnome-disk-utility*, o *GNOME Disks*) es una aplicación gráfica de *udisks* incluido en el paquete *gnome-disk-utility*. Se puede utilizar GNOME Disk para la gestión de particiones (cifrar), motorización de tipo *SMART*, evaluación comparativa y Software *RAID*.

Para instalar usamos:

```
# apt install gnome-disk-utility
```

y para usar:

```
# gnome-disks
```

esto nos permite entre otras cosas al formatear una unidad y cifrarla.

Cifrar Discos y Particiones con cryptsetup es uno de los Softwares de cifrado de disco más usado. Este se encarga de cifrar/descifrar los datos escritos en un dispositivo de almacenamiento. El cifrado a nivel de disco garantiza que los archivos siempre se almacenen en el disco de forma cifrada. Los archivos solo están disponibles para poder ser leídos mientras el sistema está en ejecución y desbloqueado por uno de los usuarios con acceso a la clave de descifrado. Una persona no autorizada que intente acceder al contenido del disco, solo encontrará datos confusos, en lugar de los archivos reales.

Todos los métodos de cifrado de disco funcionan de tal manera que, aunque el disco realmente contiene datos cifrados, el sistema operativo lo ve como los datos legibles normales, siempre que el contenedor cifrado (es decir, la parte del disco que contiene los datos cifrados) ha sido desbloqueado y montado en alguna partición del sistema.

Instalar Paquete usamos:

```
# apt install cryptsetup
```

Es bueno conocer el rendimiento del programa *cryptsetup* para los diferentes tipos de cifrado soportado, para ello usamos:

```
# cryptsetup benchmark
```

algunos de ellos son:

> PBKDF2-sha1, PBKDF2-sha256, PBKDF2-sha512, PBKDF2-ripemd160, PBKDF2-whirlpool, argon2i, argon2id, aes-cbc 128, serpent-cbc 128, twofish-cbc 128, aes-cbc 256, serpent-cbc 256, twofish-cbc 256, aes-xts 256, serpent-xts 256, twofish-xts 256, aes-xts 512, serpent-xts 512, twofish-xts 512

esto nos permitirá elegir el algoritmo óptimo para nuestro equipo, que nos de el mejor rendimiento en el cifrado y mayor seguridad posible.

El primer paso es escoger la unidad de disco que vamos a cifrar, podemos usar el comando *lsblk* para obtener una lista de dispositivos de bloque montados en el equipo de cómputo, mediante:

```
$ lsblk
```

supongamos que trabajaremos con: */dev/sdb1*.

Crear partición cifrada

```
# cryptsetup -v -y -c aes-xts-plain -s 512 luksFormat /dev/sdb1
# cryptsetup luksOpen /dev/sdb1 Respaldos
# ls -l /dev/mapper/Respaldos
# cryptsetup -v status Respaldos
```

Formatear partición

```
# mkfs.ext4 /dev/mapper/Respaldos
# mkdir -p /opt/Respaldos_crypt
# mount /dev/mapper/Respaldos /opt/Respaldos_crypt
# df -Th | grep crypt
# cd /opt/Respaldos_crypt
```

Montar y activar sistema de archivos

```
# cryptsetup luksOpen /dev/sdc1 Respaldos
# mkdir -p /opt/Respaldos_crypt
# mount /dev/mapper/Respaldos /opt/Respaldos_crypt
# df -Th | grep crypt
# cd /opt/Respaldos_crypt
```

Desmontar y desactivar el sistema de archivos

```
# umount /opt/Respaldos_crypt
# cryptsetup luksClose Respaldos
```

El uso del dispositivo cifrado en la gran mayoría de las distribuciones de GNU/Linux que tengan instalado el paquete *cryptsetup* reconocerán el dispositivo al momento de ser introducido en el equipo de cómputo y solicitarán la clave de acceso. A partir de ese momento será posible usar el dispositivo como cualquier otro -sin cifrado- y todas las tareas de montaje/desmontaje serán transparentes para el usuario.

Podemos usar el comando fsck en sistemas basados en LUKS, mediante:

```
# umount /opt/Respaldos_crypt
# fsck -vy /dev/mapper/Respaldos
# mount /dev/mapper/Respaldos /opt/Respaldos_crypt
```

Para cambiar la contraseña para la partición cifrada, usamos:

```
# cryptsetup luksDump /dev/sdb1
# cryptsetup luksAddKey /dev/sdb1
```

se pueden configurar un máximo de 8 contraseñas para cada dispositivo
Remover o eliminar la contraseña:

```
# cryptsetup luksRemoveKey /dev/sdb1
```

tengamos en cuenta que debemos ingresar la contraseña guardada.

Cifrado Basado en Archivos con gocryptfs Podemos tratar de mantener nuestros datos fuera de miradas indiscretas usando gocryptfs en las máquina a las que tengamos acceso (incluso del usuario administrador *root)*[95]. Además, podemos respaldarlos y transportarlos manteniendo estos siempre cifrados y que sea casi transparente para nosotros el uso de dicho programa.

El programa *gocryptfs* utiliza cifrado basado en archivos que se implementa como un sistema de archivos *FUSE* que se puede montar. Cada archivo en *gocryptfs* se almacena como un archivo cifrado correspondiente en el disco. Los archivos cifrados se pueden almacenar en cualquier carpeta del disco, unidad *USB* o incluso dentro de una carpeta en la nube como *Google Drive* o *Dropbox*. Una ventaja del cifrado basado en archivos en comparación con el cifrado de disco es que los archivos cifrados de pueden sincronizar de manera eficiente utilizando herramientas como rsync. Además, el tamaño del sistema de archivos cifrado es dinámico y solo está limitado por el espacio disponible en disco.

Para instalarlo usamos:

```
# apt install gocryptfs
```

después, es necesario crear los directorios para guardar los datos cifrados (por ejemplo en el directorio ~/.cifrados) y los descifrados (por ejemplo en el directorio ~/descifrados), mediante:

```
$ mkdir -p ~/.cifrados ~/descifrados
```

Para inicializar la carpeta usamos:

```
$ gocryptfs -init ~/.cifrados
```

nos pedirá la clave de acceso y su confirmación. Se pueden crear tantas carpetas independientes con *gocyptfs* como se requieran y el cualquier parte del sistema de archivos al que tengamos acceso.

Ahora ya podemos montar el directorio mediante:

```
$ gocryptfs ~/.cifrados ~/descifrados
```

[95] Existen múltiples proyectos -algunos multiplataforma- que permiten hacer lo mismo que *gocryptfs* como son: *encfs, ecrptfs, cryptomator, securefs y CryFS*, entre otros.

el sistema nos pedirá nuestra clave de acceso y de ser correcta nos permitirá hacer el montaje. De esta forma, la carpeta virtual *~/cifrados* mostrará nuestros datos descifrados, solo visibles para el usuario que monta la carpeta (no para root) y la carpeta *~/.cifrados*, contendrá nuestros datos de forma cifrada[96], esta carpeta puede ser copiada, respaldada y restaurada aún estando montada, manteniendo el anonimato de nuestros archivos.

Una vez que ya no sea requerida la carpeta, la desmontamos usando:

```
$ fusermount -u ~/descifrados
```

Es posible usar *gocryptfs* en modo inverso, primero debemos inicializar la carpeta usando:

```
$ gocryptfs -reverse -init ~/.descifrados
```

y usarla mediante:

```
$ gocryptfs -reverse ~/.descifrados ~/.cifrados
```

Cifrar Archivos con GnuPG es una herramienta en línea de comandos de seguridad en comunicaciones electrónicas en donde se utiliza criptografía de clave pública para que los usuarios puedan comunicarse de un modo seguro. En un sistema de claves públicas cada usuario posee un par de claves, compuesto por una clave privada y una clave pública. Cada usuario debe mantener su clave privada secreta; no debe ser revelada nunca. La clave pública se puede entregar a cualquier persona con la que el usuario desee comunicarse. GnuPG implementa un esquema algo más sofisticado en el que un usuario tiene un par de claves primario, y ninguno o más de un par de claves adicionales subordinadas. Los pares de claves primarios y subordinados se encuentran agrupados para facilitar la gestión de claves, y el grupo puede ser considerado como un sólo par de claves.

Dentro de las funciones de GnuPG se incluyen generar un par de claves, intercambiar y comprobar la autenticidad de claves, cifrar y descifrar documentos, etc. Para instalar el paquete GnuPG, usamos:

```
# apt install gnupg
```

[96] El nombre de los archivos tienen un límite, por ello hay que tenerlo en cuenta si se usan nombre largos en el sistema de archivos (el límite aproximado es 255 caracteres), pero no importa el número de archivos o carpetas almacenadas internamente.

La opción de la línea de comandos *–gen-key* se usa para generar un nuevo par de claves primario, mediante:

```
$ gpg –gen-key
```

también podemos pedir –full-generate-key, mediante:

```
$ gpg –full-generate-key
```

GnuPG es capaz de crear varios tipos diferentes de pares de claves, pero debe existir una clave primaria capaz de generar firmas. Al momento de ejecutar el programa, este pide[97] nombre, correo del usuario y contraseña[98], para después generar la clave.

Para poder comunicarse con otros, el usuario debe intercambiar las claves públicas. Para obtener una lista de las claves en el fichero («anillo») de claves públicas, se puede usar la opción de la línea de comandos *–list-keys*, mediante:

```
$ gpg –list-keys
```

también podemos usar:

```
$ gpg –list-secret-keys
$ gpg –list-public-keys
```

Para poder enviar una clave pública a un interlocutor, antes hay que exportarla. Para ello se usará la opción de la línea de comandos *–export*. Es necesario un argumento adicional para poder identificar la clave pública que se va a exportar, por ejemplo:

```
$ gpg –output antonio.gpg –export ant@na.mx
```

[97] Según la versión de GnuPG puede solicitar otros datos como: el tipo de clave, longitud, cuando expira está, entre otros posibles datos de configuración.

[98] Cuanto más larga sea la contraseña, más segura será contra ataques de «fuerza bruta». No hay límite para la longitud de una contraseña, y esta debe ser escogida con sumo cuidado. Desde un punto de vista de seguridad, la contraseña que desbloquea la clave privada es uno de los puntos más débiles en GnuPG (y en otros sistemas de cifrado de clave pública), ya que es la única protección que tiene el usuario si alguien se apodera de su clave privada. Para una contraseña lo ideal es que no se usen palabras de un diccionario, y que se mezclen mayúsculas y minúsculas, dígitos, y otros caracteres. Una buena contraseña es crucial para el uso seguro de GnuPG.

La clave se exporta en formato binario, y esto puede no ser conveniente cuando se envía la clave por correo electrónico o se publica en una página Web. Por tanto, GnuPG ofrece una opción de la línea de comandos *–armor* que fuerza que la salida de la orden sea generada en formato *armadura-ASCII*, parecido a los documentos codificados con *UUEncode*. Por regla general, cualquier salida de una orden de GnuPG, v.g. claves, documentos cifrados y firmas, pueden ir en formato *armadura-ASCII* añadiendo a la orden la opción *–armor*, por ejemplo:

$ gpg –armor –output antonio.gpg –export ant@na.mx

Cada clave pública y privada tiene un papel específico en el cifrado y descifrado de documentos. Se puede pensar en una clave pública como en una caja fuerte de seguridad. Cuando un remitente cifra un documento usando una clave pública, ese documento se pone en la caja fuerte, la caja se cierra, y el bloqueo de la combinación de esta se gira varias veces. La parte correspondiente a la clave privada, esto es, el destinatario, es la combinación que puede volver a abrir la caja y retirar el documento. Dicho de otro modo, sólo la persona que posee la clave privada puede recuperar un documento cifrado usando la clave pública asociada al cifrado.

Con este modelo mental se ha mostrado el procedimiento de cifrar y descifrar documentos de un modo muy simple. Si el usuario quisiera cifrar un mensaje para Javier, lo haría usando la clave pública de Javier, y lo descifraría con su propia clave privada. Si Javier quisiera enviar un mensaje al usuario, lo haría con la clave pública del usuario, y este lo descifraría con su propia clave privada.

Cifrar un Archivo para cifrar un archivo se usa la opción *–encrypt*. El usuario debe tener las claves públicas de los pretendidos destinatarios. El programa espera recibir como entrada el nombre del archivo que se desea cifrar o, si este se omite, una entrada estándar. El resultado cifrado se coloca en la salida estándar o donde se haya especificado mediante la opción *–output*. El archivo se comprime como medida adicional de seguridad, aparte de cifrarlo, por ejemplo:

$ gpg –output doc.gpg –encrypt –recipient ant@na.mx doc

La opción *–recipient* se usa una vez para cada destinatario, y lleva un argumento extra que específica la clave pública con la que sería cifrado el

archivo. El archivo cifrado sólo puede ser descifrado por alguien con una clave privada que complemente una de las claves públicas de los destinatarios. El usuario, en este caso el remitente, no podría descifrar un archivo cifrado por sí mismo a menos que haya incluido su propia clave pública en la lista de destinatarios.

Descifrar un Archivo para descifrar un archivo se usa la opción –*decrypt*. Para ello es necesario poseer la clave privada para la que el archivo ha sido cifrado. De igual modo que en el proceso de cifrado, el archivo a descifrar es la entrada, y el resultado descifrado la salida, por ejemplo:

```
$ gpg –output doc –decrypt doc.gpg
```

Cifrar y Descifrar Usando Cifrado Simétrico también es posible cifrar archivos sin usar criptografía de clave pública. En su lugar, se puede usar sólo una clave de cifrado simétrico para cifrar el archivo. La clave que se usa para el cifrado simétrico deriva de la contraseña dada en el momento de cifrar el documento, y por razones de seguridad, no debe ser la misma contraseña que se esta usando para proteger la clave privada. El cifrado simétrico es útil para asegurar archivos cuando no sea necesario dar la contraseña a otros. Un archivo puede ser cifrado con una clave simétrica usando la opción –*symmetric*, por ejemplo:

```
$ gpg –symmetric doc
```

o

```
$ gpg -c doc
```

y podemos descifrar usando:

```
$ gpg doc.gpg
```

Otras Opciones para Cifrar y Descifrar

ccrypt es otra herramienta para cifrar y descifrar archivos basada en el cifrado de bloque Rijndael, Se cree que *ccrypt* (basada en el cifrado de bloque *Rijndael*) proporciona una seguridad fuerte. Tengamos en cuenta que *ccrypt* elimina el archivo original (cifra el archivo en su lugar), no cambia significativamente el tamaño del archivo y no altera la fecha/hora del archivo para reflejar la hora en que se realizó el cifrado.

```
$ ccrypt -e archivo
```

para descifrar usamos:

```
$ ccrypt -d archivo.cpt
```

mcrypt el comando permite cifrar y descifrar archivos, cuando cifra deja el archivo original intacto y cambia los permisos del archivo para que el archivo cifrado solo proporcione permisos de acceso de lectura y escritura al propietario del archivo. Ofrece muchas opciones con respecto a los algoritmos de cifrado y también ofrece opciones para comprimir los archivos antes del cifrado (opciones -z comprime GZip y -p comprime Bz2). Puede funcionar con varios archivos, pero los cifra por separado, ejemplo:

```
$ mcrypt archivo
```

para descifrar usamos:

```
$ mcrypt -d archivo.nc
```

para enumerar los algoritmos de cifrado disponibles, usamos:

$ *mycrypt –list*,

El comando *mcrypt* parece usar *Rijndael-128* como su algoritmo de cifrado predeterminado.

age este comando permite cifrar usando una clave pública o frase, no viene instalado por omisión en el sistema, pero lo podemos instalar usando:

```
# apt install age
```

Para generar una clave pública en el archivo llave.txt usamos:

```
$ age-keygen -o llave.txt
```

Para cifrar un archivo *archivo.tar* con clave pública usamos:

```
$ tar cvz archivos | age -r llave.txt > archivo.tar.gz.age
```

para descifrarlo usamos:

```
$ age –decrypt -i llave.txt > archivo.tar.gz
```

que nos retornará a nuestro archivo original.
Para cifrar un archivo *archivo.tar* con frase usamos:

```
$ age –passphrase –output archivo.tar.gz.age archivo.tar.gz
```

para descifrarlo usamos:

```
$ age –decrypt –output archivo.tar.gz archivo.tar.gz.age
```

que nos retornará a nuestro archivo original.

Generar Contraseñas Para generar contraseñas disponemos de múltiples opciones, algunas de ellas son:

- Podemos usar GnuPG, para instalarlo usamos:

  ```
  # apt install gnupg
  ```

  para generar una contraseña de 32 caracteres, usamos:

  ```
  $ gpg –gen-random –armor 1 32
  ```

- Podemos usar OpenSSL, para instalarlo usamos:

```
# apt install openssl
```

para generar una contraseña de 32 caracteres mediante:

```
$ openssl rand -base64 32
```

- Podemos usar APG, para instalarlo usamos:

```
# apt install apg
```

para generar una contraseña de mínimo 31 y máximo 32 caracteres y genere 4 claves:

```
$ apg -n 4 -m 31 -x 32 -a 1
```

- Podemos usar PWGEN, para instalarlo usamos:

```
# apt install pwgen
```

para generar una contraseña de 32 caracteres:

```
$ pwgen -ys 32 1
```

- Podemos usar Makepasswd, para instalarlo usamos:

```
# apt install makepasswd
```

para generar una contraseña de 32 caracteres:

```
$ makepasswd –char 32
```

Cifrado Avanzado En los procesadores modernos es común encontar el motor Intel/AMD Advanced Encryption Standard (AES) o New Instructions (AES-NI) que permite el cifrado y descifrado por Hardware de alta velocidad para el cifrado de disco completo, OpenSSL, ssh, VPN, Linux / Unix / OSX y más. ¿Cómo verifico la compatibilidad con Intel/AMD AES-NI en mi sistema basado en Linux, incluido OpenSSL?

El conjunto de instrucciones del estándar de cifrado avanzado y las nuevas instrucciones del estándar de cifrado avanzado permiten que Intel/AMD específicas y otras CPU realicen un cifrado y descifrado de hardware extremadamente rápido.

Tengamos en cuenta que la compatibilidad con AES-NI se habilita automáticamente si el procesador detectado lo soporta. Para obtener una lista de procesadores que admiten el motor AES-NI, consulte el sitio y la documentación del procesador. El AES-NI es una extensión de la arquitectura del conjunto de instrucciones x86 para microprocesadores de Intel y AMD. Aumenta la velocidad de las aplicaciones que realizan el cifrado y el descifrado mediante AES.

Uno puede descubrir que el procesador tiene el conjunto de instrucciones AES / AES-NI usando el comando:

```
$ lscpu
```

o

```
$ grep -o aes / proc / cpuinfo
```

o

```
$ grep -m1 -o aes / proc / cpuinfo
```

o

```
# cpuid | grep -i aes | sort | uniq
```

En cualquiera de los casos si aparece la bandera *aes* el conjunto de instrucciones AES / AES-NI está presente y activo.

También podemos usar el siguiente comando para verificar la compatibilidad con AES-NI en nuestro procesador:

```
$ sort -u /proc/crypto | grep module
```

Ahora que hemos verificado la compatibilidad, es hora de probarla, para ello usamos:

```
$ openssl engine
```

ejemplo de una salida que soporta AES:

```
(padlock) VIA PadLock (no-RNG, no-ACE)
(dynamic) Dynamic engine loading support
```

ejemplo de una salida que soporta AES-NI:

(aesni) Intel AES-NI engine
(dynamic) Dynamic engine loading support

y podemos probar su desempeño entre un equipo que soporte el cifrado:

$ dd if=/dev/zero count=1000 bs=1M | ssh -l usuario -c aes128-cbc maquina "cat >/dev/null"

1000+0 records in

1000+0 records out

1048576000 bytes (1.0 GB) copied, 10.6691 s, 98.3 MB/s

y uno que no la soporte:

$ dd if=/dev/zero count=1000 bs=1M | ssh -l usuario -c aes128-cbc maquina "cat >/dev/null"

1000+0 records in

1000+0 records out

1048576000 bytes (1.0 GB) copied, 31.6675 s, 33.1 MB/s

¿Cómo puedo comparar el rendimiento de OpenSSL? podemos correr los comandos:

$ openssl speed

o

$ openssl speed aes-128-cbc

Para la última versión de OpenSSL, podemos probar los dos comandos, el segundo comando debería tener números más altos que el primero gracias a EntropyZer0:

$ openssl speed aes-256-cbc
$ openssl speed -evp aes-256-cbc

5.6 AntiVurus

Aunque es poco probable que un virus afecte a un sistema Linux, es posible que sea un vector de transmisión a través de los servicios de E-Mail o del servidor de archivos, por ejemplo. Además, estos mismos servicios pueden integrar un antivirus, lo que contribuye a aumentar la seguridad de la red local.

Tenemos varias opciones pero por múltiples razones preferimos *ClamAV* y de ser requerida su interfaz gráfica ClamTK, para su instalación mínima podemos usar:

```
# apt install clamav clamav-freshclam
```

para que *ClamAV* también pueda verificar archivos comprimidos, debemos instalar algunos paquetes para descomprimir archivos:

```
# apt install arc arj bzip2 cabextract lzop nomarch p7zip \
pax tnef unrar-free unzip
```

Si tenemos acceso a los repositorios "non-free", podemos instalar otros paquetes adicionales:

```
# apt install lha unrar
```

La actualización de la base de datos de firmas de virus es descargada de internet, después de la instalación y antes de usar el antivirus, debemos realizar la actualización de la base de datos de firmas de virus, usando:

```
# freshclam
```

y podemos revisar la ruta que deseemos:

```
# clamscan /home/antonio/Descargas/
```

La distribución Debian ofrece un paquete de archivos de prueba "infectados" con la firma de un virus falso. *ClamAV* debe ser capaz de identificar correctamente estos archivos en la prueba. Para instalarlo usamos:

```
# apt install clamav-testfiles
```

y podemos efectuar la prueba, mediante:

```
# clamscan /usr/share/clamav-testfiles/
```

Si así lo queremos, podemos eliminar el paquete de pruebas:

```
# apt remove clamav-testfiles
```

También, podemos hacer la instalación completa y usarlo como demonio:

```
# apt install clamav clamav-docs clamav-daemon clamav-
freshclam
# apt install arc arj bzip2 cabextract lzop nomarch p7zip \
pax tnef unrar-free unzip
# apt install lha unrar
```

La actualización de la base de datos de firmas de virus es descargada de internet por el demonio *clamav-freshclam* 24 veces al día. Esta frecuencia puede modificarse en el archivo: */etc/clamav/freshclam.conf*

y debemos reiniciar el servicio, para que tenga en cuenta las alteraciones de configuración:

```
# service clamav-freshclam restart
```

También se puede probar el demonio *clamdscan*:

```
# clamdscan /usr/share/clamav-testfiles/
```

ahora el antivirus está listo para ser usado manualmente y ser integrado en otros sistemas y servicios.

Para la detección de virus, puede utilizarse los comandos *clamscan* y *clamdscan*. Pero, la segunda forma *clamdscan* es mucho más veloz, porque al ser un demonio, ya está presente en la memoria. Al contrario, cuando se ejecuta el comando *clamscan*, el sistema primero lee el disco.

5.7 Programando en Bash

Un lenguaje de programación interpretado es aquel que no necesita ser compilado para ejecutarse, sino que se puede ejecutar directamente desde el código fuente usando un intérprete, que no es más que un programa que puede traducir el código a unas instrucciones comprensibles por la máquina. Esto aporta algunas ventajas:

- Multiplataforma: al no ser binario, se pueden ejecutar en diversas plataformas sin modificaciones, lo que es una clara ventaja si queremos que el código funcione en cualquier sistema.
- Portabilidad: si el intérprete está listo para una plataforma, entonces el Script o lenguaje interpretado funcionará en dicha plataforma.

Sin embargo, estos lenguajes interpretados también tienen sus desventajas:

- Una de ellas es el rendimiento, ya que necesitan del intérprete siempre ejecutándose en segundo plano para que funcione.
- La propia dependencia del intérprete.

Como ejemplo de lenguajes interpretados se pueden citar algunos como Bash, Java, C#, JavaScript, Visual Basic .NET y VBScript, Perl, Python, Lips, Ruby, PHP, ASP, etc.

Un Script Bash no es más que un código creado con un lenguaje de programación interpretado para realizar una tarea. Generalmente es un programa sencillo, con un suceso de comandos u órdenes que se van ejecutándose de forma secuencial.

#!/bin/bash, que se conoce en la jerga de Unix como Shebang. Aunque este es el más habitual, no siempre se necesita usar para que el Script funcione. En otros proyectos también tienen sus propios Shebangs, como puede ser #!/usr/bin/env python3, #!/bin/sh, etc.

El objetivo del Shebang es simplemente indicar la ruta completa del intérprete de órdenes, para que se pueda ser localizado sea donde sea donde se ejecute el Script. Además, como puedes comprobar, no solo se determina la ruta en él, también el intérprete, en estos casos a Bash, Python 3, y otros intérpretes con los que trabajar.

La mayoría de los *Shell Scripts*[99] (guiones de intérprete de órdenes) Bourne pueden ejecutarse por Bash sin ningún cambio, con la excepción de aquellos Scripts del intérprete de órdenes o consola Bourne que hacen referencia a variables especiales de Bourne o que utilizan una orden interna de Bourne. La sintaxis de órdenes de Bash incluye ideas tomadas desde los intérpretes Korn Shell (ksh) y C Shell (csh), como la edición de la línea de órdenes, el historial de órdenes, la pila de directorios, las variables *$RANDOM* y *$PPID*, y la sintaxis de substitución de órdenes *POSIX*: $(...). Cuando se utiliza como un intérprete de órdenes interactivo, Bash proporciona autocompletado de nombres de programas, nombres de archivos, nombres de variables, etc., cuando el usuario pulsa la tecla *TAB*.

Redirecciones de entrada/salida la sintaxis de Bash permite diferentes formas de redirección de entrada/salida de las que la Shell Bourne tradicional carece. Bash puede redirigir la salida estándar y los flujos de error estándar a la vez utilizando la sintaxis:

```
orden >& archivo
```

que es más simple que teclear la orden Bourne equivalente: "orden > archivo 2>&1". Desde la versión 2.05b, Bash puede redirigir la entrada estándar desde una cadena utilizando la siguiente sintaxis (denominada "Here Strings"):

```
orden <<< "cadena a leer como entrada estándar"
```

si la cadena contiene espacios en blanco, deben utilizarse comillas.

[99] Para generar un archivo de BASH o Script, usemos cualquier editor de texto, por ejemplo *nano miScript*. En el, la primera línea se acostumbra poner el Shebang que se utiliza para indicar al sistema qué intérprete de comandos se debe llamar:

#!/bin/bash

ya creado el Script, es necesario hacerlo ejecutable, para ello usamos:

```
$ chmod u+x miScript
```

o

```
$ chmod 755 miScript
```

y lo ejecutamos en la línea de comandos mediante:

```
$ ./miScript
```

Argumentos Pondremos pasar múltiples argumentos al Shell Script desde la línea de comandos, ejemplo de ellos son:

- $0 El nombre del Script BASH.
- $1, $2 ... $n Los argumentos del Script BASH.
- !$ El último argumento.
- $$ La identificación del proceso del Shell actual.
- $# El número total de argumentos pasados al Script.
- $@ El valor de todos los argumentos pasados al Script.
- $? El estado de salida del último comando ejecutado.
- $! La identificación del proceso del último comando ejecutado.

Ejemplo, si llamamos a este archivo como *mi-Script*, con este contenido:

```
#!/bin/bash
echo "Nombre del Script: $0"
echo "Identificador del Script: $$"
echo "Numero total de argumentos: $#"
echo "Valor de todos los argumentos: $@"
```

lo hacemos ejecutable, mediante:

```
$ chmod u+x mi-Script
```

entonces podemos ejecutarlo usando:

```
$ ./mi-Script texto.txt:
```

También podemos usarlo para simplificar el trabajo en la consola, por ejemplo:

```
$ mkdir -p dir1/dir2/dir3
$ cd !$
```

Uso de condicionales La estructura más fundamental en cualquier estructura de toma de decisiones es una condición *if*. La sintaxis general de una instrucción *if* básica es la siguiente:

```
if [condicion]; then
    código
else
    código
fi
```

y podemos usar las siguientes condiciones:

Condición	Equivalencia
$a -lt $b	$a < $b
$a -gt $b	$a > $b
$a -le $b	$a <= $b
$a -ge $b	$a >= $b
$a -eq $b	$a es igual a $b
$a = $b	$a es igual a $b
$a -ne $b	$a no es igual a $b
|| Operador lógico OR	
&& Operador lógico AND	
-e $FILE	$FILE existe
-d $FILE	$FILE existe y es un directorio
-f $FILE	$FILE existe y es un archivo regular
-L $FILE	$FILE existe y es una liga suave
$STRING1 = $STRING2	$STRING1 es igual a $STRING2
$STRING1 != $STRING2	$STRING1 no es igual a $STRING2
-z $STRING1	$STRING1 es vacía
$a =~ .*subcad*.	$a contiene a las subcadena buscada

por ejemplo:

```
#!/bin/bash
if [ $(whoami) = 'root']; then
    echo "Eres root"
else
    echo "No eres root"
fi
```

o en una sola línea:

```
if [ $(whoami) = 'root' ]; then echo "Eres root"; else echo "No
eres root"; fi
```

Podemos utilizar una declaración *elif* (*else-if*) siempre que se desee probar más de una expresión al mismo tiempo, por ejemplo:

```
#!/bin/bash
if [ $1 -lt 13 ]; then
        echo "Eres un niño"
elif [ $1 -lt 18 ]; then
        echo "Eres un adolecente"
elif [ $1 -lt 65 ]; then
        echo "Eres un adulto"
else
        echo "Eres un adulto mayor"
fi
```

Podemos utilizar una instrucción *if* dentro de otra instrucción *if*. Por ejemplo:

```
#!/bin/bash
if [ $1 -gt 5 ]; then
        if [ $1 -lt 15 ]; then
                echo "Esta fresco"
        elif [ $1 -lt 25 ]; then
                echo "Esta rico el clima"
        else
                echo "Hace mucho calor"
        fi
else
        echo "Esta muy frio ..."
fi
```

otro ejemplo:

```
#!/bin/bash
if [ $# -ne 1 ]; then
      echo "Error: Invalido el numero de argumentos"
      exit 1
fi
file=$1
if [ -f $file ]; then
      echo "$file es un archivo regular"
elif [ -L $file ]; then
      echo "$file es una liga suave"
elif [ -d $file ]; then
      echo "$file es un directorio"
else
```

También podemos usar declaraciones de casos para remplazar varias declaraciones *if*, ya que a veces pueden llegar a ser confusas y difíciles de leer. La sintaxis general de una construcción de casos es la siguiente:

```
#!/bin/bash
CHAR=$1
case $CHAR in
[a-z])
      echo "Letras minusculas" ;;
[A-Z])
      echo "Letras mayusculas" ;;
[0-9])
      echo "Numeros" ;;
*)
      echo "Caracteres especiale"
esac
echo "$file el archivo no existe"
fi
```

Ciclos con for podemos usar ciclos al estilo del lenguaje de programación *C* usando la sintaxis:

```
for ((inicialización; condición; incremento)); do
      comandos
done
```

ejemplo:

```
for ((i=0; i < 10; i++)); do
    echo $i
done
```

O usando una lista o rango, ejemplo

```
for i in {1..10}; do
    echo $i
done
```

o

```
for i in /var/*; do
    echo $i
done
```

otro ejemplo:

```
primos=(2 3 5 7 11 13 17 19 23 29)
for i in "${primos[@]}"; do
    echo $i
done
```

Podemos usar *break* en un ciclo *for* mediante:

```
for ((i=0; i < 10; i++)); do
    echo $i
    if [ $i -eq 3 ]; then
        break
    fi
```

o podemos usar *continue* en un ciclo *for* mediante:

```
for ((i=0; i <= 10; i++)); do
    if [ $(($i % 2)) -ne 1 ]; then
        continue
    fi
    echo $i
done
```

o un ciclo infinito mediante:

```
for ((;;)); do
    comandos
done
```

Ciclos con while la sintaxis general del comando *while* es:

```
while [ condicion ]; do
    comandos
done
```

ejemplo

```
num=1
while [ $num -le 10]; do
    echo $(($num * 3))
    num=$(($num+1))
done
```

o un ciclo infinito mediante:

```
while [ true ]; do
    comandos
done
```

Ciclos con until la sintaxis general del comando *until* es:

```
until [ condicion ]; do
    comandos
done
```

ejemplo:

```
num=1
until [ $num -gt 10]; do
    echo $(($num * 3))
    num=$(($num+1))
done
```

Funciones como en casi todo lenguaje de programación, puede utilizar funciones para agrupar trozos de código de una manera más lógica, o practicar el divino arte de la recursión.

Declarar una función es sólo cuestión de escribir function mi_func { mi_código }.

Llamar a la función es como llamar a otro programa, sólo hay que escribir su nombre.

```
#!/bin/bash
function salir {
    exit
}
function hola {
    echo ¡Hola!
}
hola
salir
echo algo
```

Ejemplo de funciones con parámetros

```
#!/bin/bash
function salir {
   exit
}
function e {
    echo $1
}
e Hola
e Mundo
salir
echo algo
```

Este Script es casi idéntico al anterior. La diferencia principal es la función 'e'. Esta función imprime el primer argumento que recibe. Los argumentos, dentro de las funciones, son tratados de la misma manera que los argumentos proporcionados al Script.

La sintaxis de Bash tiene muchas extensiones que no proporciona el intérprete Bourne. Varias de las extensiones mencionadas se enumeran a continuación:

- Los guiones o Scripts de Bash reciben los argumentos que se le pasa al Shell como *$1, $2, ..., $n*. Se puede obtener el número total de argumentos con el símbolo: *$#*, otras opciones son:*$0* el nombre del Script, *$$* la identificación de ejecución del Script, *$@* el valor de todos los argumentos pasados al Script, *$?* el estado de salida del último comando ejecutado, *$!* la identificación del proceso del último comando ejecutado.

- Usando $# es posible comprobar el número de argumentos entregados al guión antes de realizar alguna acción con ellos:

```
#!/bin/bash
if [ $# -lt 2 ]; then
   echo "Necesitas pasar dos argumentos."
   exit 1
fi
```

- Otra forma de acceder a los argumentos es a través del array: *$@*, por medio del cual se puede iterar sobre todos los argumentos dados:

```
#!/bin/bash
for arg in "$@"
do
   echo "$arg"
done
```

- Una gran limitación del intérprete Bourne es que no puede realizar cálculos con enteros sin lanzar un proceso externo. En cambio, un proceso Bash puede realizar cálculos con enteros[100] utilizando la orden ((...)) y la sintaxis de variables $[...] de la siguiente manera:

  asigna el valor entero 55 a la variable VAR.

[100] Reconoce suma (+), resta (-), multiplicación (*), división entera (/), residuo de la división (%), exponenciación (**).

```
VAR=55
```

suma uno a la variable VAR. Observe la ausencia del carácter '$':

```
((VAR = VAR + 1))
```

otra forma de sumar uno a VAR. Preincremento estilo C:

```
((++VAR))
```

otra forma de sumar uno a VAR. Postincremento estilo C:

```
((VAR++))
```

multiplica la variable VAR por 22 y sustituye la orden por el resultado:

```
echo $[VAR * 22]
```

otra forma de realizar lo mismo:

```
echo $((VAR * 22))
```

- Podemos hacer cálculos de punto flotante instalando el comando *bc*, mediante:

```
# apt install bc
```

y lo podemos usar de esta forma:

```
F=$(echo "3/5" | bc -l)
echo "Resultado $F"
```

- La orden: ((...)) también se puede utilizar en sentencias condicionales, ya que su código de retorno es 0 ó 1 dependiendo de si la condición es cierta o falsa:

```
#!/bin/bash
if ((VAR == Y * 3 + X * 2))
then
    echo Si
fi
((Z > 23)) && echo Si
```

- La orden ((...)) soporta los siguientes operadores relacionales: '==', '!=', '>', '<', '>=', y '<='.
- Manipulación de cadenas, para definir una cadena usamos:

```
var="Valor"
echo $var
```

podemos concatenar dos cadenas, mediante:

```
var1="Prueba"
var2=" de Cadenas"
var3=$var1$var2
```

extraer una porción de la cadena:

```
var="Esto es una prueba"
echo ${var:1:6}
```

extraer desde una posición hasta el final de la cadena

```
var="Esto es una prueba"
echo ${var:5}
```

remplazar una subcadena de una cadena[101]:

```
var="Linux es un gran sistema operativo"
echo ${var/Linux/Debian}
```

borrar una subcadena de una cadena:

```
var="9938-333-334-334"
echo ${var/-}
```

borrar todas las apariciones de la subcadena:

[101] Para ver por separado cada uno de los directorios que componen el PATH podemos usar:
$ echo "${PATH//:/$'\n'}"

```
var="9938-333-334-334"
echo ${var//-}
```

también podemos utilizar [] para especificar rangos de caracteres:

```
var="esta es una prueba"
echo ${var//[a-z]/*}
```

convertir a mayúsculas una cadena:

```
var="esta es una prueba"
echo ${var^^}
```

si sólo se necesita convertir a mayúscula la primera letra usamos:

```
echo ${var^}
```

convertir a minúsculas una cadena:

```
var="Esta es una Prueba"
echo ${var,,}
```

si sólo se necesita convertir a minúscula la primera letra usamos:

```
echo ${var,}
```

- Manejo de ciclos con el contenido arrojado por algún comando, ejemplo:

```
#!/bin/bash
for i in $( ls *.cpp); do
   echo archivo: $i
done
```

- Manejo de secuencias usando: *seq* Primero Incremento Ultimo, ejemplos:

```
$ seq 7
$ seq 3 5
$ seq 0.1 0.1 1
$ seq 10 -4 2
$ seq -w 100
$ seq -f 'Mayo %g, 2020' 5
$ seq -s"  " 5
```

- Manejo de ciclos con una secuencia numérica, ejemplo:

```
#!/bin/bash
for i in ‘seq 1 10‘;
do
   echo $i
done
```

- Manejo de ciclos con *while*, ejemplo:

```
#!/bin/bash
COUNTER=0
while [ $COUNTER -lt 10 ]; do
   echo The counter is $COUNTER
   let COUNTER=COUNTER+1
done
```

- Manejo de ciclos con *until*, ejemplo:

```
#!/bin/bash
COUNTER=20
until [ $COUNTER -lt 10 ]; do
   echo COUNTER $COUNTER
   let COUNTER-=1
done
```

- Códigos de salida, cuando un programa Bash concluye emite un código que puede ser visualizado y atrapado, estos códigos son:

  0, ejecución exitosa
  1, error general no definido
  2, mal uso de caracteres
  126, comando no encontrado
  128, argumento no valido
  128+nm error fatal
  130, terminado por Ctrl-c
  255, código de salida fuera de rango

  Ejemplo de esto, probemos para código cero:

```
$ date ; echo $?
```

para código 127:

```
$ fallo; echo $?
```

- Solicitud de datos por teclado mediante el comando *read*, algunas de sus opciones son:

```
$ read variable ; echo $variable
$ read -p "Nombre del archivo a buscar" archivo ; echo $archivo
$ read -s -p "Escriba su clave de acceso" secreta ; echo $secreta
$ read -n 5 -p "Nombre de usuario" usuario ; echo $usuario
$ read -a -p "Introduzca los valores a buscar" arreglo; echo
${arreglo[@]}
```

- Si queremos mostrar mensajes de texto en color, podemos usar los siguientes códigos de escpe ANSI:

0;30	Negro
0;34	Azul
0;32	Verde
0;36	Cyan
0;31	Rojo
0;35	Púrpura
0;33	Café
0;37	Gris Claro
1;30	Gris Oscuro
1;34	Azul Claro
1;32	Verde Claro
1;36	Cyan Claro
1;31	Rojo Claro
1;35	Fucsia
1;33	Amarillo
1;37	Blanco

Al escribir uno de los códigos anteriores, activaremos el color correspondiente. Para desactivarlo, tan sólo tenemos que usar el código *0m*, por ejemplo:

```
VERDE='\033[0;32m'
NC='\033[0m'
echo -e "${VERDE}\nActualizando paquetes ...\n${NC}"
```

- Tenemos dos tipos de variables del Shell: Variables de medio ambiente (global), las del Shell y definidas por el usuario.

Para conocer las variables del medio ambiente podemos usar:

```
$ printenv
```

o

```
$ env
```

o conocer el valor de solo una, usamos:

```
$ printenv PATH
```
[102]

Para conocer la lista de todas las variables incluidas las definidas por el usuario, usamos:

```
$ set
```

Para asignar un valor a una variable usamos:

```
$ var=valor
```

o

```
$ export var=valor
```

para ver el valor de una variable podemos usar al comando *echo*, por ejemplo:

```
$ echo "$PS1"
```

[102] Para ver por separado cada uno de los directorios que componen el PATH podemos usar:
$ echo "${PATH//:/$'\n'}"

y si deseamos limpiar una variable usamos:

```
$ unset var
```

Podemos también definir variables de solo lectura, usando:

```
$ readonly var=valor
```

Observación 3 *Un proceso Bash no puede realizar cálculos en punto flotante sin lanzar un proceso externo (como bc). Las únicas Shell Unix capaces de esto son Korn Shell (versión de 1993) y zsh (a partir de la versión 4.0).*

Manejo de Errores El manejo de errores es importante en los Scripts Bash, sin él es posible que ni siquiera sepamos que algo ha fallado y encontrar la fuente del error en los Scripts problemáticos se vuelve mucho más difícil. A continuación, mostramos algunas formas de manejar los errores en los Scripts de Bash.

Si bien existe la opción *set -x* que permite habilitar el modo depuración, una mejor opción es usar *set -e* (set -o errexit) para salir de un Script cuando falla un comando, esto también establece el estado de salida del Script al del comando fallido. Si no usamos *set -e*, la secuencia de comandos continuará ejecutándose incluso cuando un comando haya fallado y el estado de salida de la secuencia de comandos será el del último comando en la secuencia de comandos. Utilice *set +e* para desactivar la funcionalidad *set -e* dentro de un Script. Ejemplo:

```
#!/bin/bash
set -e
echo "Script iniciado."
date
sleep 2
date
sleep 2
# El comando dateeeee no existe y debe fallar.
# set -e causa que el Script falle y salga con el código de
estado de dateeeee.
dateeeee
echo "Script finalizado exitosamente."
```

Además, podemos usar set -u (set -o nounset) para salir de un Script cuando se intente usar una variable que no ha sido inicializada y podemos solicitar que ciertas variables sean declaradas estáticas usando:

```
#!/bin/bash
set -u
readonly password_file="/etc/passwd"
```

Manejo de errores personalizado si deseamos hacer algo más que simplemente salir en caso de error, podemos verificar un comando en busca de fallas y luego manejarlo. No usemos $? para comprobar si hay fallos, no es necesario. Ejemplo:

```
#!/bin/bash
set -e
echo "Script iniciado."
date
sleep 2
date
sleep 2
# El comando dateeeee no existe y debe fallar.
if ! dateeeee 2>/dev/null
then
    echo "Debe haber un problema, lo volvemos a intentar"
    date
fi
echo "Script finalizado exitosamente."
```

Manejar errores en Pipeline si usamos Pipelines en nuestros Scripts, usemos *set -e* en combinación con *set -o* pipefail para salir de un Script si falla algún comando en un Pipeline; de lo contrario, el estado de salida de un Pipeline será el último comando en el Pipeline y cualquier comando que falló anteriormente en la canalización no se detectará. Ejemplo:

```
#!/bin/bash
set -eo pipefail
echo "Script started."
# El comando dateeeee no existe y debe fallar.
```

```
    # set -eo pipefail causa que el Script salga con el código de
  salida de dateeeee.
    dateeeee | echo "todo bien"
    echo "Script finalizado exitosamente."
```

Compilar un Script algunas veces es necesario compilar un Script (el resultado dejará al Script ejecutable pero no podrás ver su código fuente), para ello instalamos la herramienta *shc*, mediante:

```
# apt install shc
```

El comando realiza un proceso en dos pasos. Primero genera un código en C a partir del Shell Script, que luego compila para crear un programa binario. Si tenemos un Script HolaMundo.sh, entonces el código en C será HolaMundo.sh.c y el ejecutable HolaMundo.sh.x, para ello usamos:

```
$ shc -v -f HolaMundo.sh
```

y lo ejecutamos usando:

```
$ ./HolaMundo.sh.x
```

Podemos indicarle al ejecutable que tenga fecha de expiración *-e* (y *-m* un mensaje sobre la expiración) o crear un binario redistribuible *-r* para que se ejecute en diversas distribuciones de Linux, entre otros parámetros.

¿Qué es un Código de Estado de Salida? Un código de salida o estado de salida nos informa sobre el estado del último comando ejecutado. Si el comando se completó correctamente o finalizó con un error. Esto se obtiene después de que finaliza el comando.

La ideología básica es que los programas devuelven el código de salida 0 para indicar que se ejecutó exitosamente y sin problemas. El código 1 o cualquier otro distinto de 0 se considera fallido. Hay muchos más códigos de salida además del 0 y el 1, que veremos en esta sección.

Echemos un vistazo rápido a los códigos de salida destacados en el Shell de Linux:

Código de Salida y Significado del código

0 Comando ejecutado sin errores

1 Código para errores genéricos.

2 Uso incorrecto de comandos (o argumentos)

126 Permiso denegado (o) incapaz de ejecutar

127 Comando no encontrado o error de RUTA

$128 + n$ El comando finalizó externamente al pasar señales o encontró un error fatal

130 Terminación mediante Ctrl+C o SIGINT (código de terminación 2 o interrupción del teclado)

143 Terminación por SIGTERM (terminación por defecto)

255/$*$ El código de salida superó el rango 0-255, por lo que se cerró

Recuperando el Código de Salida el código de salida del comando ejecutado previamente se almacena en la variable especial $?. Puede recuperar el estado de salida ejecutando:

```
$ echo $?
```

esto se utilizará en todas nuestras demostraciones para recuperar el código de salida.

Código de Salida 0 el código de salida 0 significa que el comando se ejecuta sin errores. Idealmente, este es el mejor caso para completar comandos. Por ejemplo, ejecutemos un comando básico como este:

```
$ neofetch
$ echo $?
```

este código de salida 0 significa que el comando en particular se ejecutó exitosamente, ni más ni menos. Puedes intentar matar un proceso; también devolverá el código 0.

```
$ pkill lxappearance
```

matar una aplicación (mismo Shell) da como resultado el código 0. Ver el contenido de un archivo también devolverá un código de salida 0, lo que implica sólo que el comando 'cat' se ejecutó correctamente.

Código de Salida 1 el código de salida 1 también es común. Generalmente significa que el comando terminó con un error genérico. Por ejemplo, usar el administrador de paquetes sin permisos da como resultado el código 1, si intento esto:

```
$ su root
```

me dará un código de salida 1, lo que significa que el último comando resultó en un error. Si bien esto es un entendimiento general, también podemos interpretarlo como "operación no permitida". Operaciones como dividir por cero también dan como resultado el código 1.

Código de Salida 2 este código de salida se proporciona cuando el comando ejecutado tiene un error de sintaxis. El mal uso de los argumentos de los comandos también produce este error. Generalmente sugiere que el comando no se pudo ejecutar debido a un uso incorrecto. Por ejemplo, agregué dos guiones a una opción que se supone que tiene un guión:

```
$ grep –z archivo.txt
```

regresa un código 2. cuando se deniega el permiso, como acceder a la carpeta /root, aparece el código de error 2.

Código de Salida 126 126 es un código de salida peculiar ya que se utiliza para indicar que un comando o Script no se ejecutó debido a un error de permiso. Este error se puede encontrar cuando intenta ejecutar un Script de Shell sin otorgar permisos de ejecución.

Código de Salida 127 este es otro común. El código de salida 127 se refiere a "comando no encontrado". Suele ocurrir cuando hay un error tipográfico en el comando ejecutado o el ejecutable requerido no está en la variable $PATH.

Código de Salida Serie 128+n cuando se finaliza una aplicación o comando o su ejecución falla debido a un error fatal, se produce el código adyacente a 128 (128+n), donde n es el número de señal. Esto incluye todos los tipos de códigos de terminación, como SIGTERM, SIGKILL, etc., que se aplican al valor 'n' aquí.

Código 130 o SIGINT la señal SIGINT o Señal para interrupción del teclado se induce interrumpiendo el proceso mediante la señal de terminación 2 o mediante Ctrl+C. Como la señal de terminación es 2, obtenemos un código 130 (128+2).

Código 137 o SIGKILL la señal de terminación SIGKILL que finaliza el proceso instantáneamente tiene una señal de terminación 9. Este es el último método que se debe utilizar al finalizar una aplicación.

Código 143 o SIGTERM la señal SIGTERM o Señal para terminar es el comportamiento predeterminado cuando se finaliza un proceso sin especificar argumentos. El código de terminación para SIGTERM es 15, por lo tanto, esta señal obtiene un código de salida de 143

¿Qué pasa si el Código Supera 255? las versiones recientes de BASH conservan el valor del código de salida original incluso más allá de 255, pero generalmente, si el código excede 255, se cierra. Es decir, el código 256 se convierte en '0', 257 se convierte en '1', 383 se convierte en '127', y así sucesivamente. Para garantizar una mejor compatibilidad, mantenga los códigos de salida entre 0 y 255.

Observación 4 *Si está utilizando estos códigos en un Script de Shell, asegúrese de comprender el significado de cada código para facilitar la resolución de problemas.*

Algunos Consejos para Programar en Bash Un Bash Script es un archivo en el que se codifican múltiples comandos de Shell para realizar una tarea en particular, algunas sanas costumbres para escribir un Script, principalmente para mejorar la eficiencia y hacerlo más legible son:

Comentar el código definitivamente algo básico, pero que muchos olvidan y que es siempre de mucha utilidad, sea para uno mismo, o para otros que quieran revisar o modificar el Script, es sin duda determinante para lograr un código limpio y que logre explicar varias partes de códigos complejos. También es eficaz cuando se trabaja en grupo en un gran proyecto, ya que ayuda a comprender qué hace realmente la función o el método.

Uso de Funciones una función es un conjunto de comandos agrupados para realizar una tarea específica que ayuda a modular el flujo de trabajo, eliminando la repetición del código. Hace que el código sea limpio y más legible, así como fácil de mantener:

```
#!/bin/bash
function check_root() {
    echo "la función ha sido llamada";
}
```

Uso de comillas dobles ayudará a eliminar el englobamiento innecesario, así como la división de palabras, incluidos los espacios en blanco, cuando los valores de las variables contienen un carácter separador o un espacio en blanco.

Terminar ejecución por error a veces, al ejecutar un Script, puede haber algún error de ejecución. Incluso si un comando no se ejecuta, la secuencia de comandos puede continuar corriendo y afectar a los demás comandos del Script. Para evitar más errores lógicos, debemos incluir *set -o errexit* o *set -e* para finalizar el comando en caso de error:

```
#!/bin/bash
# Terminar el Script si hay error
set -o errexit
```

Declarando Variables según su tipo de datos y usos. Cuando la variable no es declarada, es posible que Bash no pueda ejecutar el comando relacionado. Las variables se pueden declarar global o localmente en el Script. Por ejemplo:

```
    #!/bin/bash
    # Declaración de variable, como notas es de formato vari-
able=valor
    declare -r -i x=30
    function my_variable(){
        local -r name = ${HOME}
    }
```

El uso de llaves vincula las variables con llaves mientras usa la concatenación de variables con cadenas para evitar usos innecesarios de variables. Esto también ayuda a identificar fácilmente la variable mientras se usa en una cadena. Por ejemplo:

```
#!/bin/bash
# Termina el Script si hay error
set -o errexit
# Variable personalizada
data= "${USER}_data is being used"
```

Sustitución del comando al asignar la salida del comando a la variable, Bash utiliza la función de sustitución de comandos. Necesitamos usar $() en lugar de comillas inversas para asignar la salida a las variables como se recomienda:

```
#!/bin/bash
# Termina el Script si hay error
set -o errexit
#Muestra la Fecha
date_now = ${date}
```

Nomenclatura de variables en nuestro sistema, todas las variables de entorno se nombran con letras mayúsculas. Entonces, cuando declaramos una variable local, debemos declararla usando letras minúsculas para evitar conflictos entre el entorno y el nombre de la variable local:

```
#!/bin/bash
# Termina el Script si hay error
set -o errexit
user_var = "$HOME es tu login correcto."
```

Declarar Variables Estáticas si tienes datos estáticos que permanecen sin cambios durante todo el Script, puedes asignar el valor a una variable estática cuyo valor no se puede modificar. Puedes declarar la variable estática usando el comando de solo lectura:

```
#!/bin/bash
# Probando configuración nginx
readonly test_conf_path = "/etc/nginx/conf.d/test.conf"
```

Depuración la depuración es la parte esencial para identificar un problema. Podemos verificar el error de sintaxis del Script, por lo que para verificarlo necesitamos ejecutar el Script Bash con *-n* usando el comando Bash:

```
$ bash -n script_name
```

además, podemos habilitar y ejecutar el Script en modo de depuración usando el siguiente comando.

```
$ bash -x script_name
```

5.8 Algunos Ejemplos en Bash

Para generar un archivo de Bash o Script, usemos cualquier editor de texto, por ejemplo:

```
$ nano miScript
```

en el, la primera línea se acostumbra poner:

```
#!/bin/bash
```

ya creado el Script, es necesario hacerlo ejecutable, para ello usamos:

```
$ chmod u+x miScript
```

o

```
$ chmod 755 miScript
```

y lo ejecutamos en la línea de comandos mediante:

```
$ ./miScript
```

si necesitamos conocer la línea que se esta ejecutando de un Script podemos usar:

```
$ bash -x ./miScript
```

Estos y otros ejemplos de Bash se pueden descargar de:
Bash

Conocer el uso de memoria de los procesos Podemos usar el comando *top* que nos informa en tiempo real el consumo de RAM o algunas de estas opciones:

```
$ ps aux –sort -rss | head
$ top -c -b -o +%MEM | head -n 20
```

Uso de comillas dobles y sencillas este pequeño ejemplo nos muestra visualmente el uso de comillas sencillas y dobles:

```
$ a=7; echo $a; echo "$a"; echo '$a'; echo "'$a'"; echo '"$a"'
```

Convertir imágenes usando xargs para convertir imagenes .png a .jpg, usaremos el comando *ls* para obtener los nombres de archivo que pasaremos a *xargs* que nos permite llamar a *conver* mediante *bash* para hacer la conversión, mediante:

```
$ ls -l *.png | xargs -n 1 bash -c 'convert "$0" "${0%.png}.jpg"'
```

y para convertir imagenes de .jpg a png, usamos:

```
$ ls -l *.jpg | xargs -n 1 bash -c 'convert "$0" "${0%.jpg}.png"'
```

Números pseudoaleatorios se pueden generar números pseudoaleatorios entre 0 y 32767, por ejemplo generamos números entre 0 y 100 mediante:

```
$ echo $((RANDOM % 100+1))
```

Longitud de una cadena disponemos de varias formas de conocer la longitud de una cadena, por ejemplo:

```
#!/bin/bash
cadena="Esta es una cadena"
tam1=${#cadena}
tam2=$(expr length "$cadena")
tam3=$(echo $cadena | awk '{print length}')
tam4=$(echo -n $cadena | wc -m)
echo $tam1 $tam2 $tam3 $tam4
```

Buscar en una cadena una subcadena podemos buscar en una cadena una subcadena y preguntar sobre ella, mediante:

```
#!/bin/bash
url="http://www.mmc.geofisica.unam.mx/acl/Textos/"
if [[ $url =~ .*acl*. ]]
then
    echo "Se encontro la cadena acl en ${url}."
else
    echo "No se encontro."
fi
```

Emoticonos en la consola podemos también generar emoticonos, muestra de ello es:

```
$ printf $(printf '\\U%x\\x20' {{128512..128591},{128640..128725}})
```

Formatear números es posible dar formato a números mediante el uso de *numfmt*, por ejemplo:

```
$ numfmt –grouping 123456789
```

acepta sufijos como 1K, 1Ki, 1M, 1Mi, 1G, 1Gi, por ejemplo:

```
$ numfmt –from=iec-i 2Gi
$ numfmt –to=iec 1000000
```

y lo podemos usar por ejemplo en un ls:

```
$ ls -l | numfmt –header –field 5 –to=iec
```

También podemos agregar ceros a la izquierda al visualizar una cantidad numérica, por ejemplo:

```
for((i=1; i<=10; ++i)): do
    printf "%03d\n" $i
done
```

en caso de no querer mostrar directamente el resultado, podemos asignarlo a una variable usando el parámetro *-v* del comando *printf*:

```
for((i=1; i<=10; ++i)): do
    printf -v j "%03d\n" $i
    echo $j
done
```

Realizar lista de potencias de 3 podemos usar el Bash para visualizar las primeras 19 potencias de 3, usando:

```
$ printf "%s\n" $[3**{1..19}] | nl
```

o

```
$ printf "%s\n" 3^{1..19} | bc | nl
```

o

```
$ echo -n 3^{1..19} | xargs -d' ' -I@ sh -c 'printf "%6s =
%48s\n" @ $(echo "@" | bc)'
```

o

```
$ for i in {1..19}; do printf "%0.2d %s\n" $i $[3**$i]; done
```

Manipular ruta de un archivo dada la ruta de un archivo:

```
archivo="/usr/share/icons/64x64/application-wireshark-doc.png"
```

para obtener sólo la ruta usamos:

```
echo "${archivo%/*}"
```

para obtener sólo el nombre de archivo usamos:

```
echo "${archivo##*/}"
```

para eliminar la extensión del nombre del archivo usamos:

```
nombre="${archivo##*/}"
echo ${nombre%.*}
```

para extraer la extensión del nombre del archivo usamos:

```
echo ${achivo##*.}
```

También podemos usar al comando basename, por ejemplo:

```
$ basename /etc/passwd
$ basename /etc/sysctl.conf .conf
```

de esta forma podemos renombar todos los archivos con extensión .jpeg a jpg, mediante:

```
for file in *.jpeg; do
    mv – "$file" "$(basename $file .jpeg).jpg"
done
```

en caso de necesitar la trayectoria pero no el nombre de archivo, podemos usar:

```
$ dirname /var/spool/mail/antonio.txt??
```

y podemos combinar a *basename* y *dirname*, por ejemplo:

```
Trayectoria="/home/antonio/data/fichero.txt"
tray=$(dirname "$Trayectoria")
nomb=$(basename "$Trayectoria")
echo $tray
echo $nomb
```

Manipular nombre de archivos Cambiar los espacios por subrayado:

```
$ for FILE in * ; do NEWFILE=‘echo $FILE | sed ’s/ /_/g’‘
; mv "$FILE" $NEWFILE ; done
```

Añadir un sufijo .txt a cada archivo:

```
$ for FILE in * ; do mv $FILE $FILE.txt ; done
```

Quitar el sufijo .txt a cada archivo:

```
$ for FILE in *.txt ; do NEWFILE=‘echo $FILE | sed ’s/.txt$//’‘
; mv $FILE $NEWFILE ; done
```

Convertir el sufijo .TXT a .txt

```
$ for FILE in *.TXT ; do NEWFILE=‘echo $FILE | sed
’s/.TXT$/.txt/’‘ ; mv $FILE $NEWFILE ; done
```

Añadir un prefijo maths_ a cada archivo:

```
$ for FILE in * ; do mv $FILE maths_$FILE ; done
```

Quitar el prefijo maths_ a cada archivo_

```
$ for FILE in * ; do NEWFILE=‘echo $FILE | sed ’s/^maths_//’‘
; mv $FILE $NEWFILE ; done
```

Convertir nombre de mayúsculas a minúsculas a cada archivo:

```
$ for FILE in * ; do mv $FILE ‘echo $FILE | tr ’[:upper:]’
’[:lower:]’‘ ; done
$ for FILE in * ; do mv $FILE ‘echo $FILE | tr ’[A-Z]’ ’[a-z]’‘
; done
```

Convertir nombre de minúsculas a mayúsculas a cada archivo:

```
$ for FILE in * ; do mv $FILE ‘echo $FILE | tr ’[:lower:]’
’[:upper:]’‘ ; done
$ for FILE in * ; do mv $FILE ‘echo $FILE | tr ’[a-z]’ ’[A-Z]’‘
; done
```

Quiero que los espacios cambien a un guión, salvo cuando haya dos o más espacios consecutivos, en el que necesito un solo subrayado. Mientras estamos en ello, si un espacio está al lado del punto de un prefijo, vamos a quitarlo. También quiero hacer los sufijos minúsculas.

```
$ for FILE in * ; do NEWFILE=‘echo $FILE | sed -e ’s/.TXT$/.txt/’
-e ’s/[ ]*[ ]/_/g’ -e ’s/_[.]/./g’‘ ; mv "$FILE" $NEWFILE ; done
```

Dar a todos los archivos el mismo nombre, pero con un número diferente:

```
$ NUM=0 ; for FILE in * ; do NUM=‘expr $NUM + 1‘ ; mv
$FILE Archivo\($NUM\) ; done
```

Mantener el nombre original y separar el número secuencial con un subrayado:

```
$ NUM=0 ; for FILE in * ; do NUM=‘expr $NUM + 1‘ ; mv
$FILE ${FILE}_$NUM ; done
```

Crear y remover archivos podemos usar al comando *seq* para crear y borrar un grupo de archivos:

```
$ touch $(seq -f "Archivo%g" 4)
$ rm $(seq -f "Archivo%g" 4)
```

otra forma es:

```
$ for i in web{0..10} db{0..2} otro_{a,b}{1..4}; do touch $i;
done
$ for i in web{0..10} db{0..2} otro_{a,b}{1..4}; do rm $i;
done
```

Manejo de parámetros Si generamos un Script con nombre *test.sh*, entonces podemos controlar por ejemplo dos parámetros (*–alpha* o *-a* y *–config* o *-c*) mediante:

```
#!/bin/bash
while [ True ]; do
if [ "$1" = "–alpha" -o "$1" = "-a" ]; then
      ALPHA=1
      shift 1
elif [ "$1" = "–config" -o "$1" = "-c" ]; then
      CONFIG=$2
      shift 2
else
      break
fi
done
echo $ALPHA
echo $CONFIG
ARG=( "${@}" )
for i in ${ARG[@]}; do
      echo $i
done
```

y lo podemos usar mediante:

```
$ ./test.sh –config my.conf foo bar
$ ./test.sh -a –config my.conf baz
```

Cree un árbol de carpetas usando el comando *mkdir* podemos crear un árbol de carpetas, con carpetas A - Z en el primer nivel, luego 0 a 100 en cada uno de ellos. Y todas las demás letras que comienzan en a en las carpetas pares y todas las demás que comienzan con b en impares

```
$ mkdir -p tree/{A..Z}/{0..100..2}/{a..z..2} tree/{A..Z}/{1..100..2}/{b..z..2}
```

Busca archivos ASCII y extrae direcciones IP de ellos podemos buscar en la trayectoria actual a todos los archivos ASCII que contienen direcciones IP validas y pedir que nos muestre estas, usando:

```
$ find . -type f -exec grep -Iq . {} \; -exec grep -oE "(25[0-5]|2[0-4]\
[0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25\
[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9]\
[0-9]?)" {} /dev/null \;
```

Crear directorios a partir del contenido de un archivo si tenemos un archivo con distintos nombres de directorios que necesitamos crear, por ejemplo:

```
servidores
comando/Linux
comando/Bash
editores/vin
editores/nano
```

y están en el archivo *directorios.txt*, podemos usar para crearlos:

```
$ xargs -I{} mkdir -p "{}" < directorios.txt
```

Crear archivos temporales cuando se trabaja en Bash se requiere construir archivos, muchos de ellos temporales, para ellos usamos:

```
$ tmp_file=$(mktemp)
```

si queremos ver el archivo creado podemos usar:

```
$ tmp_file=$(mktemp) ; echo $tmp_file
```

y para borrarlo podemos usar:

```
$ rm ${temp_file}
```

Otras opciones son:

```
$ tmp_file=/tmp/foobar.$$
$ tmp_file=/tmp/foobar.$RANDOM
$ tmp_file=‘date ’+%Y-%m-%d-%H-%M’‘-‘uuidgen -t | head
-c 5‘
```

Partir un archivo de texto en múltiples archivos el comando *split* nos permite partir un archivo de texto en múltiples archivos de un tamaño indicado, por ejemplo en 50 líneas:

```
$ split –lines=50 archivo.txt
```

Formateador de texto simple el comando *fmt* nos permite formatear la entrada que se le de, básicamente rellena y junta líneas de texto mientras que mantiene líneas en blanco y sangrías, por ejemplo reformateamos el texto del archivo para ajustarlo a una columna de 50 caracteres de ancho:

```
$ fmt -w 50 archivo.txt
```

Crear directorios temporales cuando se trabaja en Bash se requiere construir directorios donde contener los archivos de nuestro programa, muchos de ellos temporales, para ellos usamos:

```
$ tmp_dir=$(mktemp -d)
```

si queremos ver el directorio creado podemos usar:

```
$ tmp_dir=$(mktemp -d) ; echo $tmp_dir
```

y para borrarlo podemos usar:

```
$ rmdir ${temp_dir}
```

o

```
$ rm -R ${temp_dir}
```

Otras opciones son:

```
$ tmp_dir=$(mktemp -d -t ci-XXXXXXXXXX)
$ tmp_dir=$(mktemp -d -t ci-$(date +%Y-%m-%d-%H-%M-
%S)-XXXXXXXXXX)
$ tmp_dir=$(mktemp -d -t ci-XXXXXXXXXX –tmpdir=/home/antonio/tmp)
```

Leer un nombre de archivo dentro del Script se solicita al usuario un nombre de archivo mediante el comando *read*, para por ejemplo, contar el número de líneas del archivo:

```
#!/bin/bash
echo -n "Introduzca un nombre de archivo: "
read archivo
nlineas=$(wc -l < $archivo)
echo "Tiene $nlineas lineas el archivo $archivo"
```

Correr el Scrypt solo por el usuario root hay veces que debemos tener certeza de que solo el usuario root deba correr el contenido del Script, para ello podemos usar:

```
#!/bin/bash
if [[ "$(whoami)" != root ]]; then
    echo "Solo el usuario root puede correr este script."
    exit 1
fi
echo "Haciendo algo..."
exit 0
```

Pasar argumentos al Script podemos pasar argumentos al Script en la línea de comandos al momento de su ejecución, si llamamos a este archivo como *mi-Script*, con este contenido:

```
#!/bin/bash
echo "Nombre del Script: $0"
echo "Numero total de argumentos: $#"
echo "Valor de todos los argumentos: $@"
nlineas=$(wc -l < $1)
echo "Tiene $nlineas lineas el archivo $1"
```

entonces al ejecutar usando:

```
$ ./mi-Script texto.txt
```

nos permitirá contar el número de líneas del archivo *texto.tx*t

Agregar pausa a un Script podemos agregar a nuestro Script que haga una pausa antes de continuar su ejecución, algunas opciones son:

```
$ read -p "Presione [Enter] para continuar ..."
$ read -t 5 -p "Esperaremos 5 segundos para continuar"
$ sleep 5 && comando
```

Hacer cálculos de punto flotante podemos hacer conversión de grados Celsius a Fahrenheit, mediante en programa ConvierteC2F:

```
#!/bin/bash
# bc se le indica que use dos decimales: scale=2
F=$(echo "scale=2; $1 * (9/5) + 32" | bc -l)
echo "$1 grados Celsius es igual a $F grados Fahrenheit."
```

y para usarlo:

```
$ ./ConvierteC2F 25
```

Leer un archivo línea por línea en muchos ejemplos de Bash es necesario leer un archivo línea por línea, aquí mostramos algunas opciones:

```
$ while read line; do echo -e "$line\n"; done < archivo.txt
$ cat emails.txt |while read line; do echo "$line"; done
```

Leer un archivo en formato .CSV Hay muchas aplicaciones que generan archivos de texto con valores separados por comas (*.CSV*), los cuales son fácilmente leíble en BASH, por ejemplo un archivo *a.csv* con dos columnas, puede ser leído usando:

```
#!/bin/bash
while IFS=, read -r C1 C2
do
    echo "$C1 y C2"
done
```

si lo grabamos con nombre leeCSV.sh, entonces lo podemos ejecutar mediante:

```
$ leeCSV.sh < a.csv
```

Uso del comando column permite mostrar la salida de algún comando separado por columnas:

```
$ mount | column -t
```

Poner un texto al revés el comando *rev* permite que cualquier texto que se le pase lo ponga al revés, ejemplo:

```
$ rev <<< "esta es una cadena"
```

Encontrar los factores primos de un número el comando factor permite encontrar los factores primos de un número, ejemplo:

```
$ factor <<< 27832878
```

Conversión de base convierte el número hexadecimal 2F al decimal 47, usando:

```
$ echo $((0x2F))
```

o mediante:

```
$ echo $((16#2F))
```

Crear un archivo compactado por cada archivo indicado

```
$ for f in *.txt ; do tar cvf "$f.tar.bz2" "$f"; done
```

Crear un archivo tar por cada directorio hijo de nuestra actual posición:

```
$ find . -maxdepth 1 -mindepth 1 -type d -exec tar cvf \
{}.tar {} \;
```

Renombrar archivos si bien existe el comando *rename* para renombrar archivos **.bak* a **.txt*, usamos:

```
$ rename .bak .txt *.bak
```

también podemos hacer un Script para renombrar todos los archivos **.bak* a **.txt*, ejemplos:

```
$ for j in *.bak; do mv – "$j" "${j%.bak}.txt"; done
$ for j in *.bak; do mv -v – "$j" "${j%.bak}.txt"; done
```

Encontrar archivos y directorios grandes para encontrar, digamos los diez directorios más grandes en nuestra actual trayectoria, usamos:

```
$ du -h | sort -hr | head -n 10
```

para encontrar, digamos los diez archivos más grandes en nuestra actual trayectoria, usamos:

```
$ du -ah | sort -hr | head -n 10
```

Buscar un directorio y borrar su contenido un opción para buscar un directorio y borrar su contenido es:

```
$ find . -type d -iname nombre -delete
```

este comando encontrará todos los directorios y tratará de borrarlos, pero mandará error si el directorio no esta vació. Para solucionarlo podemos usar:

```
$ find . -type d -name "nombre" -exec rm -rf {} +
```

otras opciones son:

```
$ find . -type d -name "nombre" -exec rm -rf \;
$ find . -type d -name "nombre" -exec rm -rf "{}" \;
$ find . -type d -name "nombre*" -print0 | xargs -0 -I {}
/bin/rm -rf "{}"
```

podemos indicarle que realice la búsqueda en hasta 4 niveles de profundidad en el presente directorio:

```
$ find . -type d -name "nombre" -depth +4 -print0 -exec rm
-rf {} +
```

Algunas veces es necesario solo borrar los directorios vacíos, una forma de hacerlo es:

```
$ find . -type d -empty -print0 | xargs -0 -I {} /bin/rm -rf
"{}"
```

Conocer la velocidad de grabación del disco usando el comando *dd* y generando un archivo temporal, podemos conocer la velocidad de grabación de nuestro disco, mediante:

```
$ dd if=/dev/zero of=/tmp/salida.img bs=5G count=1 oflag=dsync;
rm -rf /tmp/salida.img
```

y para medir la latencia usamos:

```
$ dd if=/dev/zero of=/tmp/salida.img bs=512 count=1000
oflag=dsync; rm -rf /tmp/salida.img
```

Aprender algo nuevo nos da el manual de un comando diferente instalado en nuestro sistema en cada llamada:

```
$ man $(ls /bin | shuf | head -1)
```

Comandos usados y cuantas veces si deseamos conocer cuales son los comandos usados y cuantas veces los hemos usado, escribimos:

```
$ history | awk '{print $2}' | sort | uniq -c | sort -nr
```

Generando contraseñas si bien los programas GPG y OpenSSL nos permiten generar contraseñas, podemos usar en Bash para ello, en todos los ejemplos generamos contraseñas de 32 caracteres:

```
$ date +%s | sha256sum | base64 | head -c 32 ; echo
$ < /dev/urandom tr -dc _A-Z-a-z-0-9 | head -c${1:-32};echo;
$ tr -cd '[:alnum:]' < /dev/urandom | fold -w32 | head -n1
$ strings /dev/urandom | grep -o '[[:alnum:]]' | head -n 33 | tr
-d '\n'; echo
$ < /dev/urandom tr -dc _A-Z-a-z-0-9 | head -c32
$ dd if=/dev/urandom bs=1 count=32 2>/dev/null | base64
-w 0 | rev | cut -b 2- | rev
```

```
$ </dev/urandom tr -dc '12345!@#$%qwertQWERTasdfgAS-
DFGzxcvbZXCVB' | head -c32; echo ""
$ randpw(){ < /dev/urandom tr -dc _A-Z-a-z-0-9 | head -
c${1:-32};echo;} && randpw
```

Visualiza la IP del equipo imprime las direcciones de IPv4 asociada a las redes del equipo:

```
$ ip address | grep inet | grep -v inet6 | awk '{ print $2 }'
```

Lanzar procesos después de cierto tiempo algunas veces queremos que el sistema lance eventos después de cierto tiempo, supongamos queremos que nos avise que en 5 minutos -pueden ser segundos (s), minutos (m), horas (h) o días (d)- tenemos junta, usamos:

```
$ sleep 5m && echo "ir a la junta ... ahora"
```

pero podemos lanzar una aplicación, por ejemplo que toque música para avisarnos de la junta:

```
$ sleep 5m && vlc /home/usuario/alarma.mp3
```

Buscar archivos duplicados y cambiarlos por ligas simbólicas duras cuando se trabaja en proyectos es común terminar con múltiples copias de algunos archivos, pero podemos encontrar todos y remplazarlos por ligas simbólicas duras, para ello usaremos el programa hardlink, lo instalamos usando:

```
# apt install hardlink
```

podemos usar hardlink con *-tnv* para que nos diga los archivos que puede procesar:

```
$ hardlink -tnv temp
```

si nos satisface, ahora hacemos el cambio de los archivos duplicados por ligas simbólicas duras:

```
$ hardlink -tv temp
```

Para conocer el número de enlaces que tiene un determinado archivo, podemos usar:

```
$ ls -l archivo
```

o

```
$ stat archivo
```

Descargar archivos de la red una actividad común es la descarga de archivos de red (*.iso*, *.deb*, etc.) para ello hay varios programas como: *wget, uget, curl, lftp, woof, httpie*, entre otros, su uso básico es el siguiente:

```
$ curl http://132.248.182.159/acl/hcl/HerramientasComputacionalesEnLinux.pdf
```

descargará este texto. Se pueden descargar archivos y/o directorios de los protocolos *HTTP, HTTPS, FTP, POP3, SMTP* y *Telnet*. Además es posible interactuar con APIs usando los métodos: *DELETE, GET, POST* y *PUT*

Uso del comando awk Visualiza solo las líneas que son de 65 caracteres o más:

```
$ cat archivo.txt | awk "length > 65"
```

Conocer todas las ligas y a quien apuntan desde nuestra actul trayectoria, usamos:

```
$ find . -type l -print | xargs ls -ld | awk '{print $9 $10 $11}'
```

Nos mostrarán todos los usuarios que tiene el sistema, los cuales están dados de alta en el archivo del sistema /etc/passwd

```
$ awk -F':' '{ print $1 }' /etc/passwd
```

Nos indicará cuantos procesos tiene corriendo cada usuario:

```
$ ps -ef | awk '{print$1}' | sort | uniq -c | sort -nr
```

Nos muestra la primera y última línea de un archivo:

```
$ awk 'NR==1{print} END {print}' archivo.txt
```

o

```
$ awk 'NR==1;END {print}' archivo.txt
```

Programa *Buscar* para encontrar los archivos y directorios modificados en una determinada trayectoria [$1] en los últimos días [$2]:

```
#!/bin/bash
# Encuentra los archivos modificados en los ultimos $2 dias
if [ -z "$1" ]; then
    echo uso: $0 [directorio] [dias]
    exit
fi
find $1 -type f -mtime -$2 -exec ls -gGh –full-time '{}' \; | cut
-d ' ' -f 4,5,7
```

por ejemplo, para buscar desde /home los archivos modificados desde el último día, usamos:

```
$ ./Buscar /home 1
```

Programa *Diferencia* para encontrar la diferencia entre los archivos y directorios de dos trayectorias [$1] y [$2]:

```
#!/bin/bash
# Encuentra las diferencias entre dos directorios y subdirectorios
if [ -z "$1" ]; then
    echo uso: $0 [directorio1] [directorio2]
    exit
fi
diff <(cd $1 && find | sort) <(cd $2 && find | sort)
```

por ejemplo, buscar la diferencia de contenido entre /home/user/a1 y /home/user/b3, usamos:

```
$ ./Diferencia /home/user/a1 /home/user/b3
```

Ejemplo de uso de rsync ejemplo de uso de rsync en el cual el usuario podría indicar cual función ejecutar dentro del Script:

```
#!/bin/bash
sync_root(){
     spath="/srv/www/unam/https"
     echo "Running rsync..."
     rsync -ar $spath/* root@backup.unam.mx:$spath
}
case "$1" in
     *unam*) sync_root ;;
     *) echo "Error: El dominio no existe.";;
esac
```

Intentar sincronizar con srync hasta lograrlo al intentar sincronizar una carpeta usando *rsync*, puede que por cualquier razón no se logre completar la operación, por ello podemos usar el comando *until* para que siga intentando la sincronización por *rsync* a través de una conexión no confiable. Cuando *rsync* falla, se iniciará un reintento después de 60 segundos. Cuando se complete *rsync*, el ciclo terminará, para ello usamos:

```
$ until rsync -avy –delete-after /volume1/backup/ \
root@backup.unam.mx:.; do sleep 60; done
```

Programa Respalda para generar el respaldo de la trayectoria indicada [$2] con el nombre [$1] (véase 5.2):

```
#!/bin/bash
#Respalda el contenido dado $2 con el nombre $1
if [ -z "$1" ]; then
    echo uso: $0 [Archivo].tar [Archivo o Directorio]
    exit
fi
# Variables de trabajo
A=$1
B=$(date +%Y%m%d)
# Genera el archivo TAR
echo Generando el archivo TAR ...
shift 1
tar -zcvpf $A-$B.tar.gz $*
# Visualiza el nuevo contenido
/bin/ls -al –color=tty
```

por ejemplo, para respaldar el contenido de /home con el nombre respaldo, usamos:

```
$ ./Respalda respaldo /home
```

Programa para un menú este ejemplo muestra un menú del cual el usuario puede elegir entre las diferentes opciones, supongamos que lo llamamos menu y tiene el siguiente código:

```
#!/bin/bash
PS3='Elige tu comida favorita: '
comida=("Pizza" "Pho" "Tacos" "Salir")
select op in "${comida[@]}"; do
  case $op in
    "Pizza")
      echo "Se come en el mundo 1000 acres de $op cada dia"
      # opcional llamar a funcion o correr mas comandos
      break
      ;;
    "Pho")
      echo "$op es un tipo de sopa Vietnamita"
      # opcional llamar a funcion o correr mas comandos
      break
      ;;
    "Tacos")
      echo "Se comen al año 3 mil millones de $op cada año"
      # opcional llamar a funcion o correr mas comandos
      break
      ;;
    "Salir")
      echo "Usuario solicito salir"
      exit
      ;;
    *) echo "Invalida la opcion $REPLY";;
  esac
done
```

y lo ejecutamos usando:

```
$ ./menu
```

Jugar a viborita podemos usar Bash para jugar, usando las flechas podemos mover la barra dentro de toda la terminal usando el código:

```
$ reset;clear;x=$[COLUMNS/2];y=$LINES;u=0;v=-1;while :;do
read -sr -t0.02 -n3 d; case "${d:2:1}" in A) v=-1;u=0;;B)v=1;u=0;;C)
v=0;u=1;;D) v=0;u=-1;;esac;s=$x;t=$y;x=$[x+u];y=$[y+v];printf
"\033[%s;%sH\033[46m \033[0m\033[%s;%sH\033[44m \033[0m\033[0;0H"
$y $x $t $s;sleep .01;done
```

una versión más completa es:

```
$ clear;x=$(($COLUMNS/2)); y=$LINES;unset spacesused[*];declare
-a spacesused;xdir=0;ydir=-1;while true ; do read -s -r -t0.02 -
n3 direction ; case "${direction:2:1}" in A) ydir=-1;xdir=0 ;;
B)ydir=1;xdir=0 ;; C) ydir=0;xdir=1 ;; D) ydir=0;xdir=-1 ;;
esac ; space=$(( $COLUMNS * $y + $x )) ; if [[ "${space-
sused[$space]}" == "1" || $x -gt $COLUMNS || $x -lt 0 || $y -gt
$LINES || $y -lt 0 ]]; then printf '\033[%d;%dHBOOM!\nEND
OF LINE.\n' $y $x; break ; else spacesused[$space]=1 ; fi ;
ox=$x; oy=$y ; x=$(( $x + $xdir )) ; y=$(( $y + $ydir ))
; printf "\033[%s;%sH\033[46m \033[0m\033[%s;%sH\033[44m
\033[0m\033[0;0H" $y $x $oy $ox ; sleep 0.01 ; done
```

Generar disco USB a partir de un archivo ISO es posible generar un USB de cualquier imagen *ISO* descargada de la red, para ello primero debemos conocer el punto de montaje después de insertar la *USB*, mediante:

```
$ lsblk
```

o

```
# fdisk -l
```

o

```
dmseg | grep -i usb
```

si suponemos que el punto de montaje es:/dev/sdb, entonces procederemos a desmontarlo mediante:

```
$ umount /dev/sdb
```

ahora podemos usar el comando *dd* para generar el *USB* del archivo *ISO* descargado, mediante:

```
# dd if=/ruta-del-iso of=/dev/sdb bs=512M; sync
```

si queremos conocer el progreso de la generación del ISO, podemos usar:

```
# dd if=/ruta-del-iso of=/dev/sdb bs=512M status=progress;
sync
```

6 Procesamiento de Textos, Imágenes y PDF

Existe una gran cantidad de usos para los programas de edición de texto, pero en las carreras de Ciencias e Ingenierías, la edición de textos con tipografía científica es común. Es por ello que la gran mayoría de los procesadores de textos más usados no proporcionan las herramientas necesarias para incluir en el texto fórmulas y/o notación matemática. En caso de proveer dichas herramientas, muchas de ellas son de uso tedioso, pues están diseñadas para uso ocasional.

Para subsanar este hecho, existen herramientas y editores hechos a exprofeso, para permitir la edición de textos científicos en los cuales numerar ecuaciones, usar tipografía matemática, manipular bibliografía y referencias cruzadas es una tarea sencilla de realizar.

Existe una gran variedad de paquetes para la edición de textos científicos -los cuales existen tanto en las plataformas de Windows, Linux, Mac-, entre los que destacan:

- Editor de ecuaciones integrado en Word en Microsoft Office (véase [26])
- MathType para Word en Microsoft Office para Windows (véase [38])
- Scientific WorkPlace LaTeXpara Windows (véase [39])
- Gummi LaTeX(véase [40])
- Kile LaTeX(véase [41])
- LED LaTeX(véase [42])
- LyX LaTeX(véase [43])
- Texmaker LaTeX(véase [45])
- TeXnicCenter LaTeX(véase [46])
- TextPad LaTeX(véase [47])
- TeXstudio LaTeX(véase [48])
- WinEdt LaTeX(véase [49])
- Formula de Libre Office (véase [28])

- Math de OpenOffice (véase [27])
- Formula de Calligra (véase [44])

Salvo para los productos de Microsoft Office, el resto de los paquetes tienen una curva de aprendizaje de media a alta, pero en contraste permiten desarrollar textos y gráficos con tipografía científica de alta calidad.

Instalación de Procesadores de Texto e IDEs Existen diversas versiones de paquetes para procesar texto en Linux, para instalar las más comunes en Debian GNU/Linux es necesario hacer:

```
# apt install science-typesetting texlive-science texstudio \
pandoc texmaker inkscape kile gummi texstudio \
texlive-latex-basetexlive-latex-recommended latexila lyx \
texlive-extra-utils texlive-full latexila libreoffice calligra \
abiword evince gnumeric kexi xpdf okular medit texworks \
libreoffice imagemagick djview4 okular gv zathura diffpdf \
mupdf pdf-presenter-console evince atril pdfcrack qpdf \
pdfsam pdfshuffler pdfmod pdfposter pdfchain pdf2djvu \
catdoc chktex cxref cxref-doc latex2rtf antiword \
unoconv a2ps rst2pdf xchm archmage htmldoc \
qpdfview-ps-plugin qpdfview kchmviewer pdfgrep \
qpdfview-djvu-plugin pdfarranger ispanish \
wspanish myspell-es myspell-en-us texlive-lang-spanish \
translate-Shell pdftk poppler-utils pdf2svg bookletimposer \
converseen
```

También podemos instalar más de 400 distintos tipos de Fonts:

```
# apt install ^fonts-* ^ttf-* ^xfonts-*
```

y los Fonts de Microsoft:

```
# apt install ttf-mscorefonts-installer fonts-arkpandora
```

para verificar las fuentes tipográficas instaladas (el paquete es fontconfig y viene instalado por omisión) en la máquina usamos:

```
$ fc-list
$ fc-list : family
$ fc-list : family style
```

Aprender a Trabajar en LaTeX En la red existen múltiples sitios especializados y una amplia bibliografía para aprender a programar cada uno de los distintos aspectos de LaTeX, nosotros hemos seleccionado diversos textos que ponemos a su disposición en:

Latex

6.1 Trabajando con LaTeX

LaTeX es un poderoso paquete que puede hacer múltiples operaciones por nosotros, a continuación haremos un viaje por sus capacidades, pero sólo es una muestra de ellas, para más información revise el manual y ejemplos en línea. Supongamos que necesitamos hacer el siguiente texto:

Pequeño Gran Teorema

Alrededor de 1630 el jurista y matemático Pier de Fermat estudió la Aritmética de Diofanto, en una traducción latina de Claude Bachet (1581-1638), publicada en 1621. El problema ocho del libro II de la Aritmética propone descomponer un número al cuadrado en suma de dos números al cuadrado y Fermat anotó en el margen la imposibilidad de hacer algo similar para los cubos, las cuartas potencias, o cualquier potencia más alta, afirmando tener una prueba notable de este hecho que no poda escribir en los márgenes estrechos del libro (Hanc marginis exiguitas non caperet).

Así, esta conjetura nació y en los siglos que se sucedieron tras su muerte, otros matemáticos fueron abordando y solucionando cada uno de los problemas que Fermat había garabateado... Hasta que sólo quedó uno por resolver:

"No existe ningún número entero positivo mayor que 2 que satisfaga la ecuación $a^n + b^n = c^n$, donde n es dicho número"

Esta conjetura fue bautizada con el nombre de "el último teorema de Fermat", precisamente porque era la última proposición de este autor que nadie había podido refutar o verificar.

Pero el pasado 15 de marzo de 1994 se entregó el premio Abel, el premio Nobel de matemáticas, a Andrew Wiles por haber confirmado una conjetura matemática cuya validez no había podido ser demostrada desde que se propuso en 1642.

Figura 4: Ejemplo a desarrollar en LaTeX.

Ahora tenemos que tomar la decisión de con que haremos la edición del texto, en Linux hay varias opciones que podemos usar, la primera es usar

un editor de texto plano[103] en la terminal de línea de comandos (véase 3.4) como *nano*, *vi o micro*; otra opción es usar un editor gráfico de texto plano[104] como *scite* o *kate*; o los editores especializados para LaTeX [105] como *Gummi* o *Texmaker*.

Entonces es necesario teclear el siguiente texto que lo generará:

```
\documentclass[letterpaper,12pt]{article}

\pagestyle{empty}

\begin{document}

\section*{Peque\~no Gran Teorema}

Alrededor de 1630 el jurista y matem\'atico Pier de Fermat estudi\'o
la Arit-m\'etica de Diofanto, en una traducci\'on latina de Claude
Bachet (1581-1638), publicada en 1621. El problema ocho del libro II
de la Aritm\'etica propone descomponer un n\'umero al cuadrado en
suma de dos n\'umeros al cuadrado y Fermat anot\'o en el margen
la imposibilidad de hacer algo similar para los cubos, las cuartas po-
tencias, o cualquier potencia m\'as alta, afirmando tener una prueba
notable de este hecho que no pod\´ia escribir en los m\'argenes estre-
chos del libro (Hanc marginis exiguitas non caperet).

As\'i, esta conjetura naci\'o y en los siglos que se sucedieron tras su
muerte, otros matem\'aticos fueron abordando y solucionando cada
uno de los problemas que Fermat hab\'ia garabateado... Hasta que
s\'olo qued\'o uno por resolver:

"No existe ning\'un n\'umero entero positivo mayor que 2 que sat-
isfaga la ecuaci\'on $a^{n}+b^{n}=c^{n}$, donde $n$ es dicho n\'umero"
```

[103] Otras opciones son: Diakonos, Jet, JOE, LE, Mined, Nano, Nice Editor, Pico, SETEdit, Vim, FTE

[104] Otras opciones son: Gedit, JEdit, Nedit, Medit, Kscope, Editra, Kate, Kwrite, Leafpad, Mousepad, Anjunta, Tea, Pluma, GVim, Emacs, Atom, Geany, Glade, Notepadqq, Scribes, Sublime Text

[105] Otras opciones son: Scientific WorkPlace, Kile, LED, LyX, TeXnicCenter, TextPad, TeXstudio, WinEdt, Formula de Libre Office, Math de OpenOffice, Formula de Calligra

Esta conjetura fue bautizada con el nombre de "el \'ultimo teorema de Fermat", precisamente porque era la \'ultima proposici\'on de este autor que nadie hab\'ia podido refutar o verificar.

Pero el pasado 15 de marzo de 1994 se entreg\'o el premio Abel, el premio Nobel de matem\'aticas, a Andrew Wiles por haber confirmado una conjetura matem\'atica cuya validez no hab\'ia podido ser demostrada desde que se propuso en 1642.

\end{document}

6.1.1 Compilando un Documento

Ahora hecho el documento, con nombre *teorema.tex* necesitamos compilar y generar el archivo de salida que usaremos para visualizar el resultado. Primero debemos tener todo lo necesario para ello, en Debian GNU/Linux se instala LaTeX mediante:

```
# apt install science-typesetting texlive-science
```

Si usamos editores para LaTeX especializados, ellos generan la salida gráfica en formato PDF[106] o DVI[107], sin requerir otra cosa que solicitarla. Si trabajamos en la terminal, necesitamos compilar y generar el archivo de visualización, para ello tenemos algunas opciones:

- Si la salida la necesitamos en DVI, podemos compilar usando:

  ```
  $ latex teorema.tex
  ```

  y visualizamos mediante:

  ```
  $ xdvi teorema.dvi
  ```

- Si la salida la necesitamos en PDF, podemos compilar usando:

[106] El formato de documento portátil (Portable Document Format PDF) es un formato de almacenamiento de documentos digitales independientes de plataformas de Software o Hardware, este formato es de tipo compuesto (imagen vectorial, mapa de bits y texto).

[107] EL formato DVI (DeVice Independent) es un formato de archivo utilizado como salida por el programa de tipografía TeX. Por lo general el fichero DVI es utilizado como entrada por un postprocesador para generar archivos PostScript o PDF.

$ texi2pdf teorema.tex

y visualizamos mediante:

xpdf teorema.pdf

El archivo *teorema.tex* y su salida *teorema.pdf* se puede descargar de:

Ejemplitos

Por último, si necesitamos trabajar con LaTeX y tenemos acceso a un equipo de cómputo al que no podemos instalar paquetería, existen diferentes servicios Web que permiten editar, compilar y ejecutar código LaTeX, esto en aras de que el usuario cuente con algún sistema de acceso a red y un navegador pueda trabajar sin hacer instalación alguna en su equipo de cómputo, tableta o teléfono celular. Por ejemplo, en la dirección: https://es.sharelatex.com/

6.1.2 Estructura del Documento

Un documento en LaTeX tiene dos partes principales: el preámbulo y el cuerpo del documento.

El preámbulo es iniciado por la instrucción *documentclass*, mientras que el cuerpo del documento esta delimitado por los comandos *begin{document}* y *end{document}*.

El esqueleto vacío de un documento en LaTeX se ve así:

```
\documentclass{article}
% pre'ambulo
\begin{document}
% cuerpo del documento
\end{document}
```

Comandos Como puedes empezar a observar, los comandos en LaTeX inician con una diagonal invertida: *comando*, mientras que los comentarios (texto que no aparecerá en el documento final y sólo sirve para agregar notas dentro del código) se escriben después de un signo de porcentaje: *% comentario*. Algunos comandos tienen parámetros obligatorios que se escriben entre llaves *{..}*. Algunos otros llevan también parámetros opcionales que van entre corchetes *[..]*.

Por ejemplo el parámetro *article* en *documentclass* indica a LaTeX que el documento se trata de un artículo y utilizará entonces el formato adecuado. Otras opciones pueden ser *book, report, letter y slides* que sirven para hacer libros, reportes, cartas, y diapositivas respectivamente.

Algunos parámetros opcionales para *documentclass* son *11pt* y *12pt* que especifican un tipo de letra más grande (el normal es de *10pt*), *twocolumn* que escribe el texto a dos columnas y *twoside* que ajusta los márgenes del documento para imprimir a dos caras. Por ejemplo, para escribir un reporte con letra tamaño 12pt y a dos columnas entonces se escribe el comando

```
\documentclass[12pt,twocolumn]{report}
```

Preámbulo En el preámbulo se pueden incluir instrucciones para activar paquetes que agregan funciones adicionales a LaTeX, así como datos generales sobre el documento que estás escribiendo. Un preámbulo típico podría verse así:

```
\documentclass{article}
\usepackage{lmodern}
\usepackage[T1]{fontenc}
\usepackage[spanish,activeacute]{babel}
\usepackage{mathtools}
\title{Ejemplo de \LaTeX{}}
\author{Nombre del autor}
\date{29 de diciembre de 2019}
```

Los dos primeros paquetes, *lmodern* y *fontenc*, se utilizan para mejorar el soporte de caracteres especiales en la fuente (tipo de letra) que se usará en tu documento. Por ejemplo para que puedas copiar y pegar texto correctamente desde el documento PDF que produzcas al final.

El siguiente paquete incluido es *babel* con la opción *spanish* que traduce algunas de las etiquetas usadas por LaTeX, y agrega opciones especiales para redactar documentos en español. Si no incluyes este paquete, o cambias *spanish* por *english*, LaTeX supondrá que estas escribiendo en inglés.

El último paquete incluido es *mathtools* que agrega algunos comandos y funciones especiales para facilitar la escritura de fórmulas y ecuaciones matemáticas.

Hay muchos otros paquetes que puedes incluir y que agregan funciones adicionales a tu documento, pero estos son los básicos que siempre es una buena idea incluir. Algunos otros paquetes típicos son: *hyperref*, que permite incluir ligas en tu documento, *biblatex*, para administrar tu bibliografía, o *tikz*, para crear todo tipo de ilustraciones.

Finalmente los campos *\title, \author y \date* especifican los datos que irán en el encabezado del documento. Normalmente, de hecho, no es necesario incluir el comando *\date* pues LaTeX usará en su lugar la fecha actual cuando generes tu documento.

Cuerpo del Documento En el cuerpo del documento es donde escribes todo el texto que quieras que aparezca en el documento final. Usualmente se inicia con el comando *\maketitle* que se encarga de escribir los datos del título con la información que indicaste en el preámbulo.

Todo el texto normal se escribe tal cual[108]. Si quieres decir "Hola" simplemente escribe: Hola.

Acentos y Signos Especiales Como podrás ver en los ejemplos anteriores, los acentos no se pueden escribir de manera directa dentro del código. Cuando necesites escribir una letra con acento como la "á" deberás escribir 'a y en lugar de "ñ" escribe 'n. Para las mayúsculas funciona lo mismo sólo utiliza la letra mayúscula adecuada.

La opción *activeacute* de *babel* te permite usar este método "corto" para escribir acentos. Esta opción, sin embargo, no funciona en el preámbulo. Para poner acentos en el preámbulo debes usar la forma larga en la que escribes *\'a* para obtener la letra acentuada "á", *\'e* para la letra "é", y *\~n* para la "ñ".

Otros símbolos que requieren atención son: ¡ y ¿ para producir "¡" y "¿", así como las comillas 'sencillas' y "dobles" que producen 'sencillas' y "dobles".

[108] a) Si dejas varios espacios en blanco entre palabras, LaTeX los toma como si fueran uno solo.

b) No es necesario dejar espacios al inicio de un párrafo para indicar una sangría, LaTeX ignora estos espacios y ajusta las sangrías adecuadas de manera automática.

c) Para separar dos párrafos simplemente deja una línea en blanco entre un párrafo y el siguiente, el simple fin de línea no hace la separación.

d) Varias líneas en blanco juntas valen lo mismo que una sola.

Fórmulas Matemáticas La primera forma de escribir fórmulas matemáticas es el modo en línea que inserta un símbolo o una fórmula sencilla dentro de la redacción de un párrafo. Este modo se obtiene encerrando entre los signos: *$..$* el contenido matemático, un ejemplo[109]:

```
... si $x = 0$ entonces $y^{2} = 4p + 7$, pero si damos otro
valor a $x$ no s'e que pase ...
```

El otro modo para insertar texto matemático es en una fórmula destacada. Este modo es para ecuaciones más grandes que, por ejemplo si incluye sumatorias o límites, no se verían bien incrustadas dentro de un párrafo. Una fórmula destacada lo que hace es abrir un espacio amplio en medio del párrafo y centrar la ecuación en la página[110]. Una forma de lograr esto es usando los comandos *\begin{equation}* y *\end{equation}* o, si no te interesa ir numerando las ecuaciones, la variante *equation**.

```
    Y despu'es de experimentar mucho con diferentes t'ecnicas
resulta que la ecuaci'on
    \begin{equation}
    w = \sum_{i=1}^{n} (x_{i}+y_{i})^{2}
    \end{equation}
    es muy importante.
    ... y como sabemos que
    \begin{equation*}
    \lim_{x \to 0} (x^{2} + 2x + 4) = 4
    \end{equation*}
    se concluye que...
```

Una gran familia de comandos que puedes utilizar son las letras griegas. Así como *\pi* puedes encontrar *\alpha, \lambda*, etc. Para obtener las letras griegas mayúsculas capitaliza la primera letra, por ejemplo en *\Omega* o *\Pi*.

Otra familia de comandos corresponden a nombres de funciones[111] como *\sin, \log, \lim*, etc. La guía completa de todos los símbolos que puedes

[109] Una de las primeras cosas que notarás es que las letras en el entorno matemático aparecen en itálicas y que puedes escribir exponentes cómo en *y^{2}*. Ojo, sin embargo, que nunca debes de usar el entorno matemático para escribir palabras en cursivas.

[110] Nota que, en el código de LaTeX, no hay separación entre la ecuación y el texto del párrafo. Esto es porque la ecuación es parte de la redacción del párrafo.

[111] Observa que no se obtiene el resultado correcto en la tipografía si escribes únicamente *sin*; eso es s por i por n, y no la función 'seno' que obtienes con *$\sin$*.

utilizar en LaTeX es un libro que se llama The Comprehensive LaTeX Symbol List de Scott Pakin. Algunos editores, como TeXnicCenter para Windows, tienen barras con botones para escribir los comandos dando Click sobre el símbolo o construcción que necesites.

Estructura del Documento Parte de la ideología de LaTeX es que el autor de los documentos no debe preocuparse por el formato o la apariencia que tendrá el documento impreso en papel, ya que eso es tarea de LaTeX (o de un diseñador de formatos). El autor debe preocuparse sólo por el contenido y la estructura de su documento. Siguiendo esta ideología, esta guía no muestra comandos para manipular el formato del texto. Si en algún lugar ya aprendiste esos comandos lo mejor es que (cuando escribes el cuerpo de un documento) te olvides de que existen.

Por ejemplo, un comando importante es *\emph{..}* que te permite agregar énfasis a palabras u oraciones. Normalmente el resultado es que el texto aparece en itálicas. Sin embargo, no debes pensar en *\emph{..}* como un comando para poner itálicas (¡eso es pensar en formato!) sino como un comando para agregar énfasis (¡eso es pensar en contenido!). Diferentes estilos de documentos podrían incluso agregar énfasis usando diferentes formatos, por ejemplo subrayando o escribiendo en rojo. Un autor decide qué enfatizar, y es tarea del editor el decidir cómo hacerlo.

Otra familia importante de comandos te permite poner títulos y dividir tu documento en secciones. Una de las ventajas importantes de estos comandos es que cosas como el índice y tablas de contenido se hacen de forma automática al utilizarlos:

```
\part{..}
\chapter{..}
\section{..}
\subsection{..}
\subsubsection{..}
```

Los comandos pueden variar según el estilo de documento usado. Por ejemplo un artículo (*article*) suele dividirse comenzando por *\section{..}*, mientras que un libro (*book*) puede incluir *\part{..}* o *\chapter{..}*.

Para conseguir que aparezca el índice en tu documento usa el comando *\tableofcontents*, por ejemplo después de *\maketitle*, y compila dos o tres veces.

Insertar Figuras en LaTeX suele ser una de las principales causas de dolores de cabeza para quienes nos enfrentamos a esta tarea por primera vez. Y la situación se complica con la diversidad e incompatibilidad que existe entre formatos para almacenar gráficos. Además, pareciera que LaTeX no es muy amigable con formatos tipo Web (JPEG, GIF) a los que podríamos estar más acostumbrados.

Antes de comenzar, y para evitar posibles confusiones, es necesario ponernos de acuerdo con los términos, y lo que significan:

- Un gráfico es cualquier dibujo, ilustración, imagen, diagrama, fotografía, gráfica de puntos o líneas, histograma, diagrama de sectores, etc.; que podrías querer insertar en tu documento para ilustrar o clarificar alguna idea. La mayor parte de esta guía trata sobre cómo preparar los gráficos para incluirlos en tu documento.

- Una figura es la forma que normalmente se utiliza para insertar un gráfico dentro de un documento. Las figuras están compuestas por un gráfico y un título (que no es parte del gráfico), así como de una numeración que indica la secuencia de figuras dentro del documento (Figura 1, Figura 2, ...).

Ya que tienes listo el gráfico en un formato adecuado, agrega en el preámbulo de tu documento principal (antes de *\begin{document}*) la instrucción:

```
\usepackage{graphicx}
```

Este paquete, permite incluir gráficos externos en tu documento. Luego, en algún lugar cercano en donde quieras que se coloque tu figura agrega el siguiente código:

```
\begin{figure}
\centering
\includegraphics{grafico}
\caption{Mi Figura}
\label{fig:ejemplo}
\end{figure}
```

El comando *\includegraphics{..}* indica el nombre del archivo que contiene el gráfico que quieres insertar. Observa que no es necesario incluir la extensión del archivo (JPEG, PNG o PDF), LaTeX buscará y utilizará el archivo apropiado.

Tamaño de la figura Si la figura resulta demasiado grande o pequeña, puedes agregar opciones al comando para incluir el gráfico. Por ejemplo:

```
\includegraphics[width=0.7\textwidth]{migrafico}
```

esta opción ajusta el ancho de la figura al 70% del ancho del texto que cabe en una página. Puedes, por supuesto, modificar el valor 0.7 por cualquier según tus necesidades. Hay muchas más opciones que provee el comando \includegraphics, si tienes curiosidad puedes leer la guía Using Imported Graphics in LaTeX2e.

Posición de la figura Una de las tareas de LaTeX es encontrar el lugar más adecuado para colocar tu figura dentro del documento. Esto suele ocasionar sorpresa y un poco de incomodidad sobre todo en usuarios principiantes, "¡pero yo quiero mi figura aquí! ¿por qué LaTeX la pone allá?". La mejor solución realmente es relajarse, cambiar de actitud y dejar que LaTeX haga su trabajo. Evita usar redacciones del tipo:

```
... como en la siguiente figura:
```

y utiliza en su lugar las bondades de LaTeX:

```
... como en la Figura~\ref{fig:ejemplo}.
```

Esto hace que la referencia no dependa del lugar donde aparezca la figura y, finalmente, todo se ve mucho más elegante.

La opción *draft* para borradores Un problema común que nos puede ocurrir es que la gráfica no aparece y, en su lugar, sólo vemos una caja con el nombre del archivo del gráfico. Esto ocurre porque se tiene activada la opción *draft*, ya sea como opción del paquete en *\usepackage[draft]{graphicx}* o como una opción global para todo el documento en \documentclass. Esta opción puede ser útil para visualizar documentos más rápidamente si es que tienes demasiados gráficos. Sin embargo recuerda quitar esta opción, o cambiarla al final, si quieres que aparezcan los gráficos en el documento.

6.1.3 Hacer Presentaciones

Para hacer presentaciones donde se requiera mos-trar expresiones matemáticas es preferible usar Beamer (véase [50]), el cual es una clase de LaTeX para la creación de presentaciones. Este funciona con pdflatex, dvips, LyX entre otros. Primero debemos tener todo lo necesario para poder trabajar con Beamer, en Debian GNU/Linux se instala LaTeX mediante:

```
# apt install -Rlatex-beamer texlive-full
```

Entonces tecleamos el siguiente texto que lo generará una presentación:

```
\documentclass{beamer}
\mode<presentation> {
 \usetheme{Darmstadt}
 \setbeamercovered{transparent}
}
\usepackage[spanish]{babel}
\usepackage[utf8]{inputenc}
\usepackage{pdfpages}
\usepackage{alltt}
\usepackage{verbatim}
\usepackage{hyperref}
\title{M\'aquinas Virtuales y sus M\'ultiples Usos}
\author[Karla y Antonio]{Karla Gonz\ ´alez y Antonio Carrillo —
{\tt aula@ciencias.unam.mx}}
\institute[FC - UNAM;]
 {Facultad de Ciencias, UNAM}
\date[Intersemestral]
 {Curso intersemestral 2020\\
 11 de enero, 2020}
\AtBeginSection[]
```

```
{
 \begin{frame}<beamer>
 \frametitle{Temas}
 \tableofcontents[currentsection]
 \end{frame}
}
\begin{document}
\begin{frame}
 \titlepage
\end{frame}
\section[¿Qu\'e es]{¿Qu\'e es la virtualizaci\'on?}
\begin{frame}
 \frametitle{¿Qu\'e significa {\it virtualizar} en el c\'omputo?}
 \begin{itemize}
 \item Proveer de algo que no est\'a all\'i, aunque parece estarlo
 \item Ofrecer y mantener una ilusi\'on, un truco de magia
 \end{itemize}
 \begin{center}
 La {\em virtualizaci\'on} es, en t\'erminos generales, es ofrecer
 recursos que no existen en realidad — Y mantener la ilusi\'on, tan
 bien como sea posible.
 \end{center}
\end{frame}
\section[Gracias]{Gracias}
\begin{frame}
 \frametitle{Gracias}
 \begin{center}
 Gracias por su atenci\'on
```

```
\end{center}
\end{frame}
\end{document}
```

Si trabajamos en la terminal, necesitamos compilar y generar el archivo de visualización, para ello tenemos algunas opciones.

Si la salida la necesitamos en DVI, podemos compilar usando:

```
$ latex Presentacion01.tex
```

y visualizamos mediante:

```
$ xdvi Presentacion01.dvi
```

Si la salida la necesitamos en PDF, podemos compilar usando:

```
$ texi2pdf Presentacion01.tex
```

y visualizamos mediante:

```
xpdf Presentacion01.pdf
```

El esqueleto de la presentación Con LaTeX se puede hacer fácilmente una presentación[112] usando la clase *{beamer}*. Cada transparencia se define entre los comandos *\begin{frame}* y *\end{frame}*, por ejemplo:

[112] Unos pequeños consejos sobre presentaciones:

- Una presentación tiene que ser sencilla y clara. Con una (o dos) figuras por transparencia. Con poco texto y letra grande, (4 o 5 líneas es ya suficiente), o mejor, con solo los puntos y palabras clave. Nada de un montón de texto pequeñito y todo apretado, como si fuera un libro. Recuerda que una presentación "es para que el público vea, no para que tú leas".
- Una presentación tiene que ser agradable y elegante. Una presentación en colores es más visible que una en blanco y negro. Pero que se aprecie el buen gusto que tienes.
- También es bueno que una presentación sea corta. Tratar de meter demasiada información, solo consigue que el público se pierda, aburra y desconecte.
- Recuerda que tienes que practicar. "La práctica te da confianza. La confianza te da profesionalidad."

```
\documentclass{beamer}
\usepackage[spanish]{babel}
\usepackage[latin1]{inputenc}
\begin{document}
\begin{frame}
 \frametitle{Marsupiales}
 Canguro, Koala, Wombat...
\end{frame}
\end{document}
```

Colores y formato LaTeX tiene varios formatos y combinaciones de colores ya definidos. Para elegir un formato, se utilizan los siguientes comandos:

- *\usetheme{Warsaw}* Define el formato.
- *\usecolortheme{crane}* Define la combinación de colores.
- *\useoutertheme{shadow}* Define el encabezado y pie de página. Puedes elegir entre: *{infolines}, {miniframes}, {shadow}, {sidebar}, {smoothbars}, {smoothtree}, {split}, {tree}...*
- \useinnertheme{rectangles} Define el formato de los puntos. Puedes elegir entre: *{circles}, {inmargin}, {rectangles}, {rouded}...*

Ejemplo:

- El tono de voz y los gestos son también muy importantes. Un tono fijo, monótono, constante... es aburrido y hace que la gente se desconecte y se duerma. Los buenos oradores, cambian la fuerza y el tono de voz. Hacen pausas, se mueven, gesticulan. Y consiguen que sus presentaciones sean dinámicos y atrayentes.
- Por último, sonríe, y mira al público. :-)

```
\usetheme{Warsaw}
\usecolortheme{crane}
\useoutertheme{shadow}
\useinnertheme{rectangles}
\begin{document}
\title[Animales]{Animales de todo tipo}
\subtitle{Dando nombres a los animales}
\author[Adan, Eva, Serpiente]{
A. Ad\´an$^{1}$ \and E. Eva$^{2}$ \and S. Serpiente$^{3}$}
\institute[EDEN \& HELL]{
 $^{1-2}$
 Universidad de Ed\´en\\
 Al lado del manzano, Para\´iso
 \and
 $^{3}$
 Universidad del Infierno\\
 Inframundo, 666, Tierra
 \and
 \texttt{\{$^{1}$eva, $^{2}$adan\}@paraiso.com, $^{3}$serpiente@infierno.com}
}
\date{\today}
\begin{document}
\frame{\titlepage}
\end{document}
```

Índice de secciones. LaTeX permite crear fácilmente el índice de nuestra presentación, con el comando *tableofcontents, ejemplo*:

```
\begin{frame}
 \frametitle{\´Indice}
 \tableofcontents
\end{frame}
\section{Animales}
\subsection{Marsupiales}
\begin{frame}
 \frametitle{Marsupiales}
 Canguro, Koala, Wombat...
\end{frame}
\subsection{Marinos}
\section{Plantas}
\subsection{Flores}
\subsection{Árboles}
```

También se puede que, cada vez que pasamos de sección y subsección, nos vuelva a aparecer el índice, marcando el punto por el que nos llevamos. Para ello, bastaría añadir el siguiente código, que hay que poner antes de *begin{document}*, ejemplo:

```
\AtBeginSection{
\begin{frame}
 \frametitle{\´Indice}
 \tableofcontents[currentsection]
\end{frame}
}
\AtBeginSubsection{
\begin{frame}
```

```
 \frametitle{\´Indice}
 \tableofcontents[currentsection,currentsubsection]
\end{frame}
}
\begin{document}
```

Si nuestro índice es muy largo, quizás sea mejor ponerlo en dos columnas. Para ello, necesitamos cargar el paquete *{multicol}*, y utilizar el código siguiente. Además, también recomiendo definir un encabezado que ocupe poco espacio, como *{tree}*, ejemplo:

```
\usepackage{multicol}
\useoutertheme{tree}
\AtBeginSection{
\begin{frame}
 \frametitle{\´Indice}
 \begin{multicols}{2}
 \tableofcontents[currentsection]
 \end{multicols}
\end{frame}
}
\AtBeginSubsection{
\begin{frame}
 \frametitle{\´Indice}
 \begin{multicols}{2}
 \tableofcontents[currentsection,currentsubsection]
 \end{multicols}
\end{frame}
}
\begin{document}
```

```
\begin{frame}
 \frametitle{Índice}
 \begin{multicols}{2}
 \tableofcontents
 \end{multicols}
\end{frame}
```

Cajas Cuando queremos agrupar ideas o palabras clave, es muy útil el comando *{block}*, como en el siguiente ejemplo:

```
\begin{frame}
 \frametitle{Marsupiales}
 \begin{block}{Marsupiales en Australia}
 Koala, Canguro, Wombat...
 \end{block}
 \begin{block}{Marsupiales fuera de Australia}
 Oposum, Zarig\ ¨uella...
 \end{block}
\end{frame}
```

Dos Columnas De manera similar a como hicimos con el índice, podemos dividir una transparencia en varias columnas, utilizando el siguiente código. Nótese que, con el comando *\column{x}*, "x" significa la anchura de cada columna, por ejemplo:

```
\begin{frame}
 \frametitle{Animales}
 \begin{columns}[t]
 \column{0.5\textwidth}
```

```
Algunos mam\ ´iferos marinos:

Ballena, Narval, Cachalote...
\column{0.5\textwidth}
Y algunos más:
Morsa, Le\ ´on marino, Foca...
\end{columns}
\end{frame}
```

Animaciones Si queremos que varios puntos, vayan apareciendo de manera secuencial, según vayamos haciendo Click con el ratón, podemos usar el siguiente código. Donde $<a->$ significa que el texto aparecerá desde el Click número $"a"$, hasta el último, por ejemplo:

```
\begin{frame}
\frametitle{Flores}
\begin{itemize}
\item<1->{Rosa}
\item<2->{Azucena}
\item<3->{Margarita}
\end{itemize}
\end{frame}
```

Otra posible animación, consiste en, cuando hagamos Click con el ratón, que un cierto texto cambie de color. Pero sin que aparezca ni desaparezca nada. Para ello, utilizamos el siguiente código:

```
\begin{frame}
\frametitle{Preguntas}
\begin{itemize}
\item \alert<1>{?‘Cuáles ponen huevos?}
```

\item \alert<3>{?‘Cuáles tienen veneno?}

\end{itemize}

Armadillo, \alert<2,4>{Ornitorrinco}, \alert<2>{Equidna}, Pangolín, Erizo.

\end{frame}

6.2 Manipulación de Archivos PDFs

Existen en GNU/Linux múltiples opciones para manipular archivos *PDF*, entre los que destacan por su poder y versatilidad desde la línea de comandos son:

libreoffice entre las muchas cosas que podemos hacer con libreoffice es tratar de convertir un archivo PDF a uno con formato Odt, Doc o Docx, en todos los casos la conversión dependerá de la complejidad del PDF, para el primer caso usamos:

```
$ libreoffice –infilter="writer_pdf_import" –convert_to odt archivo.pdf
```

para el segundo caso:

```
$ libreoffice –infilter="writer_pdf_import" –convert_to doc archivo.pdf
```

para el último caso:

```
$ libreoffice –infilter="writer_pdf_import" –convert_to docx archivo.pdf
```

Pero también podemos convertir un archivo .doc a imagen:

```
$ soffice –convert-to jpg "archivo.doc"
```

o un archivo .txt a imagen:

```
$ soffice –convert-to jpg "archivo.txt"
```

pdftk es un potente programa hecho con Java que permite eliminar o añadir páginas de un PDF, juntar páginas de distintos documentos PDF en uno solo, todo con sencillos comandos. Útil, rápido y muy potente. Para instalarlo basta invocar:

```
# apt-get install pdftk
```

Entre las distintas opciones de uso del comando está el poder unir, separar, cifrar, reparar, rotar, entre otras cosas, para mostrar sus capacidades veamos algunos ejemplos:

- Para unir dos documentos diferentes podemos ejecutar lo siguiente:

  ```
  $ pdftk archivo1.pdf archivo2.pdf cat output salida.pdf
  ```

- También podemos unirlos utilizando etiquetas:

  ```
  $ pdftk A=archivo1.pdf B=archivo2.pdf cat A B output salida.pdf
  ```

- Y por supuesto podemos usar comodines:

  ```
  $ pdftk *.pdf cat output salida.pdf
  ```

- Para separar páginas de varios documentos y crear un documento nuevo con estas hacemos lo siguiente:

  ```
  $ pdftk A=uno.pdf B=dos.pdf cat A1-7 B1-5 output salida.pdf
  ```

- Otro ejemplo con un solo documento:

  ```
  $ pdftk A=archivo1.pdf cat A1-12 A14-end output salida.pdf
  ```

- Eliminar las páginas de la 1 a la 4 y de 21 en adelante de un fichero PDF:

  ```
  $ pdftk fichero.pdf cat 1-4 21-end output ficheros_eliminados.pdf
  ```

- Extraer páginas de un fichero PDF (por ejemplo, extraer páginas 3, 4 y 5 en un nuevo PDF):

$ pdftk fichero.pdf cat 3-5 output fichero_final.pdf[/code]

- Adjuntar archivos a páginas de un fichero PDF:

  $ pdftk fichero.pdf attach_files adjunto1 adjunto2 output fichero_final.pdf

- Extraer los adjuntos de un fichero PDF:

  $ pdftk fichero.pdf unpack_files output directorio/de/salida

- Añadir una marca de agua a un fichero PDF:

  $ pdftk fichero.pdf background watermark.pdf output fichero_final.pdf[/code]

- Para cifrar con una clave de 128 bits (opción por defecto) y restringir todos los permisos (opción por defecto):

  $ pdftk archivo.pdf output archivo_cifrado.pdf owner_pw foopass

- Para cifrar igual que el caso anterior pero asignando una contraseña "miclv" que permite abrir el archivo de salida:

  $ pdftk archivo.pdf output archivo_cifrado.pdf owner_pw foo user_pw miclv

- Igual que el caso anterior pero con permiso de impresión:

  $ pdftk archivo.pdf output archivo_cifrado.pdf owner_pw foo user_pw miclv allow printing

- Para descifrar, i.e. dejar un pdf cifrado en otro sin cifrar:

  $ pdftk asegurado.pdf input_pw foopass output inseguro.pdf

- Para reparar un archivo pdf:

  $ pdftk corrupto.pdf output arreglado.pdf

- Para descomprimir un archivo pdf para su posterior edición en algún editor de texto:

```
$ pdftk midoc.pdf output midoc_desc.pdf uncompress
```

- Para separar cada una de las páginas del documento:

```
$ pdftk in.pdf burst
```

- Para generar un reporte del documento (resulta útil cuando se necesita organizar un índice de un conjunto de un PDF):

```
$ pdftk archivo.pdf dump_data output reporte.txt
```

- Multistamp:

```
$ pdftk fondo.pdf multistamp stamp.pdf output salida.pdf
```

- Stamp:

```
$ pdftk fondo.pdf stamp stamp.pdf output salida.pdf
```

- Giro[113] de todo el fichero 90^o:

```
$ pdftk archivo.pdf cat 1-endeast output salida.pdf
```

- También puedes ponerlo como:

```
$ pdftk archivo.pdf cat 1-endE output salida.pdf
```

- Giro de todo el fichero 180^o:

```
$ pdftk archivo.pdf cat 1-endsouth output salida.pdf
```

- Giro de la página 2 90^o:

```
$ pdftk archivo.pdf cat 2east output salida.pdf
```

[113] Otras opciones son: north: 0, east: 90, south: 180, west: 270, left: -90, right: +90, down: +180

poppler-utils la herramienta pdfinfo, parte del paquete "poppler-utils"[114], para instalarlo usamos:

```
# apt install poppler-utils
```

pdfinfo permite volcar por pantalla la información o metadatos de un archivo PDF pasado como parámetro:

```
$ pdfinfo archivo.pdf
```

Podemos combinar archivos en uno solo, para realizar esta acción, los archivos que se van a combinar deben estar en el mismo directorio donde se ejecuta pdfunife. Usando los archivos que he nombrado anteriormente, el comando a utilizar sería el siguiente:

```
$ pdfunite archivo1.pdf archivo2.pdf archivoCombinado
```

Podemos convertir PDFs a Portable Pixmap (.ppm) u otros formatos (PNG, JPEG, JPEGCMYK, JPEGOPT, TIFF) usando el comando *pdftoppm*, por ejemplo el archivo.pdf a imagen-1.png, imagen-2.png, etc., mediante:

```
$ pdftoppm archivo.pdf imagenes -png
```

o podemos indicar las páginas (first y last):

```
$ pdftoppm -f 10 -l 15 archivo.pdf imagenes -png
```

y en caso necesario podemos indicar la resolución:

```
$ pdftoppm -rx 300 -ry 300 archivo.pdf imagenes -png
```

para convertir en escala de gris usamos:

```
$ pdftoppm -gray archivo.pdf imagenes
```

para convertir en monocromo usamos:

[114] Otros comandos del paquete son: pdfattach, pdfdetach, pdffonts, pdfimages, pdfinfo, pdfseparate, pdfsig, pdftocairo, pdftohtml, pdftoppm, pdftops, pdftotext, pdfunite.

```
$ pdftoppm -mono archivo.pdf imagenes
```

También podemos transformar archivos pdf en archivos de texto (txt).

```
$ pdftotext -layout archivo.pdf archivo.txt
```

la opción -layout intenta mantener (en la medida de lo posible) el formato original del texto.

Podemos usar el comando pdftohtml para conventir un *PDF* en formato html (genera su salida en el directorio de trabajo actual), mediante:

```
$ pdftohtml archivo.pdf archivo.html
```

si queremos que el resultado sea más fiel al original podemos usar la opción -c. En tal caso cada página aparecerá como un archivo separado. Si el archivo original tiene muchas páginas, puede ser conveniente crear una carpeta para almacenar el resultado.

```
$ pdftohtml -c archivo.pdf carpeta_creada/archivo.html
```

Podemos convertir archivos *PDF* a archivos de texto simple. Básicamente lo que hace es extraer los datos de texto de los archivos *PDF*. En ella vamos a encontrar muchas opciones disponibles, incluida la capacidad de especificar el rango de páginas para convertir, la posibilidad de mantener el diseño físico original del texto lo mejor posible, establecer finales de línea e incluso trabajar con archivos *PDF* protegidos con una contraseña.

```
$ pdftotext -layout pdf-entrada.pdf pdf-salida.txt
```

si no nos interesa convertir todo el archivo *PDF*, y queremos acotar un rango de páginas del PDF a convertir en texto habrá que utilizar la opción -f (primera página para convertir) y -l (última página para convertir) seguida cada una de las opciones con el número de la página. El comando a utilizar sería algo como lo siguiente:

```
$ pdftotext -layout -f P -l U pdf-entrada.pdf
```

Por otro lado, podemos extraer todas las imágenes de una archivo PDF, y guardarlas como archivos de tipo Portable Pixmap (PPM) Portable Bitmap (PBM) o archivos JPEG. La sintaxis de esta herramienta es:

```
$ pdfimages archivo.pdf imagen
```

donde archivo.pdf es el fichero del que quieres extraer las imágenes e imagen será el nombre que tendrá la imagen que se extraiga, en caso de ser varias, se irán nombrando de la forma: imagen-000.(extensión), imagen-001.(extensión), etc. La extensión por defecto será .ppm, si se trata de imágenes en color, o pbm si son en grises. Si queremos que en lugar de estos formatos queremos que lo guarde en jpg, tendremos que utilizar la opción "-j", de esta manera, las imágenes que estén en formato DCT, las extraerá en formato jpeg, y el resto en los formatos anteriores según sea en escala de grises o no:

```
$ pdfimages -j archivo.pdf imagen
```

Por otro lado, si no queremos extraer imágenes de todo el documento, sino sólo de las páginas de la 8 a la 15, por ejemplo, tendremos que utilizar el siguiente comando:

```
$ pdfimages -f pagina-inicial -l pagina-final archivo.pdf imagen
```

También, puede suceder que el documento esté protegido, o bien con algunas restricciones, para lo que tendremos que proporcionar la contraseña de propietario:

```
$ pdfimages -opw contraseña_de_propietario archivo.pdf im-
agen
```

O bien, si el documento está protegido con una contraseña de usuario

```
$ pdfimages -upw contraseña_de_usuario archivo.pdf imagen
```

Otra medida de urgencia podría ser la de realizar una conversión intermedia a otro formato y posteriormente volver a regenerar el documento PDF. Para ello, podríamos utilizar los comandos pdf2ps y ps2pdf utilizando el formato intermedio PostScript:

```
$ pdf2ps original.pdf intermedio.ps
$ ps2pdf intermedio.ps optimizado.pdf
```

De la misma forma, también podríamos hacer el mismo proceso utilizando el formato intermedio DJVU y los comandos pdf2djvu y djvu2pdf:

```
$ pdf2djvu original.pdf intermedio.djvu
$ djvu2pdf intermedio.djvu optimizado.pdf
```

Eso sí, ten en cuenta que el formato DJVU está pensado para imágenes escaneadas, por lo que no es apto para todo tipo de documentos PDF. En ambos casos es necesario el intérprete GHostScript.

didjvu si se tiene un archivo DJVU y se necesita convertir a PDF una de las mejores opciones es didjvu, se instala usando:

```
# apt install didjvu
```

este puede hacer la conversión usando:

```
$ didjvu -format=pdf -quality=100 -verbose archivo.djvu archivo.pdf
```

además permite hacer separación de contenido de archivos DJVU de primer y segundo plano.

pdf2svg transforma los archivos pdf en gráficos vectoriales (svg), para instalarlo hacer:

```
# apt install pdf2svg
```

y para usarlo:

```
$ pdf2svg archivo.pdf archivo.svg
```

Convierte una página PDF en un archivo SVG. Si tenemos un documento PDF con varias páginas, y queremos transformarlas todas, escribiremos algo como esto:

```
$ pdf2svg archivo.pdf archivo_página%d.svg all
```

Con all le decimos que las transforme todas. Con %d hacemos que las numere.

ImageMagic: convert si existe una herramienta mágica con la que nunca dejas de sorprenderte es con ImageMagick. Aunque está destinada para tareas con formatos gráficos, es posible utilizarla para reducir el tamaño de un archivo PDF.

Para instalarlo usar:

```
# apt install imagemagick
```

ImageMagick es capaz de trabajar junto a GhostScript para reducir el tamaño de las imágenes que contiene. Para ello, utilizaremos la herramienta convert con los parámetros -compress y -quality.

```
$ convert original.pdf -compress jpeg speaker_jpeg.pdf
```

Con el comando convert -list compress puedes obtener una lista de las opciones de compresión que tienes a tu disposición (BZip, JPEG, LZW, Zip, Lossless...) para comprimir las imágenes del documento y mediante el parámetro -quality establecer la calidad de las imágenes (100 mayor calidad).

Podemos transfórmarlas en imágenes usando:

```
$ convert archivo.pdf archivo.png
```

Podemos convertir imágenes de un formato a otro[115], usando:

```
$ convert archivo.{svg,png}
```

o reducir una imagen a un determinado porcentaje:

```
$ convert -resize 50% original.jpg modificada.jpg
```

También podemos transformar múltiples imágenes en un archivo *pdf* mediante:

```
$ convert imagen{1...8}.png archivo.pdf
```

es probable que se deban cambiar las políticas de ImageMagick para hacer que funcione.

Es posible generar una imágen con el texto indicado, usando:

[115] Soporta los formatos: SVG, BMP, TIFF, PNG, JPG, GIF, WEBP, etc.

```
    $ convert -size 1000x600 -define gradient:radii=1000,500 \
    radial-gradient:#884b88-#010101 -font Impact -pointsize 72
\
    -fill white -gravity center -interline-spacing 50 -annotate 0,0 \
    "¡Herramientas computacionales en Linux!" Archivo.png
```

y es posible crear una paleta de colores animada con nombres usando:

```
    colors=( $( convert -list color | awk '/srgb/{print "xc:"$1}' )
) ; \
    montage -pointsize 12 -fill black -label "%f" -geometry \
    100x100+1+1 ${colors[@]} -tile 5x5 colors.gif ; convert \
    -delay 80 colors.gif colors.gif ; animate colors.gif
```

pdfgrep es una utilidad de línea de comandos para buscar texto en archivos PDF de forma simple y funcional ahorrándonos tiempo de acceder a cada archivo y buscar el texto con herramientas propias de PDF. Para instalar usamos:

```
# apt install pdfgrep
```

Algunas de sus características son:

- Compatible con Grep, podremos ejecutar muchos parámetros de grep como -r, -i, -n o -c.
- Capacidad de buscar texto en múltiples archivos PDF
- Colores destacados, esta opción de color de GNU Grep es compatible y está habilitada por defecto.
- Admite el uso de expresiones regulares.

Iniciaremos con una búsqueda simple, por ejemplo, buscaremos la palabra Solvetic en el archivo Solvetic.pdf, para ello ejecutamos lo siguiente:

```
$ pdfgrep Solvetic Solvetic.pdf
```

Las opciones generales que nos ofrece pdfgrep son:

-i, –ignore-case, Ignora las distinciones de los casos tanto en el origen como en los archivos de entrada.

-F, –fixed-strings, Interpreta PATTERN como una lista de cadenas fijas separadas por líneas nuevas.

–Cache, Usa una Caché para el texto renderizado con el fin de acelerar la operación en archivos de gran tamaño.

-P, –perl-regexp, Interpreta PATTERN como una expresión regular compatible con Perl (PCRE).

-H, –with-filename, Imprime el nombre del archivo para cada coincidencia.

-h, –no-nombre de archivo, Suprime el prefijo del nombre de archivo en la salida.

-n, –page-number, Prefija cada coincidencia con el número de la página donde se encontró el término buscado.

-c, –count, Suprime la salida normal y, en su lugar, imprime el número de coincidencias para cada archivo de entrada.

-p, –Conteo de páginas, Imprime el número de coincidencias por página. Implica -n.

–color, Permite resaltar nombres de archivos, números de página y texto coincidente con diferentes secuencias para mostrarlos en color en la terminal, algunas de sus opciones son Siempre, nunca o automático.

-o, –only-matching, Imprime solo la parte coincidente de una línea sin ningún contexto circundante.

-r, –recursive, Nos permite buscar de forma recursiva todos los archivos (restringidos por –include y –exclude) debajo de cada directorio, siguiendo los enlaces simbólicos solo si están en la línea de comando.

-R, –de referencia-recursiva, Igual que -r, pero sigue todos los enlaces simbólicos.

lowriter uso de la CLI de LibreOffice 'Lowriter' para la conversión de PDF, para instalarlo usamos:

```
# apt instal libreoffice
```

Para realizar la conversión, no tendremos más que seguir la siguiente sintaxis y usar el comando para convertir un solo archivo .doc, ubicado en nuestro directorio actual de trabajo:

```
$ lowriter –convert-to pdf Ejemplo1.doc
```

Si lo que quieres es convertir un archivo .docx, la orden a utilizar es prácticamente la misma:

```
$ lowriter –convert-to pdf Ejemplo2.docx
```

Como puede ver en las anteriores capturas de pantalla, cuando listé el contenido de mi carpeta actual a través del comando ls, también se puede ver los archivos pdf recién creados.

Si nos interesa convertir un grupo de archivos a .pdf no tendremos más que utilizar la siguiente sintaxis. Esta nos servirá para convertir por lotes todos los archivos .doc o .docx a pdf ubicados en nuestro directorio actual:

```
$ lowriter –convert-to pdf *.doc
```

Si los archivos a convertir son .docx, el comando a utilizar será el siguiente:

```
$ lowriter –convert-to pdf *.docx
```

ghostscript para realizar la conversión-optimización necesitamos (en el caso de que no se tenga instalada):

```
apt install ghostscript
```

y que el archivo que queremos optimizar tenga de nombre original.pdf (se puede cambiar pero también necesitas cambiarlo en el comando siguiente), metemos la siguiente línea en nuestra terminal de comandos:

```
gs -sDEVICE=pdfwrite -dCompatibilityLevel=1.4 -dNOPAUSE
-dQUIET -dBATCH -sOutputFile=optimizado.pdf original.pdf
```

Una vez terminado el proceso se genera un archivo llamado optimizado.pdf que tendrá un peso inferior al original:

Pero y si ¿aún sigue siendo muy grande? tenemos otro comando que aún lo reduce más:

```
gs -sDEVICE=pdfwrite -dCompatibilityLevel=1.4 -dPDFSETTINGS=/screen
-dNOPAUSE -dQUIET -dBATCH -sOutputFile=optimizado.pdf
original.pdf
```

Otras opciones de dPDFSETTINGS son:

- /screen Baja calidad (72 dpi) menor tamaño
- /ebook Mediana calidad (150 dpi) moderado tamaño
- /printer Alta calidad (300 dpi) gran tamaño
- /prepress (omisión) Alta calidad (300 dpi) preservando el color
- /default Casi idéntico a screen, pero con calidad ligeramente superior

Con estos sencillos pasos conseguirás bajar el peso del *PDF*, está claro donde más comprimé son las imágenes así que hay que revisarlas antes, para verificar que la calidad es la adecuada.

qpdf es una herramienta de la línea de comandos que permite trabajar con archivos *pdf*. Esta herramienta permite crear archivos optimizados para la *Web*, así como cifrar y descifrar archivos. También te permite convertir archivos con objetos comprimidos en archivos sin objetos comprimidos, así como soporta un modo que te permite editar el contenido del archivo *pdf* en un editor de texto.

Para instalarla usamos:

```
# apt install qpdf
```

Si necesitamos separar las páginas de un pdf (que generará los archivos salida-01.pdf, salida-02.pdf, etc.), usamos:

```
$ qpdf –split-pages original.pdf salida.pdf
```

Si queremos extraer las primeras diez hojas de un documento, solo tenemos que ejecutar el siguiente comando:

```
$ qpdf –empty –pages ejemplo.pdf 1-10 – salida.pdf
```

donde, ejemplo.pdf, es el archivo de entrada, 1-10, las páginas que copiamos del fichero de entrada, salida.pdf, el fichero de salida.

También podemos reordenar las páginas de un documento en orden inverso:

```
$ qpdf –empty –pages ejemplo.pdf z-1 – salida.pdf
```

donde, z, representa la última página.

Podemos unir las páginas de un documento a las del otro:

```
$ qpdf –empty –pages documento1.pdf 1-z documento2.pdf
1-z – salida.pdf
```

También nos permite seleccionar páginas sueltas o rangos tanto de un documento como de varios:

```
$ qpdf –empty –pages documento1.pdf 1,5-7,10-z documento2.pdf
z-8,5,4,2 – salida.pdf
```

Además podemos rotar las páginas, por ejemplo 180 grados usando:

```
$ qpdf archivo.pdf salida.pdf –rotate=+180
```

Otra interesante opción que ofrece *qpdf* es la posibilidad de cifrar y descifrar documentos *pdf*. De cualquier forma, para cifrar un archivo pdf tendrás que utilizar la etiqueta -flag, de forma que la sintaxis es algo como:

```
$ qpdf entrada.pdf –encrypt passwd_usr passwd_own longi-
tud_passwd [restricciones] – salida.pdf
```

o

$ qpdf –linearize –encrypt "" "MiContraseña" 128 –print=full –modify=none –extract=n –use-aes=y – Guia_Limpia.pdf Guia_Protegida.pdf

La longitud de la contraseña, puede tomar uno de los siguientes valores, 40, 128 ó 256. Si no se indican restricciones es permisivo por defecto. Si la longitud de la contraseña es 40, se pueden utilizar las siguientes restricciones:

-print=[yn], indica si se puede imprimir

-modify=[yn], indica si se puede modificar

-extract=[yn], indica si se puede extraer texto o imágenes

-annotate=[yn], indica si se pueden añadir comentarios, cumplimentar un formulario y firmar

Si la longitud es de 128, las restricciones podrán ser las siguientes:

-accessibility=[yn], indica si se es accesible

-extract=[yn], indica si se puede extraer texto o imágenes

-print=print-opt, controla el acceso a la impresión, pudiendo ser alguna de las siguientes opciones:

full, permite la impresión
low, permite la impresión solo a baja resolución
none, no te permite imprimir

-modify=modify-opt, determina si se puede modificar el documento, siguiendo criterios similares a los de impresión:

all, permite cualquier modificación
annotate, permite anotaciones y cumplimentar un formulario
form, permite rellenar un formulario y firmar
none, no permite ninguna modificación

Si tenemos un pdf cifrado y queremos conocer el tipo de cifrado usamos:

$ qpdf –password='passwd' –show-encryption test.pdf

y para quitar la clave usamos:

$ qpdf –password='passwd' –decrypt test.pdf salida.pdf

pdfcrack esta aplicación permite obtener la clave de un pdf usando fuerza bruta, se puede instalar con:

```
# apt install pdfcrack
```

Una vez instalado, ejecutaremos la herramienta[116] con el comando:

```
$ pdfcrack -f nombre_del_archivo.pdf
```

El proceso puede ser muy largo, y más teniendo en cuenta que pdfcrack sólo usa un procesador. Eso sí, puede usar el 100% del mismo. Para acelerar el proceso, podemos añadirle caracteres para probar, para lo que usaremos la opción -c[117]. Esto le dará un punto de partida y puede ser útil si siempre usamos un patrón. Por ejemplo, el siguiente comando sería si yo usara contraseñas con la palabra "coche" y números:

```
$ pdfcrack -f nombre_del_archivo.pdf -c coche1234
```

Si, por la razón que sea, queremos parar el proceso podemos hacerlo con Ctrl+c. En el momento lo pulsemos, *pdfcrack* intentará salvar el estado del proceso, lo que significa que podremos seguirlo más adelante. El archivo del proceso suele guardarse con el nombre "savedstat.sav" en nuestra carpeta personal y para seguirlo usaremos la opción -1. El comando quedaría así:

```
$ pdfcrack -f nombre_del_archivo.pdf -1 savedstate.sav
```

Otra opción que podemos configurar es el número mínimo (-n=valor) y máximo (-m=valor) de caracteres:

```
$ pdfcrack -f nombre_del_archivo.pdf -m=20 -n=12 -c 100690
```

Con el comando anterior le habremos dicho que:

[116] Por omisión, usara estos caracteres para encontrar el password:
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
Podemos tener guardado en un archivo los caracteres a usar y usar -w para indicar que estos se lean de dicho archivo.

[117] Podemos usar toda la colección de caracteres posibles, usando algo como:

```
-c 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789.-
_@#~%$&'
```

La contraseña tiene un mínimo de 12 caracteres.

El máximo que debe combinar son 20 caracteres.

La contraseña tiene en algún punto los caracteres "100690" (no es necesario que estén ordenados).

Otras opciones disponibles son:

-w: para abrir un archivo de texto en donde le configuremos varias palabras. Esto es lo que se conoce como un diccionario. Muchas herramientas que usan la fuerza bruta para descifrar contraseñas cuentan con un diccionario u opción para añadírselo.

-o: para que trabaje con la contraseña de un dueño.

-p: proporciona la contraseña de un usuario para facilitar obtener la contraseña de un dueño.

-s: la permutación está limitada a cambiar la primera palabra por mayúsculas.

-b: nos mostrará unos benchmarks para ver el rendimiento de pdfcrack durante el proceso.

ps2pdf este paquete es parte de ghostscript, para instalarlo usar:

```
apt install ghostscript
```

permite convertir archivos *.ps* a *.pdf*, mediante:

```
$ ps2pdf archivo.ps
```

y lo podemos usar para generar *pdf* desde la línea de comandos, ejemplo:

```
$ man -t top | ps2pdf - top.pdf
```

pdf2ps este paquete es parte de ghostscript, para instalarlo usar:

```
apt install ghostscript
```

permite convertir archivos *.pdf* a *.ps*, mediante:

```
$ pdf2ps archivo.pdf
```

Podemos usar este comando para bajar la calidad de un *pdf* mediante:

```
$ pdf2ps -dLanguageLevel=3 archivo.pdf
$ ps2pdf -dPDFSETTINGS=/ebook archivo.ps
```

htmldoc permite convertir documentos HTML a PDF y PS, para instalarlo usar:

```
# apt install htmldoc
```

y convertimos mediante:

```
$ htmldoc –webpage -f nombre.pdf nombre.html
```

wkhtmltopdf permite convertir páginas Web a PDF, para instalarlo usar:

```
# apt install wkhtmltopdf
```

y convertimos mediante:

```
$ wkhtmltopdf www.debian.org debian.pdf
```

podemos pedir que lo genere en tonos de gris mediante:

```
$ wkhtmltopdf -g www.debian.org debian.pdf
```

o que excluya las imágenes mediante:

```
$ wkhtmltopdf –no-images www.debian.org debian.pdf
```

y si lo necesitamos podemos convertir la página Web a una imagen mediante:

```
$ wkhtmltopdf www.debian.org debian.png
```

enscript convertir archivos ASCII a *.ps* o *.pdf*, para instalarlo usar:

```
# apt install enscript ghostscript
```

Supongamos que necesitamos convertir el archivo salida.txt a formato PDF. Pero además deseamos que la fuente sea "Courier" en tamaño "7", los márgenes sean de apenas 1 punto, y el formato de página o tamaño de papel sea "A4".

Primero se debe convertir el archivo .txt a PS utilizando enscript con las siguientes opciones:

```
$ enscript -M A4 –margins=1:1:1:1 -f Courier7 salida.txt -p
salida.ps
```

luego simplemente convertir el archivo PS generado por enscript a PDF utilizando ps2pdf:

```
$ ps2pdf salida.ps
```

Las opciones más comunes de enscript son:

- -a: rango de páginas a imprimir, por ejemplo: "1-10".
- -b: encabezado de página en formato de texto, por ejemplo "$n %W Página $% de $=’".
- -f: fuente y tamaño a utilizar, por ejemplo "Times-Roman12".
- -M: tamaño de papel, por ejemplo: "A4"
- -p: nombre del archivo de salida o "-" para volcar por salida estándar.
- -r: modo "landscape".
- –margins: márgenes (en puntos).
- –footer: pie de página (funciona de manera similar a -b).

Y decenas de opciones más, incluso permite generar un índice o tabla de contenidos (–toc).

pandoc es posible convertir los archivos MD (MarkDown) comunes en Github o GitLab a PDF[118], es necesario instalar el paquete mediante:

```
# apt install pandoc
```

y lo usamos mediante:

```
$ pandoc archivo.md –pdf-engine=xelatex -o archivo.pdf
```

o

```
$ pandoc archivo.md -o archivo.pdf
```

Podemos convertir un archivo MarkDown a .docx, usando:

```
$ pandoc archivo.md -o archivo.docx
```

o convertir un archivo .docx a MarkDown, usando:

```
$ pandoc archivo.docx -o archivo.md
```

Podemos convertir un archivo MarkDown a .html, usando:

```
$ pandoc archivo.md -o archivo.html
```

o convertir un archivo .html a MarkDown, usando:

```
$ pandoc archivo.html -o archivo.md
```

esta herramienta soporta una amplia variedad de conversiones, para más detalle ver su página Web.

[118] Se necesita tener instalado LaTeX, podemos instalarlo usando:

```
# apt install texlive
```

o

```
# apt install texlive-full
```

Otras Herramientas para Trabajar con PDF existen múltiples herramientas para trabajar archivos PDF, a continuación describimos algunas:

- diffpdf.- Muestra las diferencias de dos versiones de un mismo documento
- pdfarranger.- Permite juntar, separar y reordenar páginas de uno o más PDF
- pdfchain.- Interfase gráfica para PDF Tool Kit
- pdfposter.- Permite escalar y colocar en mosaico imágenes/páginas para imprimir en formato largo
- pdf-presenter-console.- Herramienta de presentación multimonitor
- pdfcrack.- Permite encontrar la clave de protección de un PDF mediante fuerza bruta

6.3 Archivos de Imágenes y Vídeos

En GNU/Linux contamos con múltiples paquetes para manipular archivos de imágenes, la gran mayoría de ellos trabajan en ambiente gráfico, pero también hay los que trabajan en línea de comandos que permiten procesar una o múltiples imágenes desde la línea de comandos.

Podemos usar el comando *file* para que nos muestre la información básica (tipo de imagen, resolución, etc.) de cualquier archivo de imagen, usando:

```
$ file archivoImagen
```

¿Y en caso de borrar accidentalmente un archivo de imagen o vídeo ... lo puedo recuperar?, la respuesta es afirmativa para algunos casos, para ello, instalamos:

```
$ apt install recoverjpeg
```

esta herramienta sólo funciona con archivos JPEG, pero en el paquete se incluye otra herramienta llamada *recovermov*, que se utiliza para recuperar archivos .MOV y trabaja en una gran variedad de sistema de archivos. Para usarla hacemos:

recoverjpeg [ruta]

la ruta puede ser algo como: */dev/sda3*, los archivos recuperados serán dejados en la carpeta donde se ejecuto el comando. Una vez recuperados los archivos, puede haber archivos duplicados, los podemos eliminar usando:

```
$ remove-duplicates
```

Al tomar fotografías o vídeos es recomendable por seguridad desactivar el guardado de datos *Exif* (Exchangeable Image File) también conocidos como metadatos, ya que estos contienen información sobre la cámara, sobre la fotografía y sobre su origen como ubicación por GPS, etc. En GNU/Linux podemos instalar el paquete *ExifTool* que permite conocer y borrar los datos *Exif* en fotografías. Para instalarlo usamos:

```
# apt install libimage-exiftool-perl
```

Para visualizar los datos Exif, usamos:

```
$ exiftool imagen.gif
```

Para borrar todos los datos Exif, usamos:

```
$ exiftool -all=imagen.gif
```

Pngcheck verifica la integridad de los archivos PNG, JNG y MNG (verificando los CRC internos de 32 bits, también conocidos como sumas de comprobación, y descomprimiendo los datos de la imagen), opcionalmente puede volcar casi toda la información a nivel de fragmentos en la imagen legible por humanos. Por ejemplo, se puede utilizar para imprimir las estadísticas básicas sobre una imagen (dimensión, profundidad de bits, etc.); para enumerar la información de color y transparencia en su paleta (suponiendo que renga una), o para extraer las anotaciones de texto incrustadas. Para usarlo, lo instalamos mediante:

```
# apt install pngcheck
```

Para conocer revisar la integridad y datos de una imagen, usamos:

```
$ pngcheck -cvt imagen.png
```

si queremos trabajar con múltiples imágenes simultáneamente, usamos:

```
$ pngcheck -c *.png
```

también podemos usar el programa *parallel* para hacer las revisiones usando todos los procesadores del equipo:

```
$ ls *.png | parallel –nice 19 –bar –will-cite "pngcheck -q {}"
```

Imagemagick algunos consideran a este paquete como la "navaja suiza" para manipular imágenes desde la línea de comandos. Para instalarlo usamos:

```
# apt install imagemagick
```

Entre los múltiples comando que contiene este paquete iniciamos con *conver*, entre sus múltiples opciones es la de reducir el tamaño de las imágenes, la calidad o transformar entre diferentes formatos. La forma más sencilla de usarlo para pasar un archivo de PNG a JPG sin cambiar tamaño ni calidad es:

```
$ convert antiguo.png nuevo.jpg
```

o, usando:

```
$ convert archivo.{svg,png}
```

para juntar dos imágenes en donde ponemos una junto a la otra, usamos:

```
$ convert +append a.png b.png Resultado.png
```

para juntar dos imágenes en donde ponemos una arriba y la otra abajo, usamos:

```
$ convert -append a.png b.png Resultado.png
```

y si queremos reducir el tamaño (alto y ancho) a la mitad, respetando el aspecto:

```
$ convert -scale 50% antiguo.png nuevo.jpg
```

también podemos reducir el tamaño y también la calidad al 80% (con esto conseguimos un archivo mucho más pequeño):

```
$ convert -scale 50% -quality 80% antiguo.png nuevo.jpg
```

o simplemente podemos pedir que la imagen sea reducida a un determinado porcentaje:

```
$ convert -resize 50% original.jpg modificada.jpg
```

Otra cosa común es querer transformar una imagen al formato PDF, por ejemplo:

```
$ convert foto.jpg archivo.pdf
```

y si son múltiples archivos usamos:

```
$ convert *.jpg archivo.pdf
```

también podemos rotar la imagen (digamos 90 grados) y añadir compresión, usando:

```
$ convert -rotate 90 foto.jpg + compress archivo.pdf
```

Podemos convertir múltiples imágenes al mismo tiempo usado:

```
$ find .  -name "*.jpg" -exec convert \{} -verbose -resize
450x250\> \{} \;
```

Otro comando es *mogrify* que permite cambiar el tamaño de una imagen, desenfocar, recortar, eliminar, difuminar, dibujar, voltear, unir, volver a muestrear y mucho más. Por omisión el comando sobreescribe el archivo de imagen original, mientras convert no. Por ejemplo para modificar el tamaño de la foto al 25%, usamos:

```
$ mogrify -resize 25% archivo.jpeg
```

o podemos indicar el tamaño, por ejemplo 100x67, usando:

```
$ mogrify -resize 100x67 archivo.jpeg
```

podemos cambiar el formato de jpeg a png, usando:

```
$ mogrify -format png archivo.jpeg
```

podemos trabajar con múltiples archivos, por ejemplo cambiando todos los archivos *.jpeg a png y reduciendo su tamaño al 50%, usando:

```
$ mogrify -resize 50% -format png *.jpeg
```

para rotar una imagen digamos 180 grados, usamos:

```
$ mogrify -rotate "+180" archivo.jpeg
```

Otro comando más es montage, su uso original es generar tablas de miniaturas de imágenes, es decir, hacer referencia con miniaturas a grandes colecciones de imágenes, especialmente fotos. Y aunque se puede usar para este propósito, también permite hacer mucho más, por ejemplo para hacer Collage de fotos.

Para este ejemplo, suponemos que tenemos cuatro imágenes (sin importar el tipo), si lo que buscamos es crear un montaje básico a partir de estas imágenes, en la terminal sólo tendremos que ejecutar:

```
$ montage img1.png img2.png img3.png img4.png salida.png
```

Si todas las imágenes son del mismo tipo, también podemos utilizar el siguiente comando para realizar el montaje con todas las imágenes situadas en el mismo directorio:

```
$ montage *.png salida.png
```

También podemos establecer el tamaño y el espacio entre las imágenes, para ello, la herramienta que nos ocupa cuenta con una opción llamada *-geometry*. Esto nos va a ser de ayuda a la hora de establecer el tamaño de la miniatura y el espacio entre cada imagen. La configuración predeterminada para esto es de 120x120>+4+3. Si en un montaje nos interesa establecer un espacio de 2 píxeles entre las imágenes, el comando a ejecutar sería :

```
$ montage -geometry +2+2 *.png salida.png
```

Esto es útil sólo cuando buscamos crear una imagen compuesta a partir de imágenes del mismo tamaño. Cosa que no ocurre con frecuencia, en caso de que nuestras imágenes tengan diferentes tamaños, es posible cambiar el tamaño de todas ellas al mismo tiempo:

```
$ montage -geometry 90x90+2+2 *.png salida.png
```

este comando va a reducir las imágenes dadas para que quepan en un cuadro de 90x90 píxeles de tamaño.

Para crear un montaje con efecto Polaroid con nuestras imágenes solo tendremos que ejecutar:

```
$ montage +polaroid *.png salida.png
```

también podremos dar un efecto Polaroid y hacer que las imágenes se superpongan, utilizando el comando:

```
$ montage -geometry 100x100-10-2 +polaroid *.png salida.png
```

Otra opción disponible será la de *-set label*. Con ella podemos indicarle a la herramienta *montage* que establezca etiquetas para cada imagen en miniatura. Este comando etiqueta las imágenes en miniatura con sus nombres de origen:

```
$ montage -set label '%f' *.png salida.png
```

si te interesa poder establecer una etiqueta personalizada para cada imagen, el comando a utilizar sería algo como:

```
$ montage -label Ejemplo1 img1.png -label Ejemplo2 img2.png
-label Ejemplo3 img3.png -label Ejemplo4 img4.png salida.png
```

además, también se puede establecer un título al montaje que acabamos de realizar. Solo tendremos que añadir la opción -title de la siguiente forma:

```
$ montage -label Ejemplo1 img1.png -label Ejemplo2 img2.png
-label Ejemplo3 img3.png -label Ejemplo4 img4.png -title 'Ejem-
plo para este texto' salida.png
```

Otra característica interesante de la herramienta de *montage* es la posibilidad de concatenar imágenes sin espacios entre ellas, para ello hacemos:

```
$ montage -mode Concatenate *.png salida.png
```

Hay muchas otras opciones de este paquete, solo nos quedamos con lo más básico que ofrece esta herramienta, se pueden consultar todas las opciones disponibles en las páginas de proyecto.

WebP es un formato de imagen de código abierto creado por Google hace ya varios años, para mejorar la gestión de imágenes al hacerlas más ligeras conservando una mejor calidad, con la finalidad de que sean más rápidas su carga y visualización sobre los sitios Web. Es un formato de imagen moderno que proporciona una compresión superior sin pérdidas y con pérdidas para las imágenes en la Web.

Las imágenes sin pérdida de WebP son 26% más pequeñas en tamaño comparadas con las PNG. Las imágenes con pérdida de WebP son 25-34% más pequeñas que las imágenes JPEG comparables con un índice de calidad SSIM equivalente. WebP sin pérdidas soporta transparencia (también conocido como canal alfa) a un costo de sólo 22% de Bytes adicionales. Para los casos en que la compresión RGB con pérdida es aceptable, la WebP con pérdida también soporta la transparencia, típicamente proporcionando tamaños de archivo tres veces más pequeños comparados con PNG. Para instalar el paquete WebP usamos:

```
# apt install webp
```

Y para usarlo escribimos:

```
$ cwebp -q 60 imagen.png -o imagen.webp
```

donde el modificador -q define la calidad de salida y -o especifica el archivo de salida. Otras herramientas del paquete WebP son;

- anim_diff - herramienta para mostrar la diferencia entre imágenes de animación
- anim_dump - herramienta para volcar la diferencia entre imágenes de animación.

- cwebp - herramienta de codificador Webp.
- dwebp - herramienta decodificadora Webp.
- gif2webp - herramienta para convertir imágenes GIF a Webp.
- img2webp - herramientas para convertir una secuencia de imágenes en un archivo Webp animado.
- vwebp - visor de archivos Webp.
- webpinfo - se usa para ver información sobre un archivo de imagen Webp.
- webpmux - herramienta de muxing Webp.

Si tenemos varios archivos y queremos aplicar la misma transformación a todos a la vez, podemos crearnos un bucle. En el siguiente ejemplo vemos cómo ir tomando uno por uno cada archivo PNG y pasándolo a JPG:

```
for img in *.png; do
    filename=${img%.*}
    convert "$filename.png" "$filename.jpg"
done
```

La siguiente orden hace algo similar pero al mismo tiempo va reduciendo la calidad de la imagen de salida:

```
$ find . -name "*png" | xargs -l -i basename -s ".png" "{}" |
xargs -l -i convert -quality 85% "{}.png" "{}.jpg"
```

Por último, dos comandos bastante útiles. El primero nos da información detallada del formato, dimensiones, paleta de color, etc de un archivo de imagen cualquiera:

```
identify imagen.jpg
```

Otras formas de usarlo, por ejemplo convirtiendo de PNG a JPG::

```
$ ls -1 *.png | xargs -n 1 bash -c 'convert "$0" "${0%.png}.jpg"'
```

y de JPG a PNG:

```
$ ls -1 *.jpg | xargs -n 1 bash -c 'convert "$0" "${0%.jpg}.png"'
```

Por otro lado, podemos poner textos a una imagen (para ser usada en un meme), mediante:

```
$ convert imagen.png -font impact -fill white -pointsize 84
-stroke black -strokewidth 3
-gravity north -annotate +0+20 'Texto superior'
-gravity south -annotate +0+20 'Texto inferior' resultado.png
```

Si necesitamos tomar capturas de pantalla, podemos usar el paquete *imagemagick*, mediante el comando *import*, por ejemplo para tomar captura de toda la pantalla, usamos:

```
$ import -window root imagen.png
```

y si solo queremos solo una parte de la pantalla, usamos:

```
$ import imagen.png
```

Otra opción es el programa es *scrot*, para instalarlo usamos:
apt install scrot
Si necesitamos tomar captura de toda la pantalla, usamos:

```
$ scrot imagen.png
```

y si solo queremos solo una parte de la pantalla, usamos:

```
$ scrot -s imagen.png
```

También es posible convertir una imagen a formato *ASCII*, para ello tenemos que instalar el paquete *jp2a*, mediante:

```
# apt install jp2a
```

y para convertir un archivo.*jpg* a *ASCII*, usamos:

```
$ jp2a archivo.jpg
```

o si lo necesitamos, indicamos los archivos a convertir:

```
$ jp2a archivo1.jpg archivo2.jpg archivo3.jpg
```

podemos indicar el tamaño de la salida, usando:

```
$ jp2a –size=50x30 archivo.jpg
```

o podemos imprimir las imágenes en formato con fondo Light/Dark, mediante:

```
$ jp2a –background=light archivo.jpg
$ jp2a –background=dark archivo,jpg
```

o podemos invertir la imagen, usando:

```
$ jp2a archivo.jpg –invert
```

Dado que *jp2a* solo trabaja con archivos de formato *.jpg*, podemos usar *convert* para poder convertir cualquier archivo de imagen, ejemplo:

```
$ convert archivo.png jpg:- | jp2a -
```

GNU Parallel permite a un usuario construir y ejecutar comandos de Shell desde la entrada estándar en paralelo. Por ejemplo convirtiendo de PNG a JPG:

```
$ parallel convert '{}' '{.}.jpg' ::: *.png
```

y de JPG a PNG:

```
$ parallel convert '{}' '{.}.png' ::: *.jpg
```

o mediante, por ejemplo de de PNG a JPG:

```
$ ls -1 *.png | parallel convert '{}' '{.}.jpg'
```

y de JPG a PNG:

```
$ ls -1 *.jpg | parallel convert '{}' '{.}.png'
```

Reconocimiento Óptico de Caracteres (OCR) es la capacidad de mirar y encontrar palabras en una imagen y luego extraerlas como texto editable. Esta sencilla tarea para los humanos es difícil de realizar para las computadoras. Podemos hacer esto desde la línea de comandos de GNU/Linux, una opción es instalar *tesseract-ocr*, mediante:

```
# apt install tesseract-ocr
```

y para usarlo, tomamos cualquier imagen con texto y escribimos:

```
$ tesseract archivo.png archivoTxt
```

podemos indicar el idioma y la resolución para un mejor reconocimiento, mediante:

```
$ tesseract -l eng –dpi 300 archivo.png archivoTxt
```

en caso de tener un archivo *.pdf*, podemos hacer:

```
$ pdftoppm -png turing.pdf turing
$ tesseract -l eng –dpi 300 turing.png turing
```

FFMPEG es una completa solución multi-plataforma para grabar, convertir y hacer Streaming de audio y vídeo a otras fuentes. Aunque a menudo se confunde a FFMPEG con un codec, en realidad es un Framework, un conjunto de Codecs y herramientas con las que lleva a cabo todas las tareas de grabar, convertir y hacer Streaming.

A grandes rasgos, este Framework multimedia está formado por varios componentes:

- ffmpeg: la herramienta para usar el framework desde línea de comandos.
- ffserver: servidor para hacer streaming multimedia.
- ffplay: reproductor multimedia integrado.
- libavcodec: biblioteca con todos los codecs de audio y vídeo de FFMPEG.

- libavformat: biblioteca que contiene los multiplexadores y demultiplexadores.
- libavutil: biblioteca de apoyo.
- libpostproc: biblioteca que se encarga del postproceso de vídeo.
- libswscale: la biblioteca de escalado de vídeo.

El cual se instala usando:

```
# apt install ffmpeg
```

para conocer los Codecs disponibles usamos:

```
$ ffmpeg -codecs
```

y para listar los formatos disponibles usamos:

```
$ ffmpeg -formats
```

Por ejemplo, muestra los datos del video:

```
$ ffmpeg -i output.mp4
```

cuando ejecuta el comando anterior, ffmpeg muestra los datos del video, pero el encabezado dificulta ver esto, para ignorar los datos iniciales, ejecute:

```
$ ffmpeg -i output.mp4 -hide_banner
```

Para conocer la resolución de un video usamos:

```
$ ffprobe -v error -select_streams v:0 -show_entries stream=width,height
-of csv=s=x:p=0 video.mp4
```

Para cambiar la resolución a digamos 1920:1080 usamos:

```
$ ffmpeg -i videoOriginal.mp4 -vf scale=1980:1080 videoFi-
nal.mp4
```

Para convertir de MP4 a GIF, usamos:

```
$ ffmpeg -i archivo.mp4 archivo.gif
```

y podemos visualizarlo usando:

```
$ ffplay archivo.gif
```

o la aplicación por omisión:

```
$ xdg-open archivo.gif
```

Podemos usarlo para convertir audio usando:

```
$ ffmpeg -i archivoEntrada.wav salida.mp3
```

Podemos convertir un video *.av* a *.mp4* usando:

```
$ ffmpeg -i video.av -code cpy video.mp4
```

Bajar la resolución de un video (por ejemplo mediante H264 y mp3), usando:

```
$ ffmpeg -i entrada.mp4 -vcodev h264 -acodec mp3 salida.mp4
```

Convertir video *.mp4* a *gif* mediante:

```
$ ffmpeg -i Archivo.mp4 archivo.gif
```

Remover el audio de un video mediante:

```
$ ffmpeg -i entrada.mp4 -c copy -an salids.mp4
```

Averiguar cuántos FPS tiene tu video

```
$ ffmpeg -i output.mp4 2>&1 | egrep -o '[0-9]+ fps'
```

Insertar ZOOM en su video

```
$ fmpeg -i input.mp4 -vf "zoompan=z='if(lte(mod(time,10),10),2,1)':d=1:x=iw/2-
  (iw/zoom/2 ):y=ih/2-(ih/zoom/2):fps=30" output.mp4
```

Optimizando tu video (disminuir el tamaño sin perder calidad de imagen):

```
$ ffmpeg -i input.mp4 -vcodec libx264 -crf 28 output.mp4
```

Cambiar el tamaño de la resolución de video (dejándolo en la resolución: 1280x720):

```
$ ffmpeg -i input.mp4 -vf scale=1280:720 -preset slow -crf 18
output.mp4
```

Cambiar el tamaño de la anchura y la altura será proporcional (especificado para un ancho de 1280):

```
$ ffmpeg -i input.mp4 -vf scale=1280:-1 output.mp4
```

Cambiar el tamaño de la altura y el ancho será proporcional (especificado para altura de 720):

```
$ ffmpeg -i input.mp4 -vf scale=-1:720 output.mp4
```

Rotar un video (parece la pantalla de un celular, el ancho se convierte en alto y viceversa):

```
$ ffmpeg -i input.mp4 -vf "transpose=clock" output.mp4
```

Rotando 180° (si el video está "al revés" lo invertirás):

```
$ ffmpeg -i input.mp4 -vf "transpose=2,transpose=2" out-
put.mp4
```

Extraer cuadros de un video:

```
$ ffmpeg -y -ss 00:00 -i input.mp4 -t 10 "frames/filename%05d.jpg"
```

Extraer fotogramas solo de los 10 segundos iniciales

```
$ ffmpeg -y -ss 00:00 -i input.mp4 -t 10 "frames/filename%05d.jpg"
```

por ejemplo extraemos 30 imágenes por segundo desde el primer segundo hasta el segundo 6 -es decir, durante 5 segundos- y las guardará siguiendo el siguiente patrón imagen001.jpg, imagen 002.jpg, etc., ejemplo:

```
$ ffmpeg -i video.mp4 -ss 00:00:01 -t 5 - r 30 -f image2 ima-
gen%03d.jpg
```

y si ahora tenemos los fotogramas y queremos formar un video, podemos hacerlo mediante:

```
$ ffmpeg -framerate 10 -i imagen-%03d.jpg video.mp4
```

Agregar subtítulos al video:

```
$ ffmpeg -i input.mp4 -i subtitle.srt -c copy -c:s mov_text
outfile.mp4
```

ejemplo del archivo: *subtitle.srt*

```
1
00:00:00,000 –> 00:00:02,827
- Terminal Raíz - Sistemas
Operacional, C++ y Desarrollo.
2
00:00:02,827 –> 00:00:06,383
Ejemplos de diferentes usos
de ffmpeg para ayudarte
3
00:00:06,383 –> 00:00:09,427
No olvides leer también
los enlaces de abajo.
```

Ver un video:

```
$ ffplay video.mp4
```

Escuchar música:

```
$ ffplay music.mp3
```

Se tienen múltiples opciones en el paquete, por ejemplo: para controlar la calidad de audio, control del Bitrate, extraer parte de un audio, unir dos o más audios, recortar un audio, agregar etiquetas en los audios, añadir un portal al audio, manipulación del volumen del audio entre otras cosas.

Descargar Vídeos de youtube En la gran mayoría de las distribuciones se tiene acceso al comando youtube-dl para descargar videos de www.youtube.com, en caso de no ser así, lo podemos instalar usando:

```
# apt install youtube-dl
```

para descargar video o Playlist, usamos:

```
$ youtube-dl [URL]
```

en caso de desear cambiar el nombre usamos:

```
$ youtube-dl -o 'nombre' [URL]
```

si se desea descargar múltiples vídeos podemos usar:

```
$ youtube-dl [URL1] [URL2]
$ youtube-dl -a archivoDireciones.txt
```

si deseamos descargar un solo video de una Playlist, usamos:

```
$ youtube-dl –playlist-items [número1, número2, etc] [URL]
```

si necesitamos descargar el video con un formato específico, usamos:

```
$ youtube-dl –format mp4 [URL]
```

si sólo deseamos descargar el audio de un video, usamos:

```
$ youtube-dl -x [URL]
```

por omisión se descarga con formato *Ogg*, pero podemos descargarlo usando *mp3*, mediante:

```
$ youtube-dl -x –audio-format mp3 [URL]
```

también podemos descargar los vídeos con su descripción, metadatos, anotaciones, subtítulos y Thumbnail, usando:

```
$ youtube-dl –write-description –write-info-json –write-annotations
–write-sub –write-thumbnails [URL]
```

Para conocer todos los formatos disponibles de video o Playlist, usamos:

$ youtube-dl –list-formats [URL]

o

$ youtube-dl -F [URL]

para indicar que se descargue con algún formato específico, usamos:

$ youtube-dl f [númeroFormato] [URL]

y para solicitar que se descargue con la mejor resolución posible, usamos:

$ youtube-dl -f best [URL]

otras opciones son: *worst, bestvideo, worstvideo, bestaudio, worstaudio*

Por omisión, el comando *youtube-dl* reanudará automáticamente la descarga donde la dejamos. Si no deseamos reanudar la descarga, podemos forzar la reanudación de los archivos parcialmente descargados, usando:

$ youtube-dl -c [URL]

6.4 Desde la Nube

Existen diferentes servicios Web[119] que permiten editar, compilar y generar el archivo PDF o DVI desde el navegador, esto en aras de que los estudiantes y profesores que cuenten con algún sistema de acceso a red y un navegador puedan trabajar en LaTeX sin hacer instalación alguna en su equipo de cómputo, tableta o teléfono celular.

Algunos ejemplos de estos servicio son:

- https://es.sharelatex.com/

[119] Cuando se trabaja desde la Web es recomendable usar el modo Privado o Incógnito para no guardar el historial de navegación, información introducida en los formularios y borrar al cerrar el navegador los datos de los sitios visitados. Pero recuerda que los sitios Web que visitamos sí guardan información de nuestra visita, nuestro proveedor de Internet también guarda constancia de nuestra visita y si descargamos algo, esto no se borra al igual que el historial de descargas, además de las marcas de páginas o favoritos se conservarán al cerrar el navegador.

- https://papeeria.com/
- https://www.overleaf.com/
- https://www.authorea.com/
- https://latexbase.com/
- https://www.codecogs.com/latex/eqneditor.php

Detexify permite conocer el código de LaTeX de un símbolo o mediante el dibujo de un símbolo nos muestra diferentes opciones y su respectiva codificación en LaTeX, se puede consultar en:
https://detexify.kirelabs.org/classify.html

Tex Match es una aplicación Snapd que puede ser instalada para diversas distribuciones de Linux que permite conocer el código de LaTeX de más de 1000 símbolos o mediante el dibujo de un símbolo nos muestra diferentes opciones y su respectiva codificación en LaTeX, se puede consultar en:
https://snapcraft.io/tex-match

Mathpix es una aplicación (Linux, MacOS, Windows) de escáner para crear documentos digitales que contienen ecuaciones matemáticas con la mínima cantidad de esfuerzo. Digitaliza texto escrito a mano o impreso y copia los resultados al portapapeles que se puede pegar en editores LaTeX, se puede consultar en:
https://mathpix.com

Mathcha es un editor en línea que permite escribir y compartir textos e imágenes de matemáticas, se puede consultar en:
https://www.mathcha.io

7 Entornos de Desarrollo y Herramientas de Programación

En los últimos años los lenguajes de programación han ido evolucionado en el desarrollo de sistemas o Software, con el objetivo principal de facilitar al usuario las actividades que realiza día con día; por tal motivo, como programador, es importante conocer los conceptos básicos de programación, los tipos de lenguajes que se utilizan para el desarrollo y su funcionamiento para la interpretación de algoritmos, así como para dar solución a los problemas que pudieran presentarse.

En términos generales, un lenguaje de programación es una herramienta que permite desarrollar Software o programas para computadora. Los lenguajes de programación son empleados para diseñar e implementar programas encargados de definir y administrar el comportamiento de los dispositivos físicos y lógicos de una computadora. Lo anterior se logra mediante la creación e implementación de algoritmos de precisión que se utilizan como una forma de comunicación humana con la computadora.

Los 5 lenguajes de programación más populares en la actualidad son Python, C++, C, Java y C# según el índice de TIOBE[120] actualizado al 2025, por otro lado en la lista de la IEEE son: Python, Java, JavaScript, C++, TypeScript.

A grandes rasgos, un lenguaje de programación se conforma de una serie de símbolos y reglas de sintaxis y semántica que definen la estructura principal del lenguaje y le dan un significado a sus elementos y expresiones, y donde programación es el proceso de análisis, diseño, implementación, prueba y depuración de un algoritmo, a partir de un lenguaje que puede ser interpretado o se compila y genera un código ejecutable para la computadora.

La función principal de los lenguajes de programación es escribir programas que permiten la comunicación usuario-máquina. Unos programas especiales (compiladores o intérpretes) convierten las instrucciones escritas en código fuente, en instrucciones escritas en lenguaje máquina.

Los intérpretes leen la instrucción línea por línea y obtienen el código

[120] Pero si vemos los cinco lenguajes más populares a septiembre de 2020 según el índice TIOBE. La lista consta de lenguajes tan conocidos como C, Java, Python, C++ y C#. Salvo Python, los otros cuatro ya estaban en el TOP 5 allá por 2015 y por 2010. Es más, si comparamos el índice de 2020 y 2019, el único cambio es entre C y Java, que pasan de ser segundo y primero a primero y segundo. El resto permanece igual.

máquina correspondiente.

En cuanto a los compiladores, traducen los símbolos de un lenguaje de programación a su equivalencia escrito en lenguaje máquina (proceso conocido como compilar). Por último, se obtiene un programa ejecutable.

Para programar, es necesario como mínimo contar con un editor de texto -como *vi*, *nano o micro*- y acceso al compilador o intérprete del lenguaje que nos interese. En Linux se tiene una gran variedad de lenguajes y herramientas de desarrollo -Linux fue hecho por programadores para programadores- que se pueden instalar. Pero, también están los entornos de desarrollo integrado o entorno de desarrollo interactivo -en inglés Integrated Development Environment (IDE)-, estas son aplicaciones informáticas que proporcionan servicios integrales para facilitarle al programador el desarrollo de Software.

Normalmente, un IDE consiste de un editor de código fuente, herramientas de construcción automáticas y un depurador. La mayoría de los IDE tienen auto-completado inteligente de código (IntelliSense). Algunos IDE contienen un compilador, un intérprete, o ambos, tales como NetBeans y Eclipse. El límite entre un IDE y otras partes del entorno de desarrollo de Software más amplio no está bien definido. Muchas veces, a los efectos de simplificar la construcción de la interfaz gráfica de usuario (GUI, por sus siglas en inglés) se integran un sistema controlador de versión y varias herramientas. Muchos IDE modernos también cuentan con un navegador de clases, un buscador de objetos y un diagrama de jerarquía de clases, para su uso con el desarrollo de Software orientado a objetos.

Los IDE están diseñados para maximizar la productividad del programador proporcionando componentes muy unidos con interfaces de usuario similares. Los IDE presentan un único programa en el que se lleva a cabo todo el desarrollo. Generalmente, este programa suele ofrecer muchas características para la creación, modificación, compilación, implementación y depuración de Software. Esto contrasta con el desarrollo de Software utilizando herramientas no relacionadas, como *vi*, GNU Compiler Collection (*gcc*) o *make*.

Uno de los propósitos de los IDE es reducir la configuración necesaria para reconstruir múltiples utilidades de desarrollo, en vez de proveer el mismo conjunto de servicios como una unidad cohesiva. Reduciendo ese tiempo de ajustes, se puede incrementar la productividad de desarrollo, en casos donde aprender a usar un IDE es más rápido que integrar manualmente todas las herramientas por separado.

Una mejor integración de todos los procesos de desarrollo hace posi-

ble mejorar la productividad en general, que únicamente ayudando con los ajustes de configuración. Por ejemplo, el código puede ser continuamente armado, mientras es editado, previendo retroalimentación instantánea, como cuando hay errores de sintaxis. Esto puede ayudar a aprender un nuevo lenguaje de programación de una manera más rápida, así como sus librerías asociadas.

Algunos IDE están dedicados específicamente a un lenguaje de programación, permitiendo que las características sean lo más cercanas al paradigma de programación de dicho lenguaje. Por otro lado, existen muchos IDE de múltiples lenguajes tales como *Eclipse, ActiveState Komodo, IntelliJ IDEA, MyEclipse, Oracle JDeveloper, NetBeans, Codenvy y Microsoft Visual Studio.* Por otro lado *Xcode, Xojo y Delphi* están dedicados a un lenguaje cerrado o a un tipo de lenguajes de programación.

Los IDE ofrecen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como *C++, Python, Java, C#, Delphi, Visual Basic,* etc. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto. Es deseable que un IDE cuente con las siguientes características:

- Multiplataforma
- Soporte para diversos lenguajes de programación
- Integración con Sistemas de Control de Versiones
- Reconocimiento de Sintaxis
- Extensiones y Componentes para el IDE
- Integración con Framework populares
- Depurador
- Importar y Exportar proyectos
- Múltiples idiomas
- Manual de Usuarios y Ayuda

- Componentes
- Editor de texto
- Compilador.
- Intérprete
- Herramientas de automatización
- Depurador
- Posibilidad de ofrecer un sistema de control de versiones
- Factibilidad para ayudar en la construcción de interfaces gráficas de usuarios

Algunos de los más usados son: *Eclipse, Aptana, NetBeans, Sublime Text, Geany, Visual Studio, Brackets, Monodevelop, Komodo, Anjuta, CodeLite, Code::Blocks, PyDev, Eric, PyCharm, PTK, Spyder, Bluefish, Glade, Kdevelop, Emacs, QtCreator, Android SDK, WxFormBuilder,* etc.

¿Por qué los editores de texto son más populares que los IDEs entre los programadores de la industria del Software? Es común que entre programadores experimentados (por ejemplo en Google), el uso de por ejemplo emacs o vi(m). Aquí hay algunas razones que se me ocurren para el uso generalizado de dichos editores:

- Velocidad: emacs y vi son básicos, por lo que nunca se cuelgan. Los bloqueos de IDEs son una especie de broma recurrente entre los programadores que lo usan. Además, emacs y vi admiten potentes atajos de teclado y comandos: una vez que los has usado varias veces, no necesitas una barra de menú para ejecutarlos.
- Hábito: la mayoría de programadores más experimentados aprendieron emacs o vi en algún momento de su educación o carrera profesional. Llámalo "inercia" o "pereza", pero se necesita una gran mejora en la productividad para que un programador cambie sus herramientas.

- Editar no es lo difícil: los IDEs facilitan algunas cosas (autocompletado, refactorización automática). Pero creo que lo más difícil del trabajo de un programador no es la introducción del código en sí, sino decidir qué bibliotecas usar, elegir el algoritmo adecuado, tomar decisiones sobre las compensaciones y diseñar las estructuras de datos para que sean claras y eficaces. Estas son cosas en las que un IDE ni siquiera puede ayudar.

- Personalización: múltiples organizaciones han añadido algunas bibliotecas a emacs y vi que realizan funciones útiles similares a las de un IDE, y con algo de programación es posible escribir algunas funciones que permitan ajustar algún aspecto de los editores.

- Herramientas que no son de edición: para explorar y comprender el código existente, en lugar de editarlo, los programadores disponen de otras herramientas. Lo mismo ocurre con la revisión de los cambios de código de otros usuarios; no se hace en emacs ni en vi.

- Estilo consistente: toda la base de código se puede adherir a una guía de estilo bastante estricta. Así, es posible identificar a simple vista qué identificadores son tipos, métodos, variables locales, variables miembro y variables globales. Y siempre se puede distinguir qué argumentos de una función son las entradas y las salidas. Así que no es necesario un IDE para eso.

- Estabilidad, en el sentido de constancia: emacs y vi nunca cambian. No hay que preocuparse de que se le obligue al programador a usar una actualización que le "mueva el piso" e interrumpa el flujo de trabajo.

Puede que al principio de la vida de un programador, las funciones de un IDE sean muy útiles, pero a medida que se adquiere experiencia, se es capaz de deambular por el programa mentalmente.

Esto no implica que los programadores que usan editores de texto sencillos sean de alguna manera "mejores" que quienes usan IDE. Ojalá existiera un IDE mejor que emacs o vi. Lo más importante, en en cualquier lugar que se programe, es que tú y tu equipo elijan las herramientas que les permitan ser lo más productivos posible.

7.1 Java

Java (véase [7]) es un lenguaje de programación de propósito general, concurrente, orientado a objetos, que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones «escriban el programa una vez y lo ejecuten en cualquier dispositivo (Write Once, Run Anywhere» o WORA)», lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para ejecutarse en otra.

El lenguaje de programación Java fue originalmente desarrollado por James Gosling, de Sun Microsystems (constituida en 1982 y posteriormente adquirida el 27 de enero de 2010 por la compañía Oracle), y publicado en 1995 como un componente fundamental de la plataforma Java de Sun Microsystems. Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son compiladas a bytecode (clase Java), que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente.

La compañía Sun desarrolló la implementación de referencia original para los compiladores de Java, máquinas virtuales y librerías de clases en 1991, y las publicó por primera vez en 1995. A partir de mayo de 2007, en cumplimiento de las especificaciones del Proceso de la Comunidad Java, Sun volvió a licenciar la mayoría de sus tecnologías de Java bajo la Licencia Pública General de GNU (véase [17]). Otros también han desarrollado implementaciones alternas a estas tecnologías de Sun, tales como el Compilador de Java de GNU y el GNU Classpath.

Orientado a Objetos La primera característica, orientado a objetos (OO), se refiere a un método de programación y al diseño del lenguaje. Aunque hay muchas interpretaciones para OO, una primera idea es diseñar el Software de forma que los distintos tipos de datos que usen, estén unidos a sus operaciones. Así, los datos y el código (funciones o métodos) se combinan en entidades llamadas objetos. Un objeto puede verse como un paquete que contiene el "comportamiento" (el código) y el "estado" (datos). El principio es separar aquello que cambia de las cosas que permanecen inalterables. Frecuentemente, cambiar una estructura de datos implica un cambio en el código que opera sobre los mismos, o viceversa. Esta separación en objetos coherentes e independientes ofrece una base más estable para el diseño de un

sistema Software. El objetivo es hacer que grandes proyectos sean fáciles de gestionar y manejar, mejorando como consecuencia su calidad y reduciendo el número de proyectos fallidos.

Otra de las grandes promesas de la programación orientada a objetos es la creación de entidades más genéricas (objetos) que permitan la reutilización del Software entre proyectos, una de las premisas fundamentales de la Ingeniería del Software.

La reutilización del Software ha experimentado resultados dispares, encontrando dos dificultades principales: el diseño de objetos realmente genéricos es pobremente comprendido y falta una metodología para la amplia comunicación de oportunidades de reutilización. Algunas comunidades de código abierto quieren ayudar en este problema dando medios a los desarrolladores para diseminar la información sobre el uso y versatilidad de objetos reutilizables y bibliotecas de objetos.

Independencia de la Plataforma La segunda característica, la independencia de la plataforma, significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de Hardware. Este es el significado de ser capaz de escribir un programa una vez y que pueda ejecutarse en cualquier dispositivo, tal como reza el axioma de Java, "write once, run anywhere".

Para ello, se compila el código fuente escrito en lenguaje Java, para generar un código conocido como "bytecode" (específicamente Java bytecode), instrucciones máquina simplificadas específicas de la plataforma Java. Esta pieza está "a medio camino" entre el código fuente y el código máquina que entiende el dispositivo destino. El bytecode es ejecutado entonces en la máquina virtual (JVM), un programa escrito en código nativo de la plataforma destino (que es el que entiende su Hardware), que interpreta y ejecuta el código. Además, se suministran bibliotecas adicionales para acceder a las características de cada dispositivo (como los gráficos, ejecución mediante hebras o threads, la interfaz de red) de forma unificada. Se debe tener presente que, aunque hay una etapa explícita de compilación, el bytecode generado es interpretado o convertido a instrucciones máquina del código nativo por «compilación al vuelo JIT (Just In Time)».

Hay implementaciones del compilador de Java que convierten el código fuente directamente en código objeto nativo, como GCJ. Esto elimina la etapa intermedia donde se genera el bytecode, pero la salida de este tipo de

compiladores sólo puede ejecutarse en un tipo de arquitectura.

La licencia sobre Java de Sun insiste en que todas las implementaciones sean "compatibles". Esto dio lugar a una disputa legal entre Microsoft y Sun, cuando este último alegó que la implementación de Microsoft no daba soporte a las interfaces RMI y JNI además de haber añadido características "dependientes" de su plataforma. Sun demandó a Microsoft y ganó por daños y perjuicios (unos 20 millones de dólares), así como una orden judicial forzando el acatamiento de la licencia de Sun. Como respuesta, Microsoft no ofrece Java con su versión de sistema operativo, y en recientes versiones de Windows, su navegador Internet Explorer no admite la ejecución de applets sin un «conector (Plugin)» aparte. Sin embargo, Sun y otras fuentes ofrecen versiones gratuitas para distintas versiones de Windows.

Las primeras implementaciones del lenguaje usaban una máquina virtual interpretada para conseguir la portabilidad. Sin embargo, el resultado eran programas que se ejecutaban comparativamente más lentos que aquellos escritos en C o C++. Esto hizo que Java se ganase una reputación de lento en rendimiento. Las implementaciones recientes de la JVM dan lugar a programas que se ejecutan considerablemente más rápido que las versiones antiguas, empleando diversas técnicas, aunque sigue siendo mucho más lentos que otros lenguajes.

La primera de estas técnicas es simplemente compilar directamente en código nativo como hacen los compiladores tradicionales, eliminando la etapa del bytecode. Esto da lugar a un gran rendimiento en la ejecución, pero tapa el camino a la portabilidad. Otra técnica, conocida como «compilación al vuelo JIT (Just In Time)», convierte el bytecode a código nativo cuando se ejecuta la aplicación. Otras máquinas virtuales más sofisticadas usan una "recompilación dinámica" en la que la VM es capaz de analizar el comportamiento del programa en ejecución y recompila y optimiza las partes críticas. La recompilación dinámica puede lograr mayor grado de optimización que la compilación tradicional (o estática), ya que puede basar su trabajo en el conocimiento que de primera mano tiene sobre el entorno de ejecución y el conjunto de clases cargadas en memoria. La compilación JIT y la recompilación dinámica permiten a los programas Java aprovechar la velocidad de ejecución del código nativo sin por ello perder la ventaja de la portabilidad en ambos.

La portabilidad es técnicamente difícil de lograr, y el éxito de Java en ese campo ha sido dispar. Aunque es de hecho posible escribir programas para la plataforma Java que actúen de forma correcta en múltiples plataformas

de distinta arquitectura, el gran número de estas con pequeños errores o inconsistencias llevan a que a veces se parodie el eslogan de Sun, "Write once, run anywhere" cómo "Write once, debug everywhere" (o "Escríbelo una vez, ejecútalo en cualquier parte" por "Escríbelo una vez, depúralo en todas partes").

El concepto de independencia de la plataforma de Java cuenta, sin embargo, con un gran éxito en las aplicaciones en el entorno del servidor, como los Servicios Web, los Servlets, los Java Beans, así como en sistemas empotrados basados en OSGi, usando entornos Java empotrados.

El Recolector de Basura En Java el problema de las fugas de memoria se evita en gran medida gracias a la «recolección automática de basura (o automatic garbage collector)». El programador determina cuándo se crean los objetos y el entorno en «tiempo de ejecución de Java (Java runtime)» es el responsable de gestionar el ciclo de vida de los objetos. El programa, u otros objetos, pueden tener localizado un objeto mediante una referencia a este. Cuando no quedan referencias a un objeto, el recolector de basura de Java borra el objeto, liberando así la memoria que ocupaba previniendo posibles fugas (ejemplo: un objeto creado y únicamente usado dentro de un método sólo tiene entidad dentro de este; al salir del método el objeto es eliminado). Aun así, es posible que se produzcan fugas de memoria si el código almacena referencias a objetos que ya no son necesarios; es decir, pueden aún ocurrir, pero en un nivel conceptual superior. En definitiva, el recolector de basura de Java permite una fácil creación y eliminación de objetos y mayor seguridad.

Instalación de Java e IDEs Existen diversas versiones de Java para Linux, la más usada es JDK de Oracle pero también está una versión abierta llamada OpenJDK, para instalar por ejemplo OpenJDK 17 en Debian GNU/Linux es necesario hacer:

```
# apt install default-jdk
```

o

```
# apt install openjdk-17-jre openjdk-17-jdk openjdk-17-doc
```

si se desea instalar solo el Run-Time JRE, para ello usamos:

```
# apt install default-jre
```

o

```
# apt install openjdk-17-jre
```

y si hay más de una versión instalada, podemos actualizar la versión por omisión de Java:

```
# update-java-alternatives -s java-1.17.0-openjdk-amd64
```

para conocer la versión instalada usamos:

```
$ java –version
```

Además, se pueden instalar diversas herramientas e IDEs para facilitar la programación en Java, para ello usar:

```
# apt install scite jedit kate gedit nedit emacs medit \
kscope geany geany-plugins editra qtcreator anjuta \
anjuta-extras codelite codelite-plugins tea vim-gtk \
mousepad eric neovim neovim-qt medit kwrite katepart
# apt install eclipse eclipse-cdt eclipse-pydev netbeans \
bluefish bluefish-plugins codeblocks codeblocks-contrib
# apt install fte fte-console fte-terminal nano joe vim \
vim-python-jedi vim-tlib vim-latexsuite vim-nox micro \
neovim kakoune vim-athena jed
# apt install kdiff3 meld diffuse dirdiff kompare numdiff \
colordiff dwdiff wdiff xxdiff tkdiff ndiff ccdiff xxdiff
# apt install alleyoop astyle c2html java2html code2html \
c2html autodia txt2html html2text
# apt install git git-all gitk gitg git-cola git-gui qgit tig \
vim-fugitive git-extras
# apt install mercurial
# apt install subversion rapidsvn
# apt install cvs tkcvs
```

Además, es posible instalar varios editores especializados de las páginas oficiales de sus proyectos:

https://netbeans.apache.org/download/index.html
https://www.eclipse.org/downloads/
http://brackets.io/
https://www.jetbrains.com/idea/download/#section=Linux
https://www.oracle.com/tools/downloads/Jdeveloper-12c-downloads.html
http://www.drjava.org/
https://www.jgrasp.org/
https://www.bluej.org/
http://www.jcreator.com/index.htm
https://codenvy.com/
https://atom.io/
https://www.sublimetext.com/

Compilar y Ejecutar Para compilar el archivo ejemplo.java usamos *javac* en línea de comandos mediante:

```
$ javac ejemplo.java
```

y lo ejecutamos con:

```
$ java ejemplo
```

Monitoreo de Aplicaciones Java Java Development Kit (JDK) provee binarios, herramientas y compiladores para el desarrollo de aplicaciones en Java. Además incluye una poderosa herramienta de monitoreo que es *jconsole*. Podemos ejecutarla en una terminal de Linux usando:

```
$ jconsole
```

al lanzar la aplicación, nos preguntará si deseamos hacer el monitoreo local o de algún equipo conectado en red. Si seleccionamos local, entonces podemos ver el consumo de las aplicaciones que corren en la máquina virtual de Java en tiempo real de CPU, memoria, Threads, Clases, etc.

Además podemos usar el comando *jps* (Java Virtual Machine Process Status) que nos permite conocer cada proceso que corre en la máquina virtual de Java. El uso básico es:

```
$ jps
```

pero podemos pedirle más información usando:

```
$ jps -v
```

esta última nos proporciona el indicador de proceso y el nombre de la clase o archivo *Jar* que se detecte en cada instancia.

Crear y Ejecutar Archivos .jar Un archivo *.jar* (Java ARchive) es un formato de archivo independiente de la plataforma que se utiliza para agregar muchos archivos de clase Java, metadatos y recursos asociados, como texto, imágenes, etc., en un solo archivo para su distribución.

Permite que los tiempos de ejecución de Java implementen de manera eficiente una aplicación completa en un archivo de almacenamiento y brinda muchos beneficios, como seguridad, sus elementos pueden comprimirse, acortar los tiempos de descarga, permite el sellado y control de versiones de paquetes, admite la portabilidad. También es compatible con el empaquetado para extensiones.

Para crear y ejecutar archivos .jar necesitamos hacer lo siguiente:

1. Primero comencemos escribiendo una clase Java simple con un método principal para una aplicación llamada MiApp, con fines de demostración.

   ```
   $ nano MiApp.java
   ```

   Copie y pegue el siguiente código en el archivo MiApp.java.

   ```
   public class MiApp {
       public static void main(String[] args){
           System.out.println("Solo ejecuta MiApp");
       }
   }
   ```

   Grabe el archivo y cierre este.

2 A continuación, necesitamos compilar y empaquetar la clase en un archivo JAR usando las utilidades javac y jar como se muestra:

   ```
   $ javac -d . MiApp.java
   $ ls
   $ jar cvf MiApp.jar MiApp.class
   $ ls
   ```

3 Una vez creado MiApp.jar, ahora podemos ejecutar el archivo usando el comando java como se muestra:

```
$ java -jar MiApp.jar
no main manifest attribute, in MiApp.jar
```

De la salida del comando anterior, encontramos un error. La JVM (Java Virtual Machine) no pudo encontrar nuestro atributo de manifiesto principal, por lo que no pudo ubicar la clase principal que contiene el método principal (public static void main (String [] args)).

El archivo JAR debe tener un manifiesto que contenga una línea con el formato Main-Class: classname que defina la clase con el método principal que sirve como punto de partida de nuestra aplicación.

4 Para corregir el error anterior, necesitaremos actualizar el archivo JAR para incluir un atributo de manifiesto junto con nuestro código. Creemos un archivo MANIFEST.MF:

```
$ nano MANIFEST.MF
```

Copie y pegue la siguiente línea en el archivo MANIFEST.MF:

```
Main-Class: MiApp
```

Guarde el archivo y agreguemos el archivo MANIFEST.MF a nuestro MiApp.jar usando el siguiente comando:

```
$ jar cvmf MANIFEST.MF MiApp.jar MiApp.class
```

5 Finalmente, cuando ejecutamos el archivo JAR nuevamente, debería producir el resultado esperado como se muestra en la salida:

```
$ java -jar MiApp.jar
Solo ejecuta MiApp
```

Para obtener más información, debemos consultar las páginas de manual de los comandos java, javac y jar.

```
$ man java
$ man javac
$ man jar
```

7.2 Python

Python (véase [9]) es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License, que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores (véase apéndice 14.2).

Python fue creado a finales de los ochenta por Guido van Rossum en el Centro para las Matemáticas y la Informática (CWI, Centrum Wiskunde & Informatica), en los Países Bajos, como un sucesor del lenguaje de programación ABC, capaz de manejar excepciones e interactuar con el sistema operativo Amoeba. Van Rossum es el principal autor de Python, y su continuo rol central en decidir la dirección de Python es reconocido, refiriéndose a él como Benevolente Dictador Vitalicio (en inglés: Benevolent Dictator for Life, BDFL).

Características y paradigmas Python es un lenguaje de programación multiparadigma. Esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación imperativa y programación funcional. Otros paradigmas están soportados mediante el uso de extensiones.

Python usa tipado dinámico y conteo de referencias para la administración de memoria.

Una característica importante de Python es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado enlace dinámico de métodos).

Otro objetivo del diseño del lenguaje es la facilidad de extensión. Se pueden escribir nuevos módulos fácilmente en *C* o *C++*. Python puede incluirse en aplicaciones que necesitan una interfaz programable.

Aunque la programación en Python podría considerarse en algunas situaciones hostil a la programación funcional tradicional del Lisp, existen bastantes analogías entre Python y los lenguajes minimalistas de la familia Lisp como puede ser Scheme.

El intérprete de Python estándar incluye un modo interactivo en el cual

se escriben las instrucciones en una especie de intérprete de comandos: las expresiones pueden ser introducidas una a una, pudiendo verse el resultado de su evaluación inmediatamente, lo que da la posibilidad de probar porciones de código en el modo interactivo antes de integrarlo como parte de un programa. Esto resulta útil tanto para las personas que se están familiarizando con el lenguaje como para los programadores más avanzados.

Existen otros programas, tales como IDLE, bpython o IPython, que añaden funcionalidades extra al modo interactivo, como el autocompletado de código y el coloreado de la sintaxis del lenguaje.

Elementos del lenguaje Python fue diseñado para ser leído con facilidad. Una de sus características es el uso de palabras donde otros lenguajes utilizarían símbolos. Por ejemplo, los operadores lógicos: !, || y &&, en Python se escriben; not, or y and, respectivamente. Curiosamente el lenguaje Pascal es junto con COBOL uno de los lenguajes con muy clara sintaxis y ambos son de la década de los 70. La idea del código claro y legible no es algo nuevo.

El contenido de los bloques de código (bucles, funciones, clases, etc.) es delimitado mediante espacios o tabuladores, conocidos como indentación, antes de cada línea de órdenes pertenecientes al bloque. Python se diferencia así de otros lenguajes de programación que mantienen como costumbre declarar los bloques mediante un conjunto de caracteres, normalmente entre llaves {}. Se pueden utilizar tanto espacios como tabuladores para indentar el código, pero se recomienda no mezclarlos.

Debido al significado sintáctico de la indentación, cada instrucción debe estar contenida en una sola línea. No obstante, si por legibilidad se quiere dividir la instrucción en varias líneas, añadiendo una barra invertida: \ al final de una línea, se indica que la instrucción continúa en la siguiente.

Variables Las variables se definen de forma dinámica, lo que significa que no se tiene que especificar cuál es su tipo de antemano y puede tomar distintos valores en otro momento, incluso de un tipo diferente al que tenía previamente. Se usa el símbolo = para asignar valores.

Módulos Existen muchas propiedades que se pueden agregar al lenguaje importando módulos, que son "minicódigos" (la mayoría escritos también en Python) que proveen de ciertas funciones y clases para realizar determinadas tareas. Un ejemplo es el módulo: Tkinter, que permite crear interfaces grá-

ficas basadas en la biblioteca Tk. Otro ejemplo es el módulo: os, que provee acceso a muchas funciones del sistema operativo. Los módulos se agregan a los códigos escribiendo la palabra reservada *import* seguida del nombre del módulo que queramos usar.

Python tiene una gran biblioteca estándar, usada para una diversidad de tareas. Esto viene de la filosofía "pilas incluídas" ("batteries included") en referencia a los módulos de Python[121]. Los módulos de la biblioteca estándar pueden mejorarse por módulos personalizados escritos tanto en *C* como en Python. Debido a la gran variedad de herramientas incluidas en la biblioteca estándar, combinada con la habilidad de usar lenguajes de bajo nivel como *C* y *C++*, los cuales son capaces de interactuar con otras bibliotecas, Python es un lenguaje que combina su clara sintaxis con el inmenso poder de lenguajes menos elegantes.

Algunos Módulos para Python

TensorFlow Models sirve para el aprendizaje automático y aprendizaje profundo. TensorFlow Models es el repositorio de fuente abierta para encontrar muchas bibliotecas y modelos relacionados con el aprendizaje profundo.

Keras es una API de redes neuronales de alto nivel, escrita en Python y es capaz de ejecutarse sobre TensorFlow, CNTK o Theano. Fue desarrollado con un enfoque para permitir la experimentación rápida.

Frasco es un framework ligero de aplicaciones Web WSGI. está diseñado para que el inicio sea rápido y fácil, con la capacidad de escalar hasta aplicaciones complejas. Comenzó como un simple envoltorio alrededor de Werkzeug y Jinja y se ha convertido en uno de los frameworks de aplicación Web Python más populares.

Scikit-learn es un módulo de Python para el aprendizaje automático construido sobre SciPy y distribuido bajo la licencia BSD.

[121] Una lista de módulos disponibles en Python está en su página oficial.
Para la versión 2 en: https://docs.python.org/2/py-modindex.html
Para la versión 3 en: https://docs.python.org/3/py-modindex.html

Zulip es una poderosa aplicación de chat grupal de código abierto que combina la inmediatez del chat en tiempo real con los beneficios de productividad de las conversaciones enhebradas. Zulip es utilizado por proyectos de código abierto, compañías de Fortune 500, cuerpos de grandes estándares y otros que necesitan un sistema de chat en tiempo real que les permita a los usuarios procesar fácilmente cientos o miles de mensajes al día. Con más de 300 colaboradores que fusionan más de 500 commits por mes, Zulip es también el proyecto de chat grupal de código abierto más grande y de más rápido crecimiento.

Django es un framework Web Python de alto nivel que fomenta un desarrollo rápido y un diseño limpio y pragmático de desarrollo Web de código abierto, escrito en Python, que respeta el patrón de diseño conocido como modelo-vista-template. Fue desarrollado en origen para gestionar varias páginas orientadas a noticias de la World Company de Lawrence, Kanzas y fue liberada al público bajo una licencia BSD en julio del 2005.

Rebound es una herramienta de línea de comandos que obtiene instantáneamente los resultados de desbordamiento de pila cuando se produce un error de compilación.

Google Images Download Este es un programa de línea de comando de Python para buscar palabras clave / frases clave en Google Imágenes y opcionalmente descargar imágenes a su computadora. También puede invocar este script desde otro archivo Python.

YouTube-dl es usado para descargar videos de: youtube.com u otras plataformas de video.

System Design Primer este repositorio es una colección organizada de recursos para ayudar a aprender a construir sistemas a escala.

Mask R-CNN es para detección y segmentación de objetos. Esta es una implementación de Mask R-CNN en Python 3, Keras y TensorFlow. El modelo genera cuadros de delimitación y máscaras de segmentación para cada instancia de un objeto en la imagen. Se basa en Feature Pyramid Network (FPN) y ResNet101 backbone.

Face Recognition es usado para reconocer y manipular caras desde Python o desde la línea de comandos con la biblioteca de reconocimiento facial más simple del mundo. Esto también proporciona una herramienta de línea de comandos: face_recognition simple que permite hacer reconocimiento de rostros en una carpeta de imágenes desde la línea de comandos.

Snallygaster Herramienta para buscar archivos secretos en servidores HTTP.

Ansible es un sistema de automatización de TI radicalmente simple. Maneja la administración de configuraciones, la implementación de aplicaciones, el aprovisionamiento en la nube, la ejecución de tareas ad-hoc y la orquestación multinodo, incluida la trivialización de cosas como actualizaciones continuas de tiempo de inactividad cero con balanceadores de carga.

Detectron es el sistema de software de Facebook AI Research que implementa algoritmos de detección de objetos de última generación, incluyendo Mask R-CNN, está escrito en Python y funciona con el marco de aprendizaje profundo Caffe2.

Asciinema registrador de sesión de terminal y el mejor compañero de asciinema.org.

HTTPie es un cliente HTTP de línea de comando. Su objetivo es hacer que la interacción de la CLI con los servicios Web sea lo más amigable posible para los humanos. Proporciona un comando http simple que permite el envío de solicitudes HTTP arbitrarias utilizando una sintaxis simple y natural, y muestra una salida coloreada. HTTPie se puede usar para probar, depurar y, en general, interactuar con servidores HTTP.

You-Get es una pequeña utilidad de línea de comandos para descargar contenidos multimedia (videos, audios, imágenes) desde la Web, en caso de que no haya otra forma práctica de hacerlo.

Sentry es un servicio que ayuda a controlar y corregir fallas en tiempo real. El servidor está en Python, pero contiene una API completa para enviar eventos desde cualquier lenguaje, en cualquier aplicación.

Tornado es un framework Web de Python y una biblioteca de red asíncrona, desarrollada originalmente en FriendFeed. Mediante el uso de E/S de red sin bloqueo, Tornado puede escalar a decenas de miles de conexiones abiertas, lo hace ideal para largos sondeos, WebSockets y otras aplicaciones que requieren una conexión de larga duración para cada usuario.

Magenta es un proyecto de investigación que explora el papel del aprendizaje automático en el proceso de creación de arte y música. Principalmente, esto implica desarrollar nuevos algoritmos de aprendizaje profundo y aprendizaje de refuerzo para generar canciones, imágenes, dibujos y otros materiales. Pero también es una exploración en la construcción de herramientas e interfaces inteligentes que permiten a artistas y músicos ampliar sus procesos utilizando estos modelos.

ZeroNet crea sitios Web descentralizados utilizando Bitcoin Crypto y la red BitTorrent.

Gym OpenAI Gym es un conjunto de herramientas para desarrollar y comparar algoritmos de aprendizaje de refuerzo. Esta es la biblioteca de código abierto de Gym, que le da acceso a un conjunto estandarizado de entornos.

Pandas es un paquete de Python que proporciona estructuras de datos rápidas, flexibles y expresivas diseñadas para que trabajar con datos "relacionales" o "etiquetados" sea fácil e intuitivo. Su objetivo es ser el componente fundamental de alto nivel para hacer un análisis práctico y real de datos en Python. Además, tiene el objetivo más amplio de convertirse en la herramienta de análisis / manipulación de datos de código abierto más potente y flexible disponible en cualquier lenguaje.

Luigi es un paquete de Python que te ayuda a construir tuberías complejas de trabajos por lotes. Maneja la resolución de dependencia, la administración del flujo de trabajo, la visualización, el manejo de fallas, la integración de línea de comando y mucho más.

SpaCy (by Explosion AI) es una biblioteca para el procesamiento avanzado del lenguaje natural en Python y Cython, está basado en las últimas investigaciones y fue diseñado desde el primer día para ser utilizado en productos reales. SpaCy viene con modelos estadísticos precompilados y vectores de palabras, y actualmente admite tokenización para más de 20 lenguajes. Cuenta con el analizador sintáctico más rápido del mundo, modelos de redes neuronales convolucionales para etiquetado, análisis y reconocimiento de una entidad nombrada y fácil integración de aprendizaje profundo.

Theano es una biblioteca de Python que permite definir, optimizar y evaluar expresiones matemáticas que involucran matrices multidimensionales de manera eficiente. Puede usar GPU y realizar una diferenciación simbólica eficiente.

TFlearn es una biblioteca de aprendizaje profundo modular y transparente construida sobre Tensorflow. Fue diseñada para proporcionar una API de nivel superior a TensorFlow con el fin de facilitar y agilizar la experimentación, sin dejar de ser totalmente transparente y compatible con ella.

Kivy es un framework Python de código abierto y plataforma para el desarrollo de aplicaciones que hacen uso de interfaces de usuario innovadoras y multitáctiles. El objetivo es permitir un diseño de interacción rápido y fácil y un prototipado rápido a la vez que hace que su código sea reutilizable.

Mailpile es un cliente de correo electrónico moderno y rápido con características de cifrado y privacidad fáciles de usar. El desarrollo de Mailpile esta financiado por una gran comunidad de patrocinadores y todo el código relacionado con el proyecto es y será lanzado bajo una licencia de Software Libre aprobada por OSI.

Matplotlib es una biblioteca de trazado 2D de Python que produce figuras con calidad de publicación en una variedad de formatos impresos y entornos interactivos en todas las plataformas. Matplotlib se puede utilizar en scripts Python, el shell Python e IPython, así como en servidores de aplicaciones Web y varios toolkits de interfaz gráfica de usuario.

YAPF (by Google) toma el código y lo reformatea con el mejor formato que cumpla con la guía de estilo, incluso si el código original no viola la guía de estilo.

Cookiecutter una utilidad de línea de comandos que crea proyectos desde cookiecutters (plantillas de proyecto), por ejemplo creando un proyecto de paquete Python a partir de una plantilla de proyecto de paquete Python.

HTTP Prompt es un cliente HTTP interactivo de línea de comandos con autocompletado y resaltado de sintaxis, basado en prompt_toolkit y HTTPie.

Speedtest-cli interfaz de línea de comandos para probar el ancho de banda de Internet con speedtest.net: http://www.speedtest.net/

Pattern es un módulo de minería Web para Python. Tiene herramientas para Minería de datos, Procesamiento de lenguaje natural, Aprendizaje automático y Análisis de red.

Gooey (Beta) convierte (casi) cualquier programa de consola Python 2 o 3 en una aplicación GUI con una línea.

Wagtail CMS es un sistema de gestión de contenido creado en Django. Se centra en la experiencia del usuario y ofrece un control preciso para diseñadores y desarrolladores.

Bottle es un micro-Framework WSGI rápido, simple y liviano para Python. Se distribuye como un módulo de archivo único y no tiene dependencias distintas de la biblioteca estándar de Python.

Prophet (by Facebook) es un procedimiento para pronosticar datos de series temporales. Se basa en un modelo aditivo en el que las tendencias no lineales se ajustan a la estacionalidad anual y semanal, más las vacaciones. Funciona mejor con datos de periodicidad diaria con al menos un año de datos históricos. Prophet es robusto para datos faltantes, cambios en la tendencia y grandes valores atípicos.

Falcon es un marco Web de Python confiable y de alto rendimiento para construir Backend de aplicaciones a gran escala y microservicios. Fomenta el estilo arquitectónico REST e intenta hacer lo mínimo posible sin dejar de ser altamente efectivo.

Mopidy es un servidor de música extensible escrito en Python. Mopidy reproduce música desde el disco local, Spotify, SoundCloud, Google Play Music y más. Edita la lista de reproducción desde cualquier teléfono, tableta o computadora usando una gama de clientes MPD y Web.

Hug tiene como objetivo hacer que el desarrollar APIs impulsadas por Python sea lo más simple posible, pero no más simple. Como resultado, simplifica drásticamente el desarrollo de la API de Python.

SymPy es una biblioteca de Python para matemática simbólica.

Visdom es una herramienta flexible para crear, organizar y compartir visualizaciones de datos vivos y enriquecidos. Admite Torch y Numpy.

Pygame es una biblioteca de plataforma cruzada diseñada para facilitar la escritura de software multimedia, como juegos en Python.

Requests es una biblioteca de Python que le permite enviar solicitudes HTTP / 1.1, agregar encabezados, datos de formularios, archivos multiparte y parámetros con simples diccionarios de Python. También le permite acceder a los datos de respuesta de la misma manera.

Statsmodels es un paquete de Python que proporciona un complemento para Scipy para cálculos estadísticos que incluyen estadística descriptiva y estimación e inferencia para modelos estadísticos.

Scrapy es ampliamente utilizada en la biblioteca de raspado Web de Python. Se usa para crear programas de rastreo. Inicialmente, fue diseñado para raspar, como su nombre indica, pero ahora se usa para muchos propósitos, incluida la extracción de datos, las pruebas automatizadas, etc. Scrapy es de código abierto.

PyTorch es una biblioteca de código abierto, básicamente es un reemplazo de la biblioteca Numpy y está equipada con funcionalidades de nivel superior para construir redes neuronales profundas. Se puede usar otro lenguaje como Scipy, Cython y Numpy, que ayudan a extender PyTorch cuando sea necesario. Muchas organizaciones, incluyendo Facebook, Twitter, Nvidia, Uber y otras organizaciones usan Pytorch para la creación rápida de prototipos en investigación y para entrenar modelos de aprendizaje profundo.

Requests es una de las famosas bibliotecas de Python que tiene licencia bajo Apache2 y esta escrita en Python. Esta biblioteca ayuda a los humanos a interactuar con los lenguajes. Con la biblioteca de solicitudes, no es necesario que agregue consultas, cadenas manualmente a las URL ni codificar los datos POST. Se puede enviar solicitudes HTTP al servidor mediante la biblioteca de solicitudes y se puede agregar datos de formularios, contenido como encabezado, archivos en varias partes, etc.

PyFlux es una biblioteca de Python que se usa para predecir y analizar series temporales, está desarrollado por Ross Taylor, esta biblioteca tiene muchas opciones para la interfaz y contiene muchas clases nuevas de tipos de modelos. Pyflux permite a los usuarios implementar muchos modelos modernos de series de tiempo como GARCH y predecir la naturaleza de cómo reaccionará en el futuro.

Zappa es uno de los mejores paquetes de Python creados por Miserlou, es tan fácil de construir e implementar aplicaciones sin servidor en API Gateway y Amazon Web Services Lambda. Dado que AWS maneja la escala horizontal de forma automática, por lo que no habrá tiempo de espera de solicitud. Con Zappa, puede actualizar su código en una sola línea con Zappa.

Arrow es una famosa biblioteca de Python amigable para los humanos que ofrece funciones sensatas como crear, formatear, manipular y convertir fechas, horas y marcas de tiempo. Es compatible con Python 2 y 3 y es una alternativa de fecha y hora, ofrece funciones completas con una interfaz más agradable.

Pendulum es un paquete de Python que se utiliza para manipular fechas y horas, el código seguirá funcionando si se reemplazan todos los elementos de DateTime. Con Pendulum, se puede analizar DateTime y mostrar la fecha y hora con la zona horaria. Básicamente, Pendulum es una versión mejorada de la biblioteca Arrow y tiene todos los métodos útiles como redondear, truncar, convertir, analizar, formatear y aritmética.

Theano es una biblioteca de aprendizaje profundo de Python, que se utiliza para optimizar, definir y evaluar ecuaciones numéricas matemáticas y matrices multidimensionales, está desarrollado por el grupo de aprendizaje automático, por lo que, básicamente, Theano es un compilador de expresión matemática y proporciona una estrecha integración con Numpy y proporciona una optimización rápida y estable.

IPython esta es una de las herramientas de Python más útiles, ya que proporciona una rica arquitectura para el usuario. Esta herramienta permite escribir y ejecutar el código Python en el navegador. Ipython funciona en varios sistemas operativos, incluidos Windows, Mac OS X, Linux y la mayoría de los sistemas operativos Unix. IPython brinda todas las características que obtendrá en el intérprete básico con algunas características adicionales como números, más funciones, funciones de ayuda, edición avanzada, etc.

Imbalanced-learn en un mundo ideal, tendríamos conjuntos de datos perfectamente equilibrados y todos entrenaríamos modelos y seríamos felices. Desafortunadamente, el mundo real no es así, y ciertas tareas favorecen datos muy desequilibrados. Por ejemplo, al predecir el fraude en las transacciones de tarjetas de crédito, es de esperar que la gran mayoría de las transacciones (+ 99.9%) sean realmente legítimas. El entrenamiento ingenuo de algoritmos de ML conducirá a un rendimiento deprimente, por lo que se necesita cuidado adicional al trabajar con estos tipos de conjuntos de datos. Afortunadamente, este es un problema de investigación estudiado y existe una variedad de técnicas. Imbalanced-learn es un paquete de Python que ofrece implementaciones de algunas de esas técnicas, para hacer la vida mucho más fácil. Es compatible con Scikit-learn y es parte de los proyectos Scikit-learn-contrib.

Caffe2 el marco original de Caffe ha sido ampliamente utilizado durante años, y es conocido por su rendimiento incomparable y base de código probado en batalla. Sin embargo, las tendencias recientes en DL hicieron que el marco se estancara en algunas direcciones. Caffe2 es el intento de llevar Caffe al mundo moderno. Admite formación distribuida, implementación (incluso en plataformas móviles), las CPU más nuevas y Hardware compatible con CUDA. Si bien PyTorch puede ser mejor para la investigación, Caffe2 es adecuado para despliegues a gran escala como se ve en Facebook.

Dash es una biblioteca de código abierto para crear aplicaciones Web, especialmente aquellas que hacen un buen uso de la visualización de datos, en Python puro. esta construido sobre Flask, Plotly.js y React, y proporciona abstracciones que te liberan de tener que aprender esos Frameworks y permitirte ser productivo rápidamente. Las aplicaciones se representan en el navegador y responderán para que se puedan usar en dispositivos móviles. No se requiere JavaScript.

Fire es una biblioteca de código abierto que puede generar automáticamente una CLI para cualquier proyecto de Python. La clave aquí es automática: ¡casi no es necesario escribir ningún código o docstrings para construir una CLI!. Para hacer el trabajo, solo se tiene que llamar a un método Fire y pasarlo como se quiera para convertirlo en una CLI: una función, un objeto, una clase, un diccionario, o incluso no pasar ningún tipo de argumento (lo que convertirá todo el código en una CLI).

Flashtext es una biblioteca para búsqueda y reemplazo de palabras en un documento. La belleza de FlashText es que el tiempo de ejecución es el mismo sin importar cuántos términos de búsqueda se tenga, en contraste con la expresión regular en la que el tiempo de ejecución aumentará casi linealmente con el número de términos.

Pipenv con Pipenv, se especifican todas las dependencias en un Pipfile, que normalmente se genera mediante el uso de comandos para agregar, eliminar o actualizar dependencias. La herramienta puede generar un archivo Pipfile.lock, lo que permite que las compilaciones sean deterministas, ayudando a evitar esos errores difíciles de detectar debido a una dependencia poco clara que ni siquiera se cree que es necesaria.

Luminoth las imágenes están en todas partes hoy en día y comprender su contenido puede ser crítico para varias aplicaciones. Afortunadamente, las técnicas de procesamiento de imágenes han avanzado mucho, impulsadas por los avances en DL. Luminoth es un kit de herramientas de código abierto Python para visión artificial, construido con TensorFlow y Sonnet. Actualmente, viene de fábrica y es compatible con la detección de objetos en forma de un modelo llamado Faster R-CNN.

Instalación de Python e IDEs Para instalar Python 3 en Debian GNU/Linux es necesario hacer:

```
# apt install ipython3 python3 idle3 python3-pip \
python3-matplotlib python3-rpy2 python3-numpy spyder3 \
python3-scipy bpython3 python3-pandas python-sklearn \
python-sklearn-docspe python-wxgtk3.0 jython xonsh \
python3-mpi4pypython3-pyqt5 python3-pyqtgraph mypy \
python-wxgtk3.0-dev python3-numba pyflakes3
# apt install flake8 black
```

Para instalar Jupyter (entorno de trabajo orientado a científicos que soporta los lenguajes R y Python):

```
# apt install jupyter-console jupyter-notebook
# pip3 install jupyter
# pip3 install matplotlib
# pip3 install ipywidgets
# jupyter nbextension enable –py –sys-prefix \
widgetsnbextension
```

y podemos instalar PYREPOT usando:

```
# pip install pyreport
```

además podemos instalar editores especializados en Python usando:

```
# apt install eric pyzo pyzo-doc thonny
```

otras opciones se pueden descargar de:

https://www.jetbrains.com/pycharm/
http://www.pydev.org/
https://wingware.com/

También, se pueden instalar diversas herramientas e IDEs para facilitar la programación en Python, para ello usar:

```
# apt install scite jedit kate gedit nedit emacs medit \
kscope geany geany-plugins editra qtcreator anjuta \
anjuta-extras codelite codelite-plugins tea vim-gtk \
mousepad eric neovim neovim-qt medit kwrite katepart
# apt install fte fte-console fte-terminal nano joe vim \
vim-python-jedi vim-tlib vim-latexsuite vim-nox micro \
neovim micro kakoune vim-athena jed
# apt install kdiff3 meld diffuse dirdiff kompare numdiff \
colordiff dwdiff wdiff xxdiff tkdiff ndiff ccdiff xxdiff
# apt install alleyoop astyle c2html java2html code2html \
c2html autodia txt2html html2text moreutils
# apt install git git-all gitk gitg git-cola git-gui qgit tig \
vim-fugitive git-extras
# apt install mercurial
# apt install subversion rapidsvn
# apt install cvs tkcvs
```

Compilar y Ejecutar Para compilar el archivo ejemplo.py en *python3* en línea de comandos:

- para compilarlo y ejecutarlo en *python3* en línea de comandos usamos:

  ```
  $ python3 ejemplo.py
  ```

- en caso de necesitar depurar de forma interactiva un código, podemos usar el módulo *pdb*, por ejemplo:

  ```
  $ pdb ejemplo.py
  ```

- en caso de necesitar hacer una compilación estática podemos usar:

  ```
  $ pyflakes ejemplo.py
  ```

Codificar Según Guía de Estilo PEP8 Python cuenta con herramientas que permiten al programador conocer en que discrepa su código de la guía de estilo de Python PEP8 y en caso de que lo requiera, se puede usar estas para que reformatee nuestro código siguiendo al PEP8.

- en caso querer revisar las discrepancias de nuestro código con respecto a la guía oficial de estilo PEP8 de Python usamos:

  $ flake8 ejemplo.py

- en caso de querer conocer qué cambios se sugiere hacer al código según la guía oficial de estilo PEP8 de Python usamos:

  $ black –check ejemplo.py
  $ black –diff ejemplo.py

- en caso de querer reformatear el código según la guía oficial de estilo PEP8 de Python usamos:

  $ black ejemplo.py
  $ black *.py

Anaconda Por otro lado existe Anaconda, una Suite de código abierto que abarca una serie de aplicaciones, librerías y conceptos diseñados para el desarrollo de la Ciencia de datos con Python. En líneas generales Anaconda Distribu-tion es una distribución de Python que funciona como un gestor de entorno, un gestor de paquetes y que posee una colección de más de 720 paquetes de código abierto. Anaconda Distribution se agrupa en 4 sectores o soluciones tecnológicas; *Anaconda Navigator, Anaconda Project, las librerías de Ciencia de Datos y Conda.* Todas estas se instalan de manera automática y en un procedimiento muy sencillo.

Para más información ver: https://www.anaconda.com/.

También está SageMath, una Suite de código abierto bajo la licencia GPL de Software matemático como: N*umPy, SciPy, matplotlib, Sympy, Maxima, GAP, FLINT, R*, entre otros. Además combina acceso a una poderosa combinación del lenguaje basada en Python o directamente vía interfaces o *Wrappers*. La misión del proyecto es crear una alternativa de Software libre a *Magma, Maple, Mathematica y Matlab.*

Para más información ver: http://www.sagemath.org/.

Instalación de Aplicaciones Usando Pip Pip es un sistema de administración de paquetes que se utiliza para instalar y administrar paquetes de Software escritos en Python. Pip se usa principalmente para instalar paquetes disponibles en Python Package Index (PyPI, página del proyecto: https://pypi.org). Los desarrolladores también pueden usar Pip para instalar módulos y paquetes desarrollados localmente.

Para instalar Pip en Python 2 hacemos:

```
# apt install python-pip
```

y para instalar alguna aplicación para todos los usuarios, por ejemplo *ratarmount*, usamos:

```
# pip2 install ratarmount
```

y para instalar alguna aplicación para el usuario, por ejemplo *ratarmount*, usamos:

```
$ pip2 install –user ratarmount
```

Para instalar Pip en Python 3 hacemos:

```
# apt install python3-venv python3-pip
```

y para instalar alguna aplicación para todos los usuarios, por ejemplo *ratarmount*, usamos:

```
# pip3 install ratarmount
```

y para instalar alguna aplicación para el usuario, por ejemplo *ratarmount*, usamos:

```
$ pip3 install –user ratarmount
```

En caso de instalación para el usuario, para usar la aplicación debemos agregar al PATH:

```
export PATH ="$PATH:/home/$USER/.local/bin"
```

Sin pérdida de generalidad (usando pip2 o pip3), podemos ver los detalles de algún paquete, usando:

```
# pip3 show nombre
```

Podemos instalar algún paquete, usando:

```
# pip3 install nombre
```

Podemos actualizar algún paquete, usando:

```
# pip3 install –upgrade nombre
```

Podemos desinstalar algún paquete, usando:

```
# pip3 uninstall nombre
```

Podemos listar los paquetes instalados, usando:

```
# pip3 list nombre
```

Podemos buscar algún paquete, usando:

```
# pip3 search nombre
```

Podemos listar los paquetes desactualizados, usando:

```
# pip3 list –outdated
```

7.3 Julia

es un lenguaje de programación homoicónico, multiplataforma y multiparadigma (véase [8]) -soporta programación orientada a objetos, por procedimientos, funcional y meta además de multietapas- de tipado dinámico de alto nivel y alto desempeño para la computación genérica, técnica y científica, con una sintaxis similar a la de otros entornos de computación similares, con licencia MIT (véase apéndice 14.2).

Dispone de un compilador avanzado (JIT), mecanismos para la ejecución en paralelo y distribuida, además de una extensa biblioteca de funciones matemáticas. La biblioteca, desarrollada fundamentalmente en Julia, también contiene código desarrollado en C o Fortran, para el álgebra lineal, generación de números aleatorios, procesamiento de señales, y procesamiento

de cadenas. Adicionalmente, la comunidad de desarrolladores de Julia contribuye con la creación y distribución de paquetes externos a través del gestor de paquetes integrado de Julia a un paso acelerado. IJulia es el resultado de la colaboración entre las comunidades de IPython y Julia, provee de una poderosa interfaz gráfica basada en el navegador para Julia.

Algunas características básicas son:

- El despacho múltiple: permite definir el comportamiento de las funciones a través de diversas combinaciones de tipos de argumentos
- Sistema de tipado dinámico: tipos para la documentación, la optimización y el despacho de funciones
- Buen desempeño, acercándose al de lenguajes estáticamente compilados como C
- Gestor de paquetes integrado
- Macros tipo Lisp y otras herramientas para la meta-programación
- Llamar funciones de Python: mediante el paquete PyCall
- Llamar funciones de C directamente: sin necesidad de usar envoltorios u APIs especiales
- Poderosas características de línea de comandos para gestionar otros procesos
- Diseñado para la computación paralela y distribuida
- Corutinas: hilos ligeros
- Los tipos definidos por el usuario son tan rápidos y compactos como los tipos estándar integrados.
- Generación automática de código eficiente y especializado para diferentes tipos de argumentos
- Conversiones y promociones para tipos numéricos y de otros tipos, elegantes y extensibles
- Soporte eficiente para Unicode, incluyendo UTF-8 pero sin limitarse solo a este

- Es de uso general, pero tiene todo lo necesario para hacer ciencia de datos, aprendizaje automático, ecuaciones diferenciales, resolvedores de sistemas lineales, etc.

Julia incluye una terminal interactiva, llamada REPL en donde se puede visualizar automáticamente los resultados de la ejecución del programa o segmento de código. Tanto el compilador justo a tiempo (JIT) basado en LLVM así como el diseño del lenguaje le permiten a Julia acercarse e incluso igualar a menudo el desempeño de C. Julia no le impone al usuario ningún estilo de paralelismo en particular. En vez de esto, le provee con bloques de construcción clave para la computación distribuida, logrando hacer lo suficientemente flexible el soporte de varios estilos de paralelismo y permitiendo que los usuarios añadan más.

Instalación de Julia e IDEs Existen diversas versiones de Julia para Linux, para instalar en Debian GNU/Linux es necesario hacer:

```
# apt install julia julia-doc
```

Además, se pueden instalar diversas herramientas e IDEs para facilitar la programación en Julia, para ello usar:

```
# apt install scite jedit kate gedit nedit emacs medit \
kscope geany geany-plugins editra qtcreator anjuta \
anjuta-extras codelite codelite-plugins tea vim-gtk \
mousepad eric neovim neovim-qt medit kwrite katepart
# apt install eclipse eclipse-cdt eclipse-pydev netbeans \
bluefish bluefish-plugins codeblocks codeblocks-contrib
# apt install fte fte-console fte-terminal nano joe vim \
vim-python-jedi vim-tlib vim-latexsuite vim-nox micro \
neovim micro kakoune vim-athena jed
# apt install kdiff3 meld diffuse dirdiff kompare numdiff \
colordiff dwdiff wdiff xxdiff tkdiff ndiff ccdiff xxdiff
# apt install alleyoop astyle c2html java2html code2html \
c2html autodia txt2html html2text
# apt install git git-all gitk gitg git-cola git-gui qgit tig \
vim-fugitive git-extras
```

```
# apt install mercurial
# apt install subversion rapidsvn
# apt install cvs tkcvs
```

7.4 C y C++

C (véase [10]) es un lenguaje de programación originalmente desarrollado por Dennis Ritchie entre 1969 y 1972 en los Laboratorios Bell, como evolución del lenguaje anterior B, a su vez basado en BCPL. Es un lenguaje orientado a la implementación de Sistemas operativos, concretamente Unix, Linux y el Kernel de Linux. *C* es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear Software de sistemas, aunque también se utiliza para crear aplicaciones.

Se trata de un lenguaje de tipos de datos estáticos, débilmente tipificado, de medio nivel, ya que dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código *C* o acceder directamente a memoria o dispositivos periféricos.

Filosofía Uno de los objetivos de diseño del lenguaje *C* es que sólo sean necesarias unas pocas instrucciones en lenguaje máquina para traducir cada elemento del lenguaje, sin que haga falta un soporte intenso en tiempo de ejecución. Es muy posible escribir *C* a bajo nivel de abstracción; de hecho, *C* se usó como intermediario entre diferentes lenguajes.

En parte, a causa de ser relativamente de bajo nivel y tener un modesto conjunto de características, se pueden desarrollar compiladores de *C* fácilmente. En consecuencia, el lenguaje *C* está disponible en un amplio abanico de plataformas (más que cualquier otro lenguaje). Además, a pesar de su naturaleza de bajo nivel, el lenguaje se desarrolló para incentivar la programación independiente de la máquina. Un programa escrito cumpliendo los estándares e intentando que sea portátil puede compilarse en muchos computadores.

C se desarrolló originalmente (conjuntamente con el sistema operativo Unix, con el que ha estado asociado mucho tiempo) por programadores para programadores. Sin embargo, ha alcanzado una popularidad enorme, y se ha usado en contextos muy alejados de la programación de Software de sistemas, para la que se diseñó originalmente.

Propiedades Núcleo del lenguaje simple, con funcionalidades añadidas importantes, como funciones matemáticas y de manejo de archivos, proporcionadas por bibliotecas. Es un lenguaje muy flexible que permite programar con múltiples estilos. Uno de los más empleados es el estructurado "no llevado al extremo", entre sus principales propiedades destacan:

- Un sistema de tipos que impide operaciones sin sentido
- Usa un lenguaje de preprocesado, el preprocesador de *C*, para tareas como definir macros e incluir múltiples archivos de código fuente
- Acceso a memoria de bajo nivel mediante el uso de punteros
- Interrupciones al procesador con uniones
- Un conjunto reducido de palabras clave
- Por defecto, el paso de parámetros a una función se realiza por valor. El paso por referencia se consigue pasando explícitamente a las funciones las direcciones de memoria de dichos parámetros
- Punteros a funciones y variables estáticas, que permiten una forma rudimentaria de encapsulado y polimorfismo
- Tipos de datos agregados (struct) que permiten que datos relacionados (como un empleado, que tiene un id, un nombre y un salario) se combinen y se manipulen como un todo (en una única variable "empleado")

Carencias Aunque la lista de las características útiles de las que carece *C* es larga, éstos factores han sido importantes para su aceptación, porque escribir rápidamente nuevos compiladores para nuevas plataformas, mantiene lo que realmente hace el programa bajo el control directo del programador, y permite implementar la solución más natural para cada plataforma. Esta es la causa de que a menudo *C* sea más eficiente que otros lenguajes. Típicamente, sólo la programación cuidadosa en lenguaje ensamblador produce un código más rápido, pues da control total sobre la máquina, aunque los avances en los compiladores de *C* y la complejidad creciente de los microprocesadores modernos han reducido gradualmente esta diferencia, Algunas carencias son:

- Recolección de basura nativa, sin embargo se encuentran a tal efecto bibliotecas como la "*libgc*" desarrollada por Sun Microsystems, o el Recolector de basura de Boehm
- Soporte para programación orientada a objetos, aunque la implementación original de *C++* fue un preprocesador que traducía código fuente de *C++* a *C*. Véase también la librería *GObjec*t
- Funciones anidadas, aunque *GCC* tiene esta característica como extensión
- Soporte nativo para programación multihilo. Disponible usando librerías como libpthread

Ventajas estas se pueden resumir en:

- Lenguaje muy eficiente puesto que es posible utilizar sus características de bajo nivel para realizar implementaciones óptimas
- A pesar de su bajo nivel es el lenguaje más portado en existencia, habiendo compiladores para casi todos los sistemas de cómputo conocidos
- Proporciona facilidades para realizar programas modulares y/o utilizar código o bibliotecas existentes

Inconvenientes El mayor problema que presenta el lenguaje *C* frente a los lenguajes de tipo de dato dinámico es la gran diferencia en velocidad de desarrollo: es más lento programar en *C*, sobre todo para el principiante. La razón estriba en que el compilador de *C* se limita a traducir código sin apenas añadir nada. La gestión de la memoria es un ejemplo clásico: en *C* el programador ha de reservar y liberar la memoria explícitamente. En otros lenguajes (como *BASIC*, *MATLAB* o *C#*) la memoria es gestionada de forma transparente para el programador. Esto alivia la carga de trabajo humano y en muchas ocasiones evita errores, aunque también supone mayor carga de trabajo para el procesador.

El mantenimiento en algunos casos puede ser más difícil y costoso que con ciertos lenguajes de más alto nivel. El código en *C* se presta a sentencias cortas y enrevesadas de difícil interpretación.

Cabe destacar el contexto y época en la que fue desarrollado *C*. En aquellos tiempos existían muy pocos programadores, los cuales, a su vez, eran prácticamente todos expertos en el área. De esta manera, se asumía que los programadores eran conscientes de sus trabajos y capaces de manejar perfectamente el lenguaje. Por esta razón es muy importante que los recién iniciados adopten buenas prácticas a la hora de escribir en *C* y manejar la memoria, como por ejemplo un uso intensivo de indentación y conocer a fondo todo lo que implica el manejo de punteros y direcciones de memoria.

Aplicabilidad Hecho principalmente para la fluidez de programación en sistemas UNIX. Se usa también para el desarrollo de otros sistemas operativos como Windows o GNU/Linux. Igualmente para aplicaciones de escritorio como GIMP, cuyo principal lenguaje de programación es *C*.

De la misma forma, es muy usado en aplicaciones científicas (para experimentos informáticos, físicos, químicos, matemáticos, entre otros, conocidos como modelos y simuladores), industriales (industria robótica, cibernética, sistemas de información y base de datos para la industria petrolera y petroquímica). Predominan también todo lo que se refiere a simulación de máquinas de manufactura, simulaciones de vuelo (es la más delicada, ya que se tienen que usar demasiados recursos tanto de Hardware como de Software para desarrollar aplicaciones que permitan simular el vuelo real de una aeronave). Se aplica por tanto, en diversas áreas desconocidas por gran parte de los usuarios noveles.

Los equipos de cómputo de finales de los 90 son varios órdenes de magnitud más potentes que las máquinas en que C se desarrolló originalmente. Programas escritos en lenguajes de tipo dinámico y fácil codificación (Ruby, Python, Perl, etc.) que antaño hubieran resultado demasiado lentos, son lo bastante rápidos como para desplazar en uso a *C*. Aun así, se puede seguir encontrando código *C* en grandes desarrollos de animaciones, modelados y escenas en 3D en películas y otras aplicaciones multimedia.

Actualmente, los grandes proyectos de Software se dividen en partes, dentro de un equipo de desarrollo. Aquellas partes que son más "burocráticas" o "de gestión" con los recursos del sistema, se suelen realizar en lenguajes de tipo dinámico o de guión (script), mientras que aquellas partes "críticas", por su necesidad de rapidez de ejecución, se realizan en un lenguaje de tipo compilado, como *C* o *C++*. Si después de hacer la división, las partes críticas no superan un cierto porcentaje del total (aproximadamente el 10%)

entonces todo el desarrollo se realiza con lenguajes dinámicos. Si la parte crítica no llega a cumplir las expectativas del proyecto, se comparan las alternativas de una inversión en nuevo Hardware frente a invertir en el coste de un programador para que reescriba dicha parte crítica.

Ya que muchos programas han sido escritos en el lenguaje *C* existe una gran variedad de bibliotecas disponibles. Muchas bibliotecas son escritas en *C* debido a que *C* genera código objeto rápido; los programadores luego generan interfaces a la biblioteca para que las rutinas puedan ser utilizadas desde lenguajes de mayor nivel, tales como Java, Perl y Python.

C++ (véase [11]) es un lenguaje de programación diseñado a mediados de 1979 por Bjarne Stroustrup. La intención de su creación fue extender al lenguaje de programación C mecanismos que permiten la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, *C++* es un lenguaje híbrido.

Posteriormente se añadieron facilidades de programación genérica, que se sumaron a los paradigmas de programación estructurada y programación orientada a objetos. Por esto se suele decir que el *C++* es un lenguaje de programación multiparadigma.

Actualmente existe un estándar, denominado ISO *C++*, al que se han adherido la mayoría de los fabricantes de compiladores más modernos. Existen también algunos intérpretes, tales como ROOT.

Una particularidad de *C++* es la posibilidad de redefinir los operadores, y de poder crear nuevos tipos que se comporten como tipos fundamentales.

El nombre "*C++*" fue propuesto por Rick Mascitti en el año 1983, cuando el lenguaje fue utilizado por primera vez fuera de un laboratorio científico. Antes se había usado el nombre "*C* con clases". En *C++*, la expresión "*C++*" significa "incremento de *C*" y se refiere a que *C++* es una extensión de *C*.

El concepto de clase Los objetos en C++ son abstraídos mediante una clase. Según el paradigma de la programación orientada a objetos un objeto consta de:

- Identidad, que lo diferencia de otros objetos (Nombre que llevará la clase a la que pertenece dicho objeto).
- Métodos o funciones miembro

- Atributos o variables miembro

Diferencias de tipos respecto a C En C++, cualquier tipo de datos que sea declarado completo (fully qualified, en inglés) se convierte en un tipo de datos único. Las condiciones para que un tipo de datos T sea declarado completo son a grandes rasgos las siguientes:

- Es posible al momento de compilación conocer el espacio asociado al tipo de datos (es decir, el compilador debe conocer el resultado de sizeof(T))
- T Tiene al menos un constructor, y un destructor, bien declarados
- Si T es un tipo compuesto, o es una clase derivada, o es la especificación de una plantilla, o cualquier combinación de las anteriores, entonces las dos condiciones establecidas previamente deben aplicar para cada tipo de dato constituyente
- En general, esto significa que cualquier tipo de datos definido haciendo uso de las cabeceras completas, es un tipo de datos completo
- En particular, y a diferencia de lo que ocurría en *C*, los tipos definidos por medio de struct o enum son tipos completos. Como tales, ahora son sujetos a sobrecarga, conversiones implícitas, etcétera

Los tipos enumerados, entonces, ya no son simplemente alias para tipos enteros, sino que son tipos de datos únicos en *C++*. El tipo de datos bool, igualmente, también pasa a ser un tipo de datos único, mientras que en *C* funcionaba en algunos casos como un alias para alguna clase de dato de tipo entero.

Compiladores Uno de los compiladores libres de C++ es el de GNU, el compilador G++ (parte del proyecto GCC, que engloba varios compiladores para distintos lenguajes). Otros compiladores comunes son Intel C++ Compiler, el compilador de Xcode, el compilador de Borland C++, el compilador de CodeWarrior C++, el compilador g++ de Cygwin, el compilador g++ de MinGW, el compilador de Visual C++, Carbide.c++, entre otros.

Instalación de C y C++ e IDEs Existen diversas versiones de C y C++ para Linux, para instalarlos en Debian GNU/Linux es necesario hacer:

```
# apt install build-essential manpages-dev glibc-doc \
glibc-doc-reference gcc-doc-base gcc-doc splint \
c++-annotations-pdf g++ jam ohcount ctags \
cppcheck cccc autoconf automake make cmake scons
```

Además, se pueden instalar diversas herramientas e IDEs para facilitar la programación en C y C++, para ello usar:

```
# apt install scite jedit kate gedit nedit emacs medit \
kscope geany geany-plugins editra qtcreator anjuta \
anjuta-extras codelite codelite-plugins tea vim-gtk \
mousepad eric neovim neovim-qt medit kwrite katepart
# apt install eclipse eclipse-cdt eclipse-pydev netbeans \
bluefish bluefish-plugins codeblocks codeblocks-contrib
# apt install fte fte-console fte-terminal nano joe vim \
vim-python-jedi vim-tlib vim-latexsuite vim-nox micro \
neovim micro kakoune vim-athena jed
# apt install kdiff3 meld diffuse dirdiff kompare numdiff \
colordiff dwdiff wdiff xxdiff tkdiff ndiff ccdiff xxdiff
# apt install alleyoop astyle c2html java2html code2html \
c2html autodia txt2html html2text indent
# apt install git git-all gitk gitg git-cola git-gui qgit tig \
vim-fugitive git-extras
# apt install mercurial
# apt install subversion rapidsvn
# apt install cvs tkcvs
```

Compilar y Ejecutar Para compilar el archivo ejemplo.c en *C* en línea de comandos usamos:

```
$ gcc ejemplo.c -o ejemplo
```

y lo ejecutamos con:

```
$ ./ejemplo
```

Para compilar el archivo ejemplo.cpp en *C++* en línea de comandos usamos:

```
$ g++ ejemplo.cpp -o ejemplo
```

y lo ejecutamos con:

```
$ ./ejemplo
```

7.5 Fortran

Fortran (véase [12]) contracción del inglés The IBM Mathematical Formula Translating System, es un lenguaje de programación de alto nivel de propósito general, procedimental e imperativo, que está especialmente adaptado al cálculo numérico y a la computación científica. Desarrollado originalmente por IBM en 1957 para el equipo IBM 704, y usado para aplicaciones científicas y de ingeniería, el Fortran vino a dominar esta área de la programación desde el principio y ha estado en uso continuo por más de medio siglo en áreas de cómputo intensivo tales como la predicción numérica del tiempo, análisis de elementos finitos, dinámica de fluidos computacional, física computacional y química computacional. Es uno de los lenguajes más populares en el área de la computación de alto rendimiento y es el lenguaje usado para programas que evalúan el desempeño (benchmark) y el ranking de los supercomputadores más rápidos del mundo.

El Fortran abarca un linaje de versiones, cada una de las cuales evolucionó para añadir extensiones al lenguaje mientras que usualmente retenía compatibilidad con las versiones previas. Versiones sucesivas han añadido soporte para procesamiento de datos basados en caracteres (Fortran 77), programación de arreglos, programación modular y programación orientada a objetos (Fortran 90/95), y programación genérica (Fortran 2003/2008).

Ventajas e inconvenientes de su sintaxis como fue una primera tentativa de creación de un lenguaje de programación de alto nivel, tiene una sintaxis considerada arcaica por muchos programadores que aprenden lenguajes más modernos. Es difícil escribir un bucle "for", y errores en la escritura de un solo carácter pueden llevar a errores durante el tiempo de ejecución en vez de errores de compilación, en el caso de que no se usen las construcciones más frecuentes. Algunas de las primeras versiones no poseían

facilidades que son consideradas muy útiles, tal como la asignación dinámica de memoria.

Se debe tener en cuenta que la sintaxis de Fortran fue orientada para el uso en trabajos numéricos y científicos. Muchas de sus deficiencias han sido abordadas en revisiones recientes del lenguaje. Por ejemplo, en 1990 se presentó un tercer estándar ANSI conocido como FORTRAN 90, que contenía muchas nuevas características y permitía una programación más estructurada. Una serie de cambios menores se presentaron en 1995 conocido como FORTRAN 95 posee comandos mucho más breves para efectuar operaciones matemáticas con matrices y dispone de tipos y soporta OpenMP. Esto no solo mejora mucho la lectura del programa sino que además aporta información útil al compilador. Actualmente se tienen estándares para las versiones 2003 y 2008 y se continúa trabajando en mejoras al lenguaje.

Por estas razones Fortran prácticamente no se usa fuera de los campos científicos y del análisis numérico, pero permanece como el lenguaje preferido para desarrollar aplicaciones de computación numérica de alto rendimiento.

Características

Tipos de datos soportados:

- Numéricos (enteros, reales, complejos y de doble precisión).
- Boleanos (logical).
- Arreglos.
- Cadenas de caracteres.
- Archivos.

FORTRAN 90/95 ya es estructurado, y no requiere sentencias GOTO. Sólo admite dos ámbitos para las variables: local y global.

Variables y constantes

- Fortran no es sensible a mayúsculas y minúsculas. Los nombres de variables tienen de 6 a 31 caracteres máximo y deben comenzar por una letra. Los blancos son significativos.

- Declaración explícita de variables.
- Enteras (I-N), el resto reales. (se modifica con IMPLICIT).
- Punteros: en los primeros FORTRAN no hay punteros y todas las variables se almacenan en memoria estática. En FORTRAN 90 se declaran INTEGER, POINTER::P.
- Para memoria dinámica ALLOCATE y DEALLOCATE

Tipos de datos

- Arrays, pueden tener hasta 7 dimensiones y se guardan por columnas.
- REAL M(20),N(-5:5)
- DIMENSION I(20,20) (tipo por nomenclatura implícita)
- Cadenas de caracteres, el primer carácter es el 1, el operador // permite concatenar cadenas.
- CHARACTER S*10, T*25
- Almacenamiento de datos. Se usa COMMON para datos compartidos y EQUIVALENCE cuando almacenamos una variable con dos posibles tipos en la misma posición de memoria (como union en C). Se usa DATA para inicializar datos estáticos.
- DATA X/1.0/,Y/3.1416/,K/20/
- Tipos definidos por el usuario, con TYPE <nombre>... END TYPE <nombre>

Control de secuencia el conjunto de estructuras de control es limitado:

- Expresiones, prioridad de operadores
- Enunciados

◦ Asignación, cuando se hace entre cadenas hay ajuste de tamaño con blancos o truncamiento.

◦ Condicional. Permite IF ELSE IF... Para selección múltiple SELECT CASE CASE.....CASE DEFAULT.... END SELECT

◦ Iteración. DO....END DO

◦ Nulo, se usa solo para la etiqueta. CONTINUE.

- Control de subprogramas. CALL invoca al subprograma y RETURN devuelve un valor al programa llamante.
- Construcciones propensas a error: GOTO.

Entrada y salida

- Tipos de archivos:
  ◦ Secuenciales
  ◦ De acceso directo
- Comandos: READ, WRITE, PRINT, OPEN , CLASE, INQUIRE (propiedades o estado del archivo) REWIND y ENDFILE (para ubicar el puntero del fichero).
- Para el tratamiento de excepciones en las sentencias READ/WRITE se puede introducir la posición de la rutina de dicho tratamiento (ERR=90).

Subprogramas

- Hay tres tipos de subprogramas:

◦ Function, devuelven un solo valor de tipo numérico, lógico o cadena de caracteres.

◦ Subroutine, devuelve valores a través de variables no locales COMMON.

◦ Función de enunciado, permite calcular una sola expresión aritmética o lógica.

◦ FN(X,Y)=SIN(X)**2-COS(Y)**2

- Gestión de almacenamiento.
  ◦ Las variables son locales o globales (COMMON)
  ◦ Recursividad: RECURSIVE FUNCTION FACTORIAL(X)
  ◦ Parámetros de subprograma. Paso por referencia.

Abstracción y encapsulación. Evaluación del lenguaje

- La abstracción es posible mediante los subprogramas y el uso de variables COMMON, aunque su uso es propenso a errores.
- FORTRAN sigue siendo utilizado en el ámbito científico y es muy eficiente realizando cálculos.

  ◦ La estructura del programa suele ser difícil de entender.

  ◦ En FORTRAN 90/95 se incluye la recursividad y la memoria dinámica.

  ◦ Las etiquetas de las sentencias ya no son necesarias, ni el GOTO, pues se ha transformado en un lenguaje estructurado.

  ◦ El aspecto de los programas sigue siendo de procesamiento por lotes

Instalación de Fortran e IDEs Existen diversas versiones de Fortran para Linux, para instalarlos en Debian GNU/Linux es necesario hacer:

```
# apt install gfortran gfortran-doc fortran77-compiler \
fortran95-compiler fortran-compiler cfortran ftc fort77
```

Además, se pueden instalar diversas herramientas e IDEs para facilitar la programación en Fortran, para ello usar:

```
# apt install scite jedit kate gedit nedit emacs medit \
kscope geany geany-plugins editra qtcreator anjuta \
anjuta-extras codelite codelite-plugins tea vim-gtk \
mousepad eric neovim neovim-qt medit kwrite katepart
# apt install fte fte-console fte-terminal nano joe vim \
vim-python-jedi vim-tlib vim-latexsuite vim-nox micro \
neovim micro kakoune vim-athena jed
# apt install kdiff3 meld diffuse dirdiff kompare numdiff \
colordiff dwdiff wdiff xxdiff tkdiff ndiff ccdiff xxdiff
# apt install alleyoop astyle c2html java2html code2html \
c2html autodia txt2html html2text
# apt install git git-all gitk gitg git-cola git-gui qgit tig \
vim-fugitive git-extras
```

```
# apt install mercurial
# apt install subversion rapidsvn
# apt install cvs tkcvs
```

Compilar y Ejecutar Para compilar el archivo ejemplo.f77 en *Fortran 77* en línea de comandos usamos:

```
$ g77 ejemplo.f77 -o ejemplo
```

y lo ejecutamos con:

```
$ ./ejemplo
```

Para compilar el archivo ejemplo.f90 en *Fortran 90/95/2003* en línea de comandos usamos:

```
$ gfortran ejemplo.f90 -o ejemplo
```

y lo ejecutamos con:

```
$ ./ejemplo
```

7.6 R

El paquete R (véase [52]) es un lenguaje y entorno de programación para análisis estadístico y gráfico. Se trata de un proyecto de Software libre, resultado de la implementación GNU del premiado lenguaje S. SPSS, R y S-Plus —versión comercial de S— son, probablemente, los tres lenguajes más utilizados en investigación por la comunidad estadística, siendo además muy populares en el campo de la investigación biomédica, la bioinformática y las matemáticas financieras. A esto contribuye la posibilidad de cargar diferentes bibliotecas o paquetes con finalidades específicas de cálculo o gráfico.

Además, R puede integrarse con distintas bases de datos y existen bibliotecas que facilitan su utilización desde lenguajes de programación interpretados como Perl y Python. R soporta hacer interfase con lenguajes de programación como C, C++ y Fortran.

Otra de las características de R es su capacidad gráfica, que permite generar gráficos con alta calidad. R posee su propio formato para la documentación basado en LaTeX (véase [51]). R también puede usarse como

herramienta de cálculo numérico, campo en el que puede ser tan eficaz como otras herramientas específicas tales como FreeMat, GNU Octave y su equivalente comercial, MATLAB. Se ha desarrollado una interfaz RWeka para interactuar con Weka (véase [53]) que permite leer y escribir ficheros en el formato arff y enriquecer R con los algoritmos de minería de datos de dicha plataforma.

Los ambientes de desarrollo integrado para R existen como proyectos externos, como pueden ser editores —que sólo soportan la sintaxis—, los IDEs (Integrate Development Environments) y los GUI (Graphical User Interfaces) —permiten editar, ejecutar y depurar código desarrollado para R—. Hay más de 20 proyectos activos, tres de los más conocidos son Tinn-R (véase [54]), RStudio (véase [55]) y Rkward de KDE.

Instalación de R Existen diversos paquetes para *R* en Debian GNU/Linux, para instalarlos es necesario hacer:

```
# apt install r-base
# apt install 'r-cran*'
# apt install 'r-bioc*'
```

Editores nativo en Debian GNU/Linux

```
# apt install rkward rkward-data
```

y podemos generar gráficos de datos usando:

```
# apt install rlplot
```

para conocer todos los paquetes asociados, usar:

```
$ apt search ^r-cran
$ apt search ^r-bioc
```

Paquetes internos

```
R2WinBugs mcmcplots devtools Rattle e1071 FactiMineR \
Ellipse xlsx hsaur shiny
```

7.7 Herramientas de Programación

En Linux existe una gran variedad de herramientas para programación, ya que este sistema operativo fue hecho por programadores y para programadores, por ello entre las miles de herramientas, tenemos algunas que son ampliamente usadas, entre las que destacan:

Editores de Terminal

- Diakonos
- Jet
- Joe
- LE
- Mined
- Nano
- Nice Editor (NE)
- Pico
- Setedit
- Vim
- Fte

Editores Sencillos con Interfaz Gráfica

- Gedit
- SciTE
- JEdit
- NEdit
- MEdit

- KScope
- Editra
- Kate
- KWrite
- Leafpad
- Mousepad
- Anjunta
- TEA
- Pluma
- GVim
- Emacs

Editores Avanzados con Interfaz Gráfica

- Atom
- Bluefish
- BlueGriffon
- Brackets
- Geany
- Glade
- Google Web Designer
- KompoZer
- Light Table
- Notepadqq
- Scribes
- Sublime Text

Entornos de Programación Integrado (IDEs)

- Aptana
- Arduino IDE
- Android Studio
- CodeLite
- Code::Blocks
- Eclipse
- Gambas
- JetBrains Suite
- NetBeans
- Ninja-IDE
- Python IDLE
- PyDev
- Postman
- Qt Creator
- Simply Fortran
- Visual Studio Code
- Wing Python IDE
- Spyder
- PyCharm
- Jupyter
- Eric

Kit de Desarrollo de Software

- .Net Core SDK
- Android SDK
- Java JDK

Comparadores de texto y fuentes

- KDiff3
- Meld
- Diffuse
- DirDiff
- kompare
- Numdiff
- colordiff
- wdiff
- xxdiff
- tkdiff
- Ndiff

Otras Herramientas

- Alleyoop
- C2HTML
- Java2HTML
- Code2HTML
- c2html

- AutoDia
- txt2html
- html2text

Programas para control de versiones que permite desarrollo colaborativo de Software:

- Git https://git-scm.com/
- Mercurial https://www.mercurial-scm.org/
- Subversion https://subversion.apache.org/
- Perforce
- Bazaar
- CVS
- LibreSource
- Monotone
- SmartGit
- GitKraken
- Git Cola

Generadores automaticos de documentación que generan salida en PDF, HTML y XML para lenguajes como C++ y Java:

- Doxygen http://www.doxygen.org/
- JavaDoc

Formateador de código fuente para C, C++, Java y C#

- Astyle http://astyle.sourceforge.net

Lenguaje Unificado de Modelado (Unified Modeling Language)[122] forja un lenguaje de modelado visual común semántica y sintácticamente rico para la arquitectura, el diseño y la implementación de sistemas de Software complejos tanto en estructura como en comportamiento:

- UML https://www.uml.org/

Depuradores de programas

- ddd https://www.gnu.org/Software/ddd/
- gdb https://www.gnu.org/Software/gdb/
- kdbg http://www.kdbg.org/

Programas para rastrear errores en la manipulación de memoria y punteros desbordados

- Valgrind http://valgrind.org/
- DUMA http://duma.sourceforge.net/

Programas para hacer análisis de rendimiento[123]

- gprof https://sourceware.org/binutils/docs/gprof/
- Callgrind http://valgrind.org/docs/manual/cl-manual.html
- kCachegrind http://kCachegrind.sourceforge.net/html/Home.html
- time https://www.cyberciti.biz/faq/unix-Linux-time-command-examples-usage-syntax/

[122] Otras opciones son: UML Diagram Generation, Code Generation, Document Generation and Reporting, Scaling, Database Schema Generation, Entity Relationship Diagrams, Data Flow Datagrams, StarUML BOUML, EclipseUML, UML Modeller, Papyrus, Nclass, PlantUML, UMLet, NetBeansIDE, Open ModelSphere, gModeler, RISE, Oracle jdeveloper, Oracle SQL Developer, Dia, Kivio, ArgoUML, Xfig, etc.

[123] Otras opciones son: Splint, cppcheck, Rough Auditing Tool for Security, C y C++ Code Counter, CppNcss, Gnocchi, CUnit, CppUnit, OProfile, Intel VTune, Nemiver, Mudflap, etc.

En este apartado, solo tocaremos las más usadas, pero abunda la documentación de estas y otras importantes herramientas en línea de comandos (véase 3.4). Iniciaremos por las de compilar[124] y depurar[125] programas compilables en C, C++, Fortran, entre otros.

7.7.1 ¿Qué es eso de ASCII, ISO-8859-1 y UTF-8?

Los tres estándares representan el esfuerzo informático por brindar un sistema de codificación que permita representar los caracteres que se usan en todos los idiomas. El primer esfuerzo lo hizo *ASCII* y fue para el idioma inglés (128 caracteres), luego ante su insuficiencia para representar otros caracteres como los latinos por ejemplo, nace *ISO-8859-1* (también llamado *LATIN-1* ó *ASCII* extendido) pero debido a que no podía representar caracteres de otros idiomas aparece el estándar Unicode (del cual es parte *UTF-8*).

Un buen detalle a saber es que mientras *ISO-8859-1* usa un byte para representar un carácter, no pasa lo mismo con *UTF-8* que puede usar hasta 4 bytes ya que es de longitud variable. Esto hace que una base de datos en *UTF-8* sea un poco más grande que una en ISO-8859-1. Esto sucede porque -por ejemplo- mientras *ISO-8859-1* usa un byte para representar la letra ñ, *UTF-8* usa dos bytes. Hay un tema más y es que muchas veces cuando vamos a migrar información nos encontramos con caracteres *ISO-8859-1* (los correspondientes a los números 147, 148, 149, 150, 151 y 133) que no pueden verse en un editor *UNIX/LINUX* pero si en un navegador *HTML*.

Unicode es un set de caracteres universal, es decir, un estándar en el que se definen todos los caracteres necesarios para la escritura de la mayoría de los idiomas hablados en la actualidad que se usan en la computadora. Su objetivo es ser, y, en gran medida, ya lo ha logrado, un superconjunto de

[124] Un compilador es un programa informático que traduce un programa que ha sido escrito en un lenguaje de programación a un lenguaje común, usualmente lenguaje de máquina, aunque también puede ser traducido a un código intermedio (bytecode) o a texto y que reúne diversos elementos o fragmentos en una misma unidad, este proceso de traducción se conoce como compilación.

[125] Un depurador (en inglés, debugger), es un programa usado para probar y depurar (eliminar) los errores del programa "objetivo". El código a ser examinado puede alternativamente estar corriendo en un simulador de conjunto de instrucciones (ISS), una técnica que permite gran potencia en su capacidad de detenerse cuando son encontradas condiciones específicas pero será típicamente más lento que ejecutando el código directamente en el apropiado (o el mismo) procesador.

todos los sets de caracteres que se hayan codificado. El texto que aparece en la computadora o en la Web se compone de caracteres. Los caracteres representan letras del abecedario, signos de puntuación y otros símbolos.

En el pasado, distintas organizaciones han recopilado diferentes sets de caracteres y han creado codificaciones específicas para ellos. Un set puede abarcar tan sólo los idiomas de Europa occidental con base en el latín (sin incluir países de la Unión Europea como Bulgaria o Grecia), otro set puede contemplar un idioma específico del Lejano Oriente (como el japonés), y otros pueden ser parte de distintos sets diseñados especialmente para representar otro idioma de algún lugar del mundo.

Lamentablemente, no es posible garantizar que su aplicación particular pueda soportar todas las codificaciones, ni que una determinada codificación pueda soportar todos sus requerimientos para la representación de un cierto idioma. Además, generalmente resulta imposible combinar distintas codificaciones en la misma página Web o en una base de datos, por lo que siempre es muy difícil soportar páginas plurilingües si se aplican enfoques "antiguos" cuando se trata de tareas de codificación.

El Consorcio Unicode proporciona un único y extenso set de caracteres que pretende incluir todos los caracteres necesarios para cualquier sistema de escritura del mundo, incluyendo sistemas ancestrales (como el cuneiforme, el gótico y los jeroglíficos egipcios). Hoy resulta fundamental para la arquitectura de la Web y de los sistemas operativos, y las principales aplicaciones y navegadores Web incluyen soporte para este elemento. En el Estándar Unicode también se describen las propiedades y algoritmos necesarios para trabajar con caracteres. Este enfoque facilita mucho el trabajo con sistemas o páginas plurilingües y responde mucho mejor a las necesidades del usuario que la mayoría de los sistemas de codificación tradicionales.

Sets de caracteres, sets de caracteres codificados y codificaciones un set de caracteres o repertorio comprende el grupo de caracteres que se utilizarían para una finalidad específica, ya sea los necesarios para el soporte de los idiomas de Europa Occidental en la computadora.

Un set de caracteres codificados es un grupo de caracteres en el que se ha asignado un número exclusivo a cada carácter. Las unidades de un set de caracteres codificados se conocen como puntos de código. El valor de un punto de código representa la ubicación de un carácter en el set de caracteres codificados. Por ejemplo, el punto de código para la letra *á* en el set de

caracteres codificados Unicode es *225* en notación decimal, o *E1* en notación hexadecimal.

La codificación de caracteres refleja la manera en la que el set de caracteres codificados se convierte a bytes para su procesamiento en la computadora. Además, en Unicode existen distintas formas de codificar el mismo carácter. Por ejemplo, la letra *á* se puede representar mediante dos bytes en una codificación y con cuatro bytes, en otra. Los formatos de codificación que se pueden usar con Unicode se denominan *UTF-8, UTF-16 y UTF-32*.

Por todo lo anterior, al programar es necesario tener en cuenta la codificación usada por el editor o IDE que se use para ello y que no todos los editores soportan las mismas codificaciones[126], además puede haber problemas de portabilidad en los archivos entre distintos sistemas operativos. En el código fuente (las instrucciones del programa) no se suele usar caracteres distintos al *ASCII*, pero en las cadenas de visualización o en la documentación es común el uso de caracteres acentuados, es aquí donde hay que tomar una decisión sobre el usar o no dichos caracteres, siempre y cuando el compilador los soporte.

Si siempre se usa el mismo editor y la misma plataforma de desarrollo, no hay razón para no usar caracteres extendidos como los acentos. Pero si se usarán múltiples sistemas operativos y no hay garantía de usar editores que soporten dichos caracteres, entonces existe la posibilidad de perder dichos caracteres o bien pueden generar errores al compilar los archivos por no ser soportados. Por ello una opción para evitar problemas es sólo usar caracteres *ASCII* o tener el cuidado de usar editores que no pierdan dichos caracteres.

En Linux, para verificar la codificación de un archivo se utiliza el comando *file –i* o *–mime*, este comando permite mostrar en pantalla el tipo de archivo y su codificación, usando:

```
$ file -i Car.java
```

El comando *iconv* es utilizado para realizar esta tarea de convertir el código de un texto a otro. La lógica para aplicar correctamente el commando *iconv* es la siguiente:

```
$ iconv options -f from-encoding -t to-encoding inputfile(s) -o
outputfile
```

[126] Dado que los archivos fuente se intercambian entre usuarios y es común el uso de diferentes sistemas operativos, la conversiones de los caracteres entre diferentes formatos puede ser causa de problemas de codificación, perdiéndose dichos caracteres en la conversión.

así, *-f* o *–from-code* significa la entrada de la codificación, y *–t* o *–to-encoding* especifica la salida de la misma. Para enumerar todos los juegos de caracteres podemos usar *-l* o *–list*:

```
$ iconv -l
```

Con todo esto en mente podemos proceder a explicar la codificación de *UTF-8* a *ASCII*. Primero hay que comenzar con conocer las codificaciones de los caracteres en el archivo y luego poder ver el contenido del mismo. Así, se podrán convertir todos los archivos a la codificación *ASCII*. Todo después de haber utilizado el comando *iconv*, para poder verificar lo que contiene la salida del archivo. Para eso hay que hacer lo siguiente:

```
$ file -i input.file
$ cat input.file
$ iconv -f ISO-8859-1 -t UTF-8//TRANSLIT input.file -o
out.file
$ cat out.file
$ file -i out.file
```

Cabe destacar que, si el comando *//IGNORE* se añade a *to-encoding*, los caracteres no pueden ser convertidos y un error se mostrará luego de la conversión. También, si el comando *//TRANSLIT* es añadido a *to-encoding* como en el ejemplo dado *(ASCII//TRANSLIT)*, los caracteres convertidos son transliterados, si es posible, como necesarios.

Esto implicaría que en este evento los caracteres no pueden ser representados como se desea, aunque pueden haber aproximaciones del mismo, inclusive dos. Por lo que, si un carácter no puede ser transliterado, no será reconocido como un objetivo para reemplazo y se mostrará la marca (?) en la salida del archivo.

Algunas veces es necesario convertir el archivo de UTF-8 a ASCII y lo hacemos mediante:

```
$ iconv -f UTF-8 -t ISO-8859-1 prog.c -o progMod.c
```

o mediante:

```
$ iconv -f UTF-8 -t ASCII//TRASLIT prog.c -o progMod.c
```

7.7.2 Uso de Espacios o Tabuladores en Fuentes

En muchos lenguajes de programación la identación es una forma opcional de hacer legible el código para el programador, en otros (como Python) es imprescindible (ya que de ella dependerá su estructura).

¿Qué es la identación? En un lenguaje informático, la identación es lo que a la sangría al lenguaje humano escrito (a nivel formal). Así como para el lenguaje formal, cuando uno redacta una carta, debe respetar ciertas sangrías, los lenguajes informáticos, pueden requerir una identación.

No todos los lenguajes de programación necesitan una identación, aunque sí se estila implementarla, a fin de otorgar mayor legibilidad al código fuente. Una identación depende del lenguaje y estilo de codificación usado, por ejemplo en C, C++ y Java se acostumbra usar tres espacios en blanco. En el caso de Python se suele usar 4 espacios en blanco o un tabulador, que indicará que las instrucciones identadas forman parte de una misma estructura de control.

Los programadores siempre han debatido entre el uso de espacios y tabulaciones para estructurar su código. Los espacios y las tabulaciones son utilizados por los programadores para estructurar el código de una forma determinada. La primera línea de código (sin espacio o tabulación) inicia un "bloque" de contenido. Si las sucesivas líneas de código forman parte de ese mismo bloque (encerrado entre corchetes) o forman nuevos subbloques, estas se van desplazando hacia la derecha para indicar esa subordinación. En caso de formar un bloque completamente nuevo, se mantiene en la misma posición que la línea inmediatamente anterior.

A nivel funcional, la diferencia entre el uso de espacios o tabulaciones es nula. Cuando el código pasa por el compilador antes de ser ejecutado, la máquina interpreta de igual forma ambos formatos. No obstante, sí existen diferencias técnicas que marcan la diferencia entre el uso de tabulaciones y espacios:

- Precisión. Una tabulación no es más que un conjunto de espacios agrupados. Por norma general, este conjunto suele ser de 8 caracteres, pero puede variar. ¿Qué quiere decir esto? Que cuando un mismo fichero de código se abre en dos máquinas diferentes, la apariencia del código puede ser diferente. En cambio, el uso de espacios no conlleva este problema: un espacio siempre ocupa el mismo "espacio" —valga la

redundancia— y asegura que el código se visualiza de la misma forma en todas las máquinas.

- Comodidad. En el caso de las tabulaciones, basta con pulsar la tecla de tabulación una única vez para estructurar correctamente el código. En el caso de los espacios, es necesario pulsar varias veces la misma tecla para lograr la estructura deseada.

- Almacenamiento. El uso de tabulaciones también reduce el tamaño del fichero final, mientras que el uso de espacios lo aumenta. Lo mismo sucedería con el uso de espacios en lugar de saltos de línea.

Entonces, ¿cuál es la más correcta? La realidad es que todo depende de las preferencias personales. Si necesitas optimizar el tamaño de los ficheros al máximo, el uso de espacios se convierte en un sacrilegio. Si, en cambio, tu código debe lucir exactamente igual en múltiples máquinas, el uso de espacios puede ser más conveniente para lograr esa homogeneidad.

Por suerte, existen múltiples editores en la actualidad que trabajan y facilitan la transición entre ambos sistemas. Asimismo, los equipos de desarrollo de software establecen en sus guidelines el uso de espacios o tabulaciones. De esta forma, se evitan conflictos entre los programadores de un mismo proyecto y se alcanza esa homogeneidad tan deseada.

El comando *expand* y *unexpand* (que vienen instalados en los paquetes *GNU Core*) permite convertir tabuladores en espacios y viceversa, según nuestras necesidades o gustos. Estos comandos sacan el resultado de stdin o de los archivos nombrados en la línea de comando. Utilizando la opción *-t* se pueden establecer una o más posiciones de tabulador.

Para ver si se usan espacios o tabuladores en un archivo fuente podemos usar el comando *cat* con la opción *-T* que nos mostrará los caracteres tabulador como *^I*, ejemplo:

```
$ cat -T archivo
```

Para convertir los espacios en tabuladores (un tabulador igual a 8 espacios) usamos:

```
$ unexpand progEsp.c
```

o redireccionando la salida usando:

```
$ unexpand progEsp.c > progTab.c
```

Para convertir los tabuladores en espacios (1 tabulador por ejemplo 4 caracteres) usamos:

```
$ expand -t 4 progTab.c
```

o redireccionando la salida usando:

```
$ expand -t 4 progTab.c > progEsp.c
```

También es posible buscar todos los archivos (digamos *.cpp) y cambiar los tabuladores por 4 espacios (para ello usamos el comando *sponge* que está contenido en el paquete *moreutils*), mediante:

```
$ find . -name '*.cpp' -type f -exec bash -c \
'expand -t 4 "$0" | sponge "$0"' {} \;
```

7.7.3 Comparar Contenido de Fuentes

Cuando se programa es común generar distintas versiones de un mismo archivo, en GNU/Linux se tiene varias herramientas para comparar y combinar cambios. En la línea de comandos el comando *diff* permite ver los cambios entre dos versiones de un archivo y el comando *merge* sirve para combinar cambios. Por otro lado *sdiff* nos permite ver las diferencias entre dos archivos y de forma interactiva combinar cambios.

Pese a que son poderosos estos comandos, en forma gráfica se puede obtener todo su potencial. Algunas de estas opciones son:

```
# apt install kdiff3 meld diffuse dirdiff kompare wdiff  \
numdiff colordiff xxdiff tkdiff ndiff
```

Estos permiten comparar dos o tres versiones de un archivo simultáneamente, y hacerlo con el contenido de una o más carpetas. Cada uno tiene la capacidad de mostrar los cambios y si se requiere hacer la combinación de ellos.

meld nos muestra gráficamente las diferencias entre dos archivos o también, entre todos los archivos de dos directorios utilizando distintos colores, y nos permite editar estos archivos desde el propio programa, actualizando dinámicamente las diferencias. El programa incluye filtros y distintas ayudas para hacer la edición más sencilla, como flechas al lado de los cambios para aplicar cambio en cualquiera de los archivos con un simple clic. Este programa se puede utilizar como un sencillo cliente de control de cambios para Git, CVS, Subversion, etc.

kdiff3 nos muestra gráficamente las diferencias entre tres archivos utilizando distintos colores, y nos permite editar estos archivos desde el propio programa, actualizando dinámicamente las diferencias. El programa incluye filtros y distintas ayudas para hacer la edición más sencilla, como flechas al lado de los cambios para aplicar cambio en cualquiera de los archivos con un simple clic.

7.7.4 Astyle

Para dar uniformidad a la codificación de los programas fuente, se puede usar un formateador automático de código, *Astyle* soporta una gran variedad de lenguajes y opciones, para instalar en Debian GNU/Linux usar:

```
# apt install astyle
```

para formatear los archivos de C, C++, C# usar:

```
$ astyle -s3 -p –style=allman –lineend=Linux *.?pp
```

para Java, una opción es

```
$ astyle -s3 -p –style=java –lineend=Linux *.java
```

Algunos estilos disponibles son:

```
style=allman style=java style=kr style=stroustrup
style=whitesmith style=vtk style=ratliff style=gnu
style=Linux style=horstmann style=1tbs style=google
style=mozilla style=pico style=lisp
```

más opciones en:

- http://astyle.sourceforge.net/astyle.html
- https://en.wikipedia.org/wiki/Programming_style
- https://en.wikipedia.org/wiki/Indent_style

7.7.5 Compilación y la Optimización del Ejecutable

Al programar es necesario revisar nuestro código por un compilador y los errores son inherentes al proceso de programación. Los errores de programación responden a diferentes tipos y pueden clasificarse dependiendo de la fase en que se presenten. Algunos tipos de errores son más difíciles de detectar y reparar que otros, veamos entonces:

- Errores de sintaxis
- Advertencias
- Errores de enlazado
- Errores de ejecución
- Errores de diseño

Errores de sintaxis son errores en el código fuente. Pueden deberse a palabras reservadas mal escritas, expresiones erróneas o incompletas, variables que no han sido declaradas, etc. Los errores de sintaxis se detectan en la fase de compilación. El compilador, además de generar el código objeto, nos dará una lista de errores de sintaxis. De hecho nos dará sólo una cosa o la otra, ya que si hay errores no es posible generar un código objeto.

Advertencias además de errores, el compilador puede dar también advertencias (Warnings). Las advertencias son errores, pero no lo suficientemente graves como para impedir la generación del código objeto. No obstante, es importante corregir estos errores la mayoría de las veces, ya que ante un aviso el compilador tiene que tomar decisiones, y estas no tienen porqué coincidir con lo que nosotros pretendemos hacer, ya se basan en las directivas que los creadores del compilador decidieron durante la creación del compilador. Por lo tanto, en ocasiones, ignorar las advertencias puede ocasionar que nuestro programa arroje resultados inesperados o erróneos.

Errores de enlazado el programa enlazador también puede encontrar errores. Normalmente se refieren a funciones que no están definidas en ninguno de los ficheros objetos ni en las bibliotecas. Puede que hayamos olvidado incluir alguna biblioteca, o algún fichero objeto, o puede que hayamos olvidado definir alguna función o variable, o lo hayamos hecho mal.

Errores de ejecución incluso después de obtener un fichero ejecutable, es posible que se produzcan errores durante la ejecución del código. En el caso de los errores de ejecución normalmente no obtendremos mensajes específicos de error o incluso puede que no obtengamos ningún error, sino que simplemente el programa terminará inesperadamente. Estos errores son más difíciles de detectar y corregir (pues se trata de la lógica como tal de nuestra aplicación). Existen herramientas auxiliares para buscar estos errores, son los llamados depuradores (Debuggers). Estos programas permiten detener la ejecución de nuestros programas, inspeccionar variables y ejecutar nuestro programa paso a paso (instrucción a instrucción). Esto resulta útil para detectar excepciones, errores sutiles, y fallos que se presentan dependiendo de circunstancias distintas. Generalmente los errores en tiempo de ejecución se dan por situaciones no consideradas en la aplicación, por ejemplo, que el usuario ingrese una letra en vez de un número y ésto no es controlado.

Errores de diseño finalmente los errores más difíciles de corregir y prevenir. Si nos hemos equivocado al diseñar nuestro algoritmo, no habrá ningún programa que nos pueda ayudar a corregirlos, pues es imposible que un programa pueda determinar qué es lo que tratamos de conseguir o un programa que realice aplicaciones cualquiera por nosotros. Contra estos errores sólo cabe practicar y pensar, realizar pruebas de escritorio, hacerle seguimiento y depuración a la aplicación hasta dar con el problema (una mala asignación, un valor inesperado, olvidar actualizar una variable, etc.), también es útil buscar un poco de ayuda de libros o en sitios y foros especializados.

Compilación y la Optimización del Ejecutable Para usar muchas de estas herramientas (en línea de comandos), primero debemos conocer cómo compilar fuentes[127], sin pérdida de generalidad trabajaremos en C++ solici-

[127] Compilador para C es gcc, para C++ es g++, para Fortran es f77 o f95, etc.

tando que el archivo ejecutable[128] tenga el nombre *ejemp*:

```
$ g++ *.cpp -o ejemp
```

Para ejecutar el programa ya compilado:

```
$ ./ejemp
```

Para compilar y ver todos los avisos usar:

```
$ g++ -pedantic -Wall -Wextra -O *.cpp
```

o de forma alternativa:

```
$ g++ -Weffc++ *.cpp
```

Por otro lado, también podemos hacer una revisión estática del código, por ejemplo en C++ usamos:

```
$ cppcheck –enable=all *.?pp
```

mostrará los avisos de análisis estático del código indicado.

Para conocer el tiempo de ejecución[129] de un programa, podemos usar el comando básico *time*, mediante:

```
$ time ejecutable
```

que entregará información del tipo:

```
$ time ls

real    0m0.004s
user    0m0.001s
sys     0m0.004s
```

[128] Un archivo ejecutable es tradicionalmente un archivo binario con instrucciones en código de máquina cuyo contenido se interpreta por el ordenador como un programa. Además, suele contener llamadas a funciones específicas de un sistema operativo.

[129] El tiempo total de ejecución de un programa (tiempo real) es la suma del tiempo de ejecución del programa del usuario (tiempo de usuario) más el tiempo de ejecución del sistema necesario para soportar la ejecución (tiempo de sistema).

Pero podemos instalar una versión optimizada de este comando que proporciona información adicional, para ello instalar:

```
# apt install time
```

y su ejecución mediante:

```
$ /usr/bin/time ejecutable
```

por ejemplo para el comando *ls*, entrega una salida del tipo:

```
$ /usr/bin/time -v ls
   Command being timed: "ls"
   User time (seconds): 0.00
   System time (seconds): 0.00
   Percent of CPU this job got: 66%
   Elapsed (wall clock) time (h:mm:ss or m:ss): 0:00.00
   Average shared text size (kBytes): 0
   Average unshared data size (kBytes): 0
   Average stack size (kBytes): 0
   Average total size (kBytes): 0
   Maximum resident set size (kBytes): 2360
   Average resident set size (kBytes): 0
   Major (requiring I/O) page faults: 0
   Minor (reclaiming a frame) page faults: 110
   Voluntary context switches: 1
   Involuntary context switches: 1
   Swaps: 0
   File system inputs: 0
   File system outputs: 0
   Socket messages sent: 0
   Socket messages received: 0
   Signals delivered: 0
   Page size (Bytes): 4096
   Exit status: 0
```

Además, podemos compilar usando diversas optimizaciones[130] disponibles en todos los compiladores GNU de Linux, estas opciones de optimización están ordenadas, desde no optimizar, a la mejor optimización posible, estas son: –O0, O1, -O2, -O3, -O3 -march=native, -O3 -march=native -fito, -Ofast -march=native.

Para compilar y solicitar la optimización usamos:

```
$ g++ -O1 *.cpp
```

y para ejecutar el programa usamos:

```
$ ./a.out
```

El resultado de las optimizaciones dependen del programa y se puede ver que el rendimiento (tiempo de ejecución) mejora en varios órdenes de magnitud.

Por ejemplo en los siguientes test[131] se obtienen estos rendimientos:
Cryto++ v8.2:

O0 (95), -O2 (660.46), -O3 (712.01), -O3 -march=native (751.56), -O3 -march=native -fito (699.80), -Ofast -march=native (751.01)

LeelaChessZero:

O0 (18,300), -O2 (157,289), -O3 (142,198), -O3 -march=native (136,608), -O3 -march=native -fito (163,773), -Ofast -march=native (157,629)

Himeno Benchmark v3.0:

O0 (597.53), -O2 (4,150.11), -O3 (4,015.86), -O3 -march=native (4,771.42), -O3 -march=native -fito (4,774.03), -Ofast -march=native (5,065.07)

[130] La optimización de código es el conjunto de fases de un compilador que transforma un fragmento de código en otro fragmento con un comportamiento equivalente y se ejecuta de forma más eficiente, es decir, usando menos recursos de cálculo como memoria o tiempo de ejecución.

[131] https://www.phoronix.com/scan.php?page=news_item&px=GCC-10.1-Compiler-Optimizations

C-Ray v1.1:

O0 (113.58), -O2 (69.70), -O3 (38.00), -O3 -march=native (30.46), -O3 -march=native -fito (30.24), -Ofast -march=native (27.13)

Geometric Mean Of All Test Results:

O0 (222.36), -O2 (681.88), -O3 (709.76), -O3 -march=native (735.14), -O3 -march=native -fito (755.97), -Ofast -march=native (758.30)

7.7.6 Análisis de Rendimiento y Depuración

El comando *gprof* produce un perfil de ejecución de programas en C, C++, Pascal, FORTRAN77, etc. El efecto de las rutinas llamadas se incorpora en el perfil de cada llamador. Los datos del perfil se toman del fichero de perfil de grafos de llamada ('gmon.out' por omisión) que es creado por programas que se han compilado con la opción -pg de cc(1), pc(1), y f77(1). La opción -pg también enlaza al programa versiones de las rutinas de biblioteca que están compiladas para la perfilación. *Gprof* lee el fichero objeto dado (el predeterminado es 'a.out') y establece la relación entre su tabla de símbolos y el perfil de grafo de llamadas de 'gmon.out'. Si se especifica más de un fichero de perfil, la salida de gprof muestra la suma de la información de perfilado en los ficheros de perfil dados.

Gprof calcula la cantidad de tiempo empleado en cada rutina. Después, estos tiempos se propagan a lo largo de los vértices del grafo de llamadas. Se descubren los ciclos, y se hace que las llamadas dentro de un ciclo compartan el tiempo del ciclo. El primer listado muestra las funciones clasificadas de acuerdo al tiempo que representan incluyendo el tiempo de sus descendientes en su grafo de llamadas. Debajo de cada entrada de función se muestran sus hijos (directos) del grafo de llamadas, y cómo sus tiempos se propagan a esta función. Un despliegue similar sobre la función muestra cómo el tiempo de esta función y el de sus descendientes se propagan a sus padres (directos) del grafo de llamadas.

También se muestran los ciclos, con una entrada para el ciclo completo y un listado de los miembros del ciclo y sus contribuciones al tiempo y número de llamadas del ciclo. En segundo lugar, se da un perfil plano, similar al producido por *prof*. Este listado de los tiempos de ejecución totales, los números de llamadas, el tiempo en milisegundos que la llamada empleó en la propia rutina, y el tiempo en ms que la llamada empleó en la propia rutina

pero incluyendo sus descendientes. Finalmente, se proporciona un índice a los nombres de función.

Para obtener el análisis de rendimiento, hacemos:

```
$ g++ -g -pg -O0 *.cpp
$ ./a.out
$ gprof -c -z a.out gmon.out > sal.txt
```

el archivo *sal.txt* contiene el análisis de rendimiento detallado. Un ejemplo de esta salida es:

```
Flat profile:
Each sample counts as 0.01 seconds.
% cumulative self self total
time seconds seconds calls s/call s/call name
23.25    0.60    0.60  40656734     0.00     0.00 retorna(int, int)
14.85    0.98    0.38  27627674     0.00     0.00 retoaNumColu(int, int)
12.89    1.31    0.33  91126931     0.00     0.00 Vector::retorna(int)
10.94    1.59    0.28        31     0.01     0.03 ResJacobi::resuelve()
...
```

que permite conocer en que parte del código se consume más tiempo de ejecución.

Depuración con ddd un depurador (en inglés: Debugger) es un programa usado para probar y depurar (eliminar) los errores de otros programas (el programa "objetivo"). El código a ser examinado puede alternativamente estar corriendo en un simulador de conjunto de instrucciones (ISS), una técnica que permite gran potencia en su capacidad de detenerse cuando son encontradas condiciones específicas pero será típicamente algo más lento que ejecutando el código directamente en el apropiado (o el mismo) procesador. Algunos depuradores ofrecen dos modos de operación - la simulación parcial o completa, para limitar este impacto.

Si es un depurador de nivel de fuente o depurador simbólico, comúnmente ahora visto en entornos de desarrollo integrados, cuando el programa "se estrella" o alcanza una condición predefinida, la depuración típicamente muestra la posición en el código original. Si es un depurador de bajo nivel o un depurador de lenguaje de máquina, muestra la línea en el fuente desensamblado (a menos que también tenga acceso en línea al código fuente

original y pueda exhibir la sección apropiada del código del ensamblador o del compilador). Un "estrellamiento" sucede cuando el programa no puede continuar normalmente debido a un error de programación. Por ejemplo, el programa pudo haber intentado usar una instrucción no disponible en la versión actual del CPU o haber intentado tener acceso a memoria protegida o no disponible.

Típicamente, los depuradores también ofrecen funciones más sofisticadas tales como correr un programa paso a paso (un paso o animación del programa), parar el programa (Breaking), es decir, pausar el programa para examinar el estado actual en cierto evento o instrucción especificada por medio de un Breakpoint, y el seguimiento de valores de algunas variables. Algunos depuradores tienen la capacidad de modificar el estado del programa mientras que está corriendo, en vez de simplemente observarlo. También es posible continuar la ejecución en una posición diferente en el programa pasando un estrellamiento o error lógico.

La importancia de un buen depurador no puede ser exagerada. De hecho, la existencia y la calidad de tal herramienta para un lenguaje y una plataforma dadas a menudo puede ser el factor de decisión en su uso, incluso si otro lenguaje/plataforma es más adecuado para la tarea.

Para hacer depuración del código mediante el depurador gráfico *ddd* usar:

```
$ g++ -g -O0 *.cpp
$ ddd ./a.out
```

Puede usarse también los depuradores *xxgdb, gdb, kdbg, nevimer, lldb* cada uno tiene sus pros y contras, depende del usuario cual es el más adecuado para usar.

Depuración con valgrind es un conjunto de herramientas libres que ayuda en la depuración de problemas de memoria y rendimiento de programas.

La herramienta más usada es *Memcheck*. *Memcheck* introduce código de instrumentación en el programa a depurar, lo que le permite realizar un seguimiento del uso de la memoria y detectar los siguientes problemas:

- Uso de memoria no inicializada.
- Lectura/escritura de memoria que ha sido previamente liberada.

- Lectura/escritura fuera de los límites de bloques de memoria dinámica.
- Fugas de memoria.
- Otros.

El precio a pagar es una notable pérdida de rendimiento; los programas se ejecutan entre cinco y veinte veces más lento al usar Valgrind, y su consumo de memoria es mucho mayor. Por ello normalmente no siempre se ejecuta un programa en desarrollo usando Valgrind, sino que se usa en situaciones concretas cuando se está buscando un error determinado se trata de verificar que no haya errores ocultos como los que Memcheck puede detectar.

Valgrind incluye además otras herramientas:

- Addrcheck, versión ligera de Memcheck que se ejecuta más rápido y requiere menos memoria pero que detecta menos tipos de errores.
- Massif, mide el rendimiento del montículo (heap).
- Helgrind, herramienta de detección de condiciones de carrera (race conditions) en código multihilo.
- Cachegrind, mide el rendimiento de la Caché durante la ejecución, de acuerdo a sus características (capacidad, tamaño del bloque de datos, grado de asociatividad, etc.).

Para el rastreo de problemas con la manipulación de memoria y punteros desbordados usamos:

```
$ g++ -g -O0 *.cpp
$ valgrind –tool=memcheck –leak-check=yes –show-reachable=yes
./a.out
```

o analizar la salida usando *kCachegrind*:

```
$ valgrind –tool=callgrind ./a.out
$ kCachegrind profile.callgrind
```

Existen varios paquetes de modo gráfico para *valgrind*, uno de ellos es *alleyoop* y se usa:

```
$ alleyoop ./a.out -v –arg1=foo
```

otro es *kCachegrind*, podemos ver más opciones en:

- http://valgrind.org/
- http://alleyoop.sourceforge.net/usage.html
- http://kCachegrind.sourceforge.net/html/Home.html

7.7.7 Mejora del Rendimiento en Python

Python es un lenguaje interpretado, pero es posible compilar el código para generar Byte Code para el intérprete (no aumenta la velocidad de ejecución). Si se necesita crear un archivo *.pyc* para un módulo que no se importa, se puede usar los módulos *py_compile* y *compile_all* desde el intérprete de Python.

El módulo py_compile puede compilar manualmente cualquier módulo. Una forma de usar la función *py_compile.compile* en ese módulo de forma interactiva es:

```
>>> import py_compile
>>> py_compile.compile('modulo.py')
```

esto escribirá el archivo *modulo*.pyc.

En la línea de comando de Linux es posible compilar todos los archivos en un directorio utilizando el módulo compileall, para ello usar:

```
$ python -m compileall *.py
```

y ejecutar mediante:

```
$ python modulo.pyc
```

También es posible hacer ligera optimización del código generado mediante:

```
$ python –O -m compileall *.py
```

esto generará código Bytecode con extensión *.pyo*, y ejecutar mediante:

```
$ python modulo.pyo
```

Python es un lenguaje razonablemente rápido, pero no es tan rápido como los programas compilados de C o Fortran. Eso es porque se interpreta *CPython*, la implementación estándar. Para ser más precisos, su código de Python se compila en un código de *Bytes* que luego se interpreta. Eso es bueno para aprender, ya que puede ejecutar el código en el *REPL* de Python y ver los resultados de inmediato en lugar de tener que compilar y ejecutar. Pero como los programas de Python no son tan rápidos, los desarrolladores han creado varios compiladores de Python[132] a lo largo de los años, incluidos[133] Numba, *Nuitka, PyPy, Cython, cx_FreezeIron, Pythran, Jython entre otros.*

Comparando Compiladores de Python Alguien ya ha hecho el trabajo de crear un punto de referencia de Python. Opté por *PyStone*, una traducción de un programa en C de Guido van Rossum, el creador de Python (el programa en C era en sí mismo una traducción de un programa *Ada*). Encontré una versión convertida por el desarrollador Christopher Arndt en *Github* que era capaz de probar Python 3. Para dar un sentido de perspectiva, aquí está el rendimiento de *CPython* (es decir, Python estándar) con *Pystone*:

```
Python 2.7.15Rc1 2: 272.647 pystones / second.
Python 3.6.5: 175,817
```

Como puede ver, hay una gran diferencia entre Python 2 y 3 (cuanto más Pystones por segundo, mejor). En los siguientes desgloses, todos los compiladores de Python se compararon con Python 3.

Nuitka Aunque puede seguir las instrucciones en la página de descarga, lo siguiente en Debian GNU/Linux funcionó bien para mí:

```
$ apt install nuitka
```

[132] El rendimiento rápido no es la única razón para compilar; Posiblemente la mayor desventaja de los lenguajes de *Scripting* como Python es que se proporciona de manera implícita su código fuente a los usuarios finales.

[133] Si está interesado en los compiladores de Python en general, tenga en cuenta que hay mucho debate y controversia sobre los "mejores" compiladores y la rapidez general del lenguaje.

adicionalmente Nuitka también puede usar otro compilador de C (además del gcc), así que descargué clang. Puedes instalarlo con esto:

```
$ apt install clang
```

De forma predeterminada, *Nuitka* usa *gcc*, pero un parámetro te permite usar el *clang*, así que lo probé con ambos. El compilador clang es parte de la familia llvm, y está pensado como un reemplazo moderno para *gcc*. Compilar pystone.py con *gcc* fue tan simple como esto (primera línea), o con *clang* (segunda línea), y con la optimización del tiempo de enlace para *gcc* (tercera línea):

```
$ nuitka pystone.py
$ nuitka pystone.py –clang
$ nuitka pystone.py –lto
```

Después de compilar, lo que tomó aproximadamente 10 segundos, ejecuté el pystone.exe desde la terminal con:

```
$ ./pystone.exe 500000
```

Hice 500,000 pases:

Tamaño Ejecución pystones / seg.

1. 223.176 Kb 597,000
2. 195,424 Kb 610,000
3. 194.2 kb 600,000

Estos fueron los promedios de más de 5 corridas. Había cerrado tantos procesos como pude, pero tómo los tiempos con un poco de variación porque había un +/- 5% en los valores de tiempo.

PyPy Guido van Rossum dijo una vez: "Si quieres que tu código se ejecute más rápido, probablemente debas usar PyPy". Para instalarlo en Debian GNU/Linux usar:

```
$ apt install pypy
```

Entonces lo corrí así:

$ pypy pystone.py

El resultado fue una asombrosa cantidad de 1,776,001 pystones por segundo, casi tres veces más rápido que *Nuitka*.

PyPy usa un compilador justo a tiempo y hace algunas cosas muy inteligentes para alcanzar su velocidad. De acuerdo con los puntos de referencia reportados, es 7.6 veces más rápido que el *CPython* en promedio. Puedo creer eso fácilmente. La única (leve) desventaja es que siempre está un poco por detrás de las versiones de Python (es decir, hasta 2.7.13 (no 2.7.15) y 3.5.3 (no 3.6.5)). Producir un exe requiere un poco de trabajo. Tienes que escribir tu Python en un subconjunto llamado *RPython*.

Cython no es solo un compilador para Python; es para un superconjunto de Python que admite la interoperabilidad con C / C++. CPython está escrito en C, por lo que es un lenguaje que generalmente se combina bien con Python.

Configurar las cosas con *Cython* es un poco complicado. No es como *Nuitka*, que acaba de salir de la caja. Primero, debes comenzar con un archivo de Python con una extensión .pyx; ejecuta *Cython* para crear un archivo pystone.c a partir de eso:

$ cython pystone.pyx –embed

No omita el parámetro *-embed*. Se agrega en *main* y eso es necesario. A continuación, compila pystone.c con esta hermosa línea:

$ gcc $ (python3-config –includes) pystone.c -lpython3.6m -o pystone.exe

Si recibe algún error, como "no se puede encontrar la versión -lpython", podría ser el resultado de su versión de Python. Para ver qué versión está instalada, ejecute este comando:

$ pkg-config –cflags python3

Después de todo eso, *Cython* solo dio 228,527 pystones / sec. Sin embargo, *Cython* necesita que hagas un poco de trabajo especificando los tipos de variables. Python es un lenguaje dinámico, por lo que no se especifican los tipos; *Cython* utiliza la compilación estática y el uso de variables de tipo C le permite producir un código mucho mejor optimizado. (La documentación es bastante extensa y requiere lectura).

Tamaño Ejecución pystones / seg.

1. 219.552 Kb 228.527

cx_freeze es un conjunto de Scripts y módulos para "congelar" Scripts de Python en ejecutables, y se pueden encontrar en *Github*.

Lo instalé y creé una carpeta congelada para administrar cosas en:

$ pip3 install cx_Freeze –upgrade

Un problema que encontré con el Script de instalación fue un error que falta "lz". Necesitas tener instalado zlib; ejecuta esto para instalarlo:

$ apt install zlib1g-dev

Después de eso, el comando *cx_Freeze* tomó el Script *pystone.py* y creó una carpeta dist que contenía una carpeta *lib*, un archivo *lib* de 5MB y el archivo de aplicación pystone:

$ cxfreeze pystone.py –target-dir dist

Tamaño Ejecución pystones / seg.

1. 10,216 174,822

No es el rendimiento más rápido, porque es la misma velocidad que *CPython*. (La congelación de Python implica enviar su aplicación en un solo archivo (o carpeta) con los elementos Python necesarios, en lugar de compilar; significa que el destino no requiere Python).

Numba Este es un compilador "justo a tiempo" para Python que optimiza el código que se usa en algoritmos numéricos como son en las matrices, bucles y funciones de NumPy (también da soporte a *Threading*, vectorización *SIMD* y aceleración por *GPUs*: *Nvidia CUDA, AMD ROC*). La forma más común de usar Numba es a través de su colección de decoradores que se pueden aplicar a sus funciones para indicar a Numba que las compile usando el estándar *LLVM*. Cuando se realiza una llamada a una función decorada de Numba, se compila en el código de máquina "justo a tiempo" para su ejecución y todo o parte de su código puede ejecutarse posteriormente a la velocidad de código de máquina nativo. Numba también trabaja bien con Jupiter notebook para computación interactiva y con ejecución distribuida como *Dask* y *Spark*.

Se puede instalar en Debian GNU/Linux mediante:

```
$ apt install python3-numba
```

y se puede descargar mediante *CONDA* paquete de *Anaconda*, usando:

```
$ conda install numba
```

o mediante *PIP* usando:

```
$ pip install numba
```

Dando mejores resultados en la ejecución de múltiples pruebas que *PyPy*, pero no en todos los casos. Por ello, la recomendación es evaluar el rendimiento mediante pruebas en cada caso particular.

Conclusión Una buena opción es *PyPy* por el rendimiento obtenido en código general (y dependiendo del código en cuestión Numba puede ser mejor que *PyPy* en aplicaciones de cómputo científico), la compilación fue muy rápida y produjo los resultados en menos de un segundo después de presionar la tecla *RETURN*. Si requieres un ejecutable, sin embargo, te recomiendo Nuitka; fue una compilación sin complicaciones y se ejecuta más rápido que CPython. Experimenta con estos compiladores de Python y vea cuál funciona mejor para tus necesidades particulares.

7.7.8 Git

Git es un programa de control de versiones que sirve para la gestión de los diversos cambios que se realizan sobre los elementos de algún proyecto de Software y sus respectivos programas fuente o configuración del mismo guardando instantáneas (Snapshots) del código en un estado determinado, que viene dado por el autor y la fecha. Fue diseñado por Linus Torvalds y es usado para controlar los cambios de diversos proyectos como los fuentes del Kernel de Linux que tiene decenas de millones de líneas de código (en la versión 4.12 cuenta con 24,170,860 líneas de código repartidos en 59,806 archivos) y es trabajado por miles de programadores alrededor del mundo.

¿Qué es control de versiones? se define como control de versiones a la gestión de los diversos cambios (Commit es un conjunto de cambios guardados en el repositorio de Git y tiene un identificador SHA1 único) que se realizan sobre los elementos de algún producto o una configuración del mismo, es decir, es lo que se hace al momento de estar desarrollando un Software o una página Web. Exactamente es eso que haces cuando subes y actualizas tu código en la nube, o le añades alguna parte o simplemente editas cosas que no funcionan como deberían o al menos no como tú esperarías.

¿A que le llamamos sistema de control de versiones? son todas las herramientas que nos permiten hacer todas esas modificaciones antes mencionadas en nuestro código y hacen que sea más fácil la administración de las distintas versiones de cada producto desarrollado; es decir *Git*.

Antes de seguir avanzando, conviene definir algunos términos comunes que encontraremos más adelante:

- Un repositorio es una carpeta que contiene los archivos y subdirectorios de un proyecto. Puede ser público o privado, dependiendo de quién deba tener acceso.
- Una rama es una ruta de desarrollo separada en el mismo repositorio. En un mismo proyecto suelen utilizarse ramas separadas para trabajar en nuevas características sin interferir con la versión de producción. Una vez que el código es revisado y probado, un administrador del repositorio puede fusionar los cambios en la rama principal.

- Un commit es una instantánea de un repositorio en un momento dado. Permite a los programadores incluir comentarios y pedir la opinión de otras personas. Usando el Hash que lo identifica puedes volver a un estado anterior del proyecto si es necesario. Antes de que los archivos y directorios puedan ser comiteados, necesitamos decirle a Git que los rastree. Normalmente nos referimos a este paso simplemente como agregar los archivos al área de Staging.

- Un Pull Request (o simplemente PR) es un método para informar a otros desarrolladores y discutir los cambios recientes antes de incorporarlos a la ruta de desarrollo principal.

- Un Fork es un proyecto independiente que se basa en un repositorio determinado. A diferencia de las ramas, no es local a este último. Sin embargo, también puede fusionarse con él a través de un Pull Request adecuado.

- Se puede utilizar un archivo .gitignore para indicar qué contenido local no queremos incluir en el repositorio. Esto nos ayuda a evitar enviar archivos temporales o exponer información sensible en una solución basada en la Web.

Git fue creado pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente, es decir *Git* nos proporciona las herramientas para desarrollar un trabajo en equipo de manera inteligente y rápida y por trabajo nos referimos a algún Software o página que implique código el cual necesitemos hacerlo con un grupo de personas.

Algunas de las características más importantes de *Git* son:

- Rapidez en la gestión de ramas (Branche es como una línea de tiempo a partir de los Commit, siempre hay siempre como mínimo una rama principal predefinida llamada Master), debido a que *Git* nos dice que un cambio será fusionado mucho más frecuentemente de lo que se escribe originalmente.

- Gestión distribuida: Los cambios se importan como ramas adicionales y pueden ser fusionados de la misma manera como se hace en la rama local.

- Gestión eficiente de proyectos grandes.
- Realmacenamiento periódico en paquetes.

Con esto en mente, vamos a hacer una introducción a Git desde cero.

Instalación de Git para instalar *Git* completo en el servidor o en la máquina de trabajo, usamos:

```
# apt install git-all
```

Para instalar lo básico de *Git*, si no está instalado:

```
# apt install git
```

otras opciones para trabajar con *Git* son:

```
# apt install git git-all gitk gitg git-cola git-gui qgit tig lighttpd
vim-fugitive
```

También existen otros proyectos parecidos a Git, algunos de ellos son:

```
# apt install mercurial
# apt install subversion rapidsvn
# apt install cvs
```

Configuración de Git si se quiere especificar la identidad del que controla el repositorio local en el equipo, debemos usar (por omisión toma la información de la cuenta del usuario y máquina):

```
$ git config –global user.name "Antonio Carrillo"
$ git config –global user.email antoniocarrillo@ciencias.unam.mx
```

si se desea configurar el editor de texto a usar por *Git*, usamos (por omisión es *vim*):

```
$ git config –global Core.editor scite
```

si se desea configurar la herramienta de control de diferencias, usamos (por omisión *vimdiff*):

$ git config –global merge.tool meld

para comprobar podemos usar:

git config –global -list

el cual creará un archivo de texto llamado .gitconfig, y podemos ver su contenido, usando:

cat ~/.gitconfig

Clonar un repositorio Git podemos iniciar clonando algún repositorio remoto de la red, para ello debemos conocer su dirección y usar:

$ git clone <dirección>

esto permitirá bajar todo el contenido del proyecto a clonar.

Crear un repositorio Git local si lo que requiero es un control personal sin necesidad de compartir los archivos con ningún otro usuario, puedo usar *Git* de forma local en cualquier directorio mediante:

$ git init

Si se desea agregar la identidad del que controla el repositorio en este directorio, se debe usar:

$ git config user.name "Antonio Carrillo"
$ git config user.email antoniocarrillo@ciencias.unam.mx

Ahora para agregar los archivos (todos los de este directorio), usar:

$ git add .

así podemos hacer la confirmación de los cambios, mediante:

```
$ git commit -m "Primer lanzamiento"
```

ahora cada que lo requiera al hacer modificaciones, puedo checar los cambios:

```
$ git status
```

o en forma gráfica con *gitk*, mediante:

```
$ gitk
```

para actualizar los cambios, usar:

```
$ git commit -a -m 'Actualizacion'
```

en caso necesario, podemos mover uno o más archivos de una trayectoria a otra, usando:

```
$ git mv archivo(s) /trayectoria
```

o podemos borrar uno o más archivos, usando:

```
$ git rm archivo(s)
```

además podemos editar el archivo *.gitignore*, para indicar los archivos a ignorar usando líneas independientes para cada tipo.

Podemos ver el histórico de las operaciones que hemos hecho, usando:

```
$ git log
```

y podemos comprobar el estado del repositorio, usando:

```
$ git status
```

La otra alternativa es preparar un directorio para el repositorio ya sea en el servidor o de forma local, mediante:

```
$ mkdir example.git
$ cd example.git
```

Para inicializar el repositorio:

```
$ git –bare init
```

Es buena opción limitar el acceso a la cuenta vía *ssh*, por ello es mejor cambiar en */etc/passwd*, la línea del usuario predeterminada:

```
tlahuiz:x:1005:1005:Tlahuizcalpan„„:/home/tlahuiz:/bin/bash
```

a esta otra:

```
tlahuiz:x:1005:1005:Tlahuizcalpan„„:/home/tlahuiz:/usr/bin/git-
Shell
```

En la máquina de trabajo o en el servidor en cualquier carpeta se genera la estructura del repositorio en un directorio temporal de trabajo para el repositorio:

```
$ mkdir tmp
$ cd tmp
$ git init
```

Para generar la estructura de trabajo para el repositorio y los archivos necesarios:

```
$ mkdir branches release trunk
$ mkdir ...
```

Para adicionar todos y cada uno de los archivos y carpetas:

```
$ git add .
```

Para subir los cambios:

```
$ git commit -m "Texto"
```

Después debemos mandarlo al servidor:

```
$ git remote add origin ssh://usr@máquina/~/trayectoria
```

o mandarlo a un directorio local:

```
$ git remote add origin ~/trayectoria
$ git push origin +master:refs/heads/master
```

Para usar el repositorio en cualquier otra máquina hay que bajar el repositorio por primera vez del servidor:

```
$ git clone ssh://usr@máquina/~/trayectoria
```

o de una carpeta local:

```
$ git clone ~/trayectoria
```

Ahora, podemos configurar algunos datos usados en el control de cambios:

```
$ git config –global usr.name "Antonio Carrillo"
$ git config –global usr.email antoniocarrillo@ciencias.unam.mx
```

cuando se requiera actualizar del repositorio los cambios:

```
$ git pull
```

para subir los cambios al repositorio:

```
$ git commit -a -m "mensaje"
$ git push
```

Comando usados para el trabajo cotidiano en *Git* para ver el estado de los archivos locales, usamos:

```
$ git status
```

para generar una nueva rama y trabajar en ella:

```
$ git branch MiIdea
$ git checkout MiIdea
```

o en un solo paso:

```
$ git checkout -b MiIdea
```

Para unificar las ramas generadas en el punto anterior:

```
$ git checkout master
$ git merge MiIdea
```

Para borrar una rama:

```
$ git branch -d MiIdea
```

Para listar ramas:

```
$ git branch
```

Para listar ramas fusionadas:

```
$ git branch –merged
```

Para listar ramas sin fusionar:

```
$ branch –no-merged
```

Para ver los cambios en el repositorio:

```
$ git log
```

o verlos en forma acortada:

```
$ git log –pretty=oneline
```

Para recuperar un archivo de una actualización anterior:

```
$ git show a30ab2ca64d81876c939e16e9dac57c8db6fb103:ruta/al/archivo
> ruta/al/archivo.bak
```

Para volver a una versión anterior:

```
$ git reset –hard 56f8fb550282f8dfaa75cd204d22413fa6081a11:
```

para regresar a la versión presente (cuidado con subir cambios en ramas anteriores):

```
$ git pull
```

Si en algún momento borramos algo o realizamos cambios en nuestra máquina y necesitamos regresar los archivos como estaban en nuestra última actualización, podemos usar:

```
$ git reset –hard HEAD
```

este trabaja con la información de nuestra copia local y no necesita conexión de red para la restitución. Eventualmente es necesario optimizar la copia local de los archivos en *Git*, para ello podemos usar:

```
$ git gc
```

Visualizador gráfico para *Git*:

```
# apt install gitk
```

Git es un proyecto pujante, amplio y bien documentado, ejemplos y documentación puede ser consultada en:

- https://git-scm.com/book/es/v1
- http://git-scm.com/documentation
- https://coderwall.com/p/kucyaw/protect-secret-data-in-git-repo

Git en Google Drive:

- http://www.iexplain.org/using-git-with-google-drive-a-tutorial/
- https://techstreams.github.io/2016/09/07/google-drive-as-simple-git-Host/

Un Resumen de los Comando de Git más Usados

`CONFIGURACION`

Configurar Nombre que salen en los commits

git config –global user.name "antonio"

Configurar Email

git config –global user.email antonio@gmail.com

Marco de colores para los comando

git config –global color.ui true

`INICIANDO REPOSITORIO`

Iniciamos GIT en la carpeta donde está el proyecto

git init

Clonamos el repositorio de github o bitbucket

git clone <url>

Añadimos todos los archivos para el commit

git add .

Hacemos el primer commit

git commit -m "Texto que identifique por que se hizo el commit"

Subimos al repositorio

git push origin master

`GIT CLONE`

Clonamos el repositorio de github, gitlab o bitbucket

git clone <url>

Clonamos el repositorio de github, gitlab o bitbucket

git clone <url> git-demo

`GIT ADD`

Añadimos todos los archivos para el commit

git add .

Añadimos el archivo para el commit

git add <archivo>

Añadimos todos los archivos para el commit omitiendo los nuevos

git add –all

Añadimos todos los archivos con la extensión especificada

git add *.txt

Añadimos todos los archivos dentro de un directorio y de una extensión especifica

git add docs/*.txt

Añadimos todos los archivos dentro de un directorios

git add docs/

`GIT COMMIT`

Cargar en el HEAD los cambios realizados

git commit -m "Texto que identifique por que se hizo el commit"

Agregar y Cargar en el HEAD los cambios realizados

git commit -a -m "Texto que identifique por que se hizo el commit"

De haber conflictos los muestra

git commit -a

Agregar al último commit, este no se muestra como un nuevo commit en los logs. Se puede especificar un nuevo mensaje

git commit –amend -m "Texto que identifique por que se hizo el commit"

`GIT PUSH`

Subimos al repositorio

git push <origien> <branch>

Subimos un tag

git push –tags

`GIT LOG`

Muestra los logs de los commits

git log

Muestras los cambios en los commits

git log –oneline –stat

Muestra gráficos de los commits

git log –oneline –graph

`GIT DIFF`

Muestra los cambios realizados a un archivo

git diff
git diff –staged

`GIT HEAD`

Saca un archivo del commit

git reset HEAD <archivo>

Devuelve el último commit que se hizo y pone los cambios en staging

git reset –soft HEAD^

Devuelve el último commit y todos los cambios

git reset –hard HEAD^

Devuelve los 2 últimos commit y todos los cambios

git reset –hard HEAD^^

Rollback merge/commit

git log
git reset –hard <commit_sha>

`GIT REMOTE`

Agregar repositorio remoto

git remote add origin <url>

Cambiar de remote

git remote set-url origin <url>

Remover repositorio

git remote rm <name/origin>

Muestra lista repositorios

git remote -v

Muestra los branches remotos

git remote show origin

Limpiar todos los branches eliminados

git remote prune origin

`GIT BRANCH`

Crea un branch

git branch <nameBranch>

Lista los branches

git branch

Comando -d elimina el branch y lo une al master

git branch -d <nameBranch>

Elimina sin preguntar

git branch -D <nameBranch>

`GIT TAG`

Muestra una lista de todos los tags

git tag

Crea un nuevo tags

git tag -a <verison> - m "esta es la versión x"

`GIT REBASE`

Los rebase se usan cuando trabajamos con branches esto hace que los branches se pongan al día con el master sin afectar al mismo

Une el branch actual con el mastar, esto no se puede ver como un merge

git rebase

Cuando se produce un conflicto no das las siguientes opciones:

cuando resolvemos los conflictos –*continue* continua la secuencia del rebase donde se pauso

git rebase –continue

Omite el conflicto y sigue su camino

git rebase –skip

Devuelve todo al principio del rebase

git reabse –abort

Para hacer un rebase a un branch en específico

git rebase <nameBranch>

`OTROS COMANDOS`

Lista un estado actual del repositorio con lista de archivos modificados o agregados

git status

Quita del HEAD un archivo y le pone el estado de no trabajado

git checkout – <file>

Crea un branch en base a uno Online

git checkout -b newlocalbranchname origin/branch-name

Busca los cambios nuevos y actualiza el repositorio

git pull origin <nameBranch>

Cambiar de branch

git checkout <nameBranch/tagname>

Une el branch actual con el especificado

git merge <nameBranch>

Verifica cambios en el repositorio Online con el local

git fetch

Borrar un archivo del repositorio

git rm <archivo>

`Fork`

Descargar remote de un fork

git remote add upstream <url>

Merge con master de un fork

git fetch upstream
git merge upstream/master

Git-crypt El paquete *git-crypt* es una solución que usa *GPG* por debajo de *Git* que permite cifrado y descifrado transparente de archivos en un repositorio *git*.

Los archivos que se requieran proteger serán cifrados al hacer commit y descifrados al hacer *checkout* y permite compartir libremente un repo-sitorio que contenga contenido tanto público como privado. De esta forma, permite trabajar de manera transparente con el contenido descifrado, de forma que desarrolladores que no tengan la clave secreta podrán clonar y hacer commit en un repositorio con archivos cifrados.

Esto te permite almacenar tu material secreto (como pueden ser claves) en el mismo repositorio que tu código sin tener que bloquearlo. Solo un usuario autorizado puede dar permisos a otros usuarios.

Para instalar el paquete git-crypt usamos:

```
# apt install git-crypt
```

Ya instalado debemos preparar el repositorio git, para crear la llave, entonces usar:

```
$ git-crypt keygen ~/crypt-key
```

Ahora podemos crear el repositorio:

```
$ cd repo
$ git-crypt init
```

Especifica que carpetas/archivos deben ser cifrados, como git-filters:

```
$ cat .gitattributes
```

keys filter=git-crypt diff=git-crypt

crear la lista de los archivos a cifrar

```
$ vi .gitattributes
```

Indicamos que se cifren, por ejemplo, los archivos *.html, .org*, directorio:secretdir/**secreto y archivo, con cualquier extensión o palabra que le preceda.

```
*.html filter=git-crypt diff=git-crypt‘
*.org filter=git-crypt diff=git-crypt
directorio_secreto/** filter=git-crypt diff=git-crypt
*archivo* filter=git-crypt diff=git-crypt
```

ahora cada vez que hagamos un commit, los archivos *.html y .org*, subirán cifrados.

Ya podemos usar la llave para cifrar los archivos indicados por .gitattributes mediante:

```
$ git-crypt unlock ~/crypt-key
```

y agregar los archivos que deseamos cifrar, usando *git add*, revisando el estado de los archivos cifrados mediante:

```
$ git-crypt status -f
```

y podemos hacer los commits necesarios.

Al clonar el repositorio, los archivos cifrados se mostrarán como tal, hasta hacer en el repositorio:

```
$ git-crypt unlock ~/crypt-key
```

mostrando los archivos descifrados a partir de ese momento

Si se desea respaldar el repositorio en un solo archivo se puede usar:

```
$ git bundle create /tmp/Respaldo –all
```

y para restaurar usar algo como:

```
$ git clone /tmp/Respaldo newFolder
```

También podemos añadir usuarios autorizados (identificados por su clave *GPG*), mediante:

```
$ git-crypt add-gpg-user USER_ID
```

Flujos de trabajo comunes

- En la máquina del desarrollador: Crea el vault, añádete como usuario fiable. Pide las claves públicas a los miembros de tu equipo y añádelas al vault.
- En el entorno de Integración Continua (CI): Añade una clave *GPG* común para los ejecutores jenkins/CI. Autorízarla en el repositorio.

Seguridad

- Git-crypt usa *GPG* internamente, así que el nivel de seguridad debería ser el dado por *GPG*, a excepción de posibles errores en el propio programa *git-crypt*.

- Git-crypt es más seguro que otros sistemas git de cifrado transparente, *git-crypt* cifra archivos usando *AES-256* en modo *CTR* con un synthetic IV derivado del *SHA-1 HMAC* del archivo. Este modo de operar proporciona seguridad semántica ante *CPAs* (chosen-plain attacks) determinísticos. Esto significa que pese a que el cifrado es determinística (lo cual es requerido para que git pueda distinguir cuando un archivo ha cambiado y cuando no), no filtra información más allá de mostrar si dos archivos son idénticos o no.

Limitaciones y Trucos

- Cualquier usuario no autorizado puede ver que estamos usando *git-crypt* basándose en la evidencia dejada en el archivo .gitattributes.

- Git-crypt no cifra nombres de archivo, mensajes de *commit, symlink targets, gitlinks*, u otros metadatos.

- Git-crypt se apoya en git filters, los cuales no fueron diseñados con el cifrado en mente. Así pues, *git-crypt* no es la mejor herramienta para cifrar la mayoría o totalidad de los archivos de un repositorio. Donde *git-crypt* destaca es en aquellos casos en que la mayoría del repositorio es público pero unos pocos archivos deben ser cifrados (por ejemplo, claves privadas o archivos con credenciales API). Para cifrar un repositorio entero, es mejor considerar usar un sistema como **git-remote-gcrypt**.

- Git-crypt no esconde cuando un archivo cambia o no, cuanto ocupa o el hecho de que dos archivos sean idénticos.

- Los archivos cifrados con *git-crypt* no se pueden comprimir. Incluso el más pequeño de los cambios en un archivo cifrado requiere que git archive el archivo modificado en su totalidad y no solo un delta.

- A pesar de que *git-crypt* protege el contenido de los archivos individuales con *SHA-1 HMAC*, *git-crypt* no puede ser usado de forma segura a menos que el repositorio entero esté protegido contra la alteración de

datos (un atacante que pueda mutar tu repositorio podrá alterar tu archivo *.gitattributes* para deshabilitar el cifrado). Si fuera necesario, usa características de *git* cómo signed tags en vez de contar únicamente con *git-crypt* para la integridad.

- El *diff* del *commit* varía cuando el vault está abierto vs cuando está cerrado. Cuando está abierto, los contenidos del archivo están en formato plano, es decir, descifrados. En consecuencia puedes ver el *diff*. Cuando el vault está cerrado, no se puede apreciar un *diff* efectivo ya que el texto cifrado cambia, pero el ojo humano no puede distinguir los contenidos.

Además de Git usado de forma local, existen diversos servicios en la nube[134] que permiten dar soporte a proyectos mediante *Git,* en los cuales es necesario crear una cuenta y subir los datos usando *Git,* algunos de estos servicios son:

7.7.9 GitLab vs GitHub

Aunque GitHub y GitLab tienen similitudes, incluso en el propio nombre que comienza por Git debido a que ambas se basa en la famosa herramienta de control de versiones escrita por Linus Torvalds, pero ni una ni otra son exactamente iguales. Por ello, el ganador de la batalla GitHub vs GitLab no está tan claro, tienen algunas diferencias que hacen que tengan sus ventajas y desventajas para los usuarios y desarrolladores que las suelen usar.

¿Qué es GitHub? es una plataforma de desarrollo colaborativo, también llamado forja. Es decir, una plataforma enfocada hacia la cooperación entre desarrolladores para la difusión y soporte de su software (aunque poco a poco se ha ido usando para otros proyectos más allá del Software).

Como su propio nombre indica, se apoya sobre el sistema de control de versiones *Git*. Así, se puede operar sobre el código fuente de los programas y llevar un desarrollo ordenado. Además, esta plataforma está escrita en Ruby on Rails.

[134] Algunos de estos proyectos gratuitos son: Gitlab, Github, Bitbucket, Beanstalk, Launchpad, SourceForge, Phabricator, GitBucket, Gogs, Gitea, Apache Allura, entre otros.

Tiene una enorme cantidad de proyectos de código abierto almacenada en su plataforma y accesibles de forma pública. Tal es su valor que Microsoft optó por comprar esta plataforma en 2018, aportando una cifra de nada menos que de 7,500 millones de dólares.

Pese a las dudas sobre esa compra, la plataforma continuó operando como de costumbre, y continúa siendo una de las más usadas. En ella se alojan proyectos tan importantes como el propio Kernel Linux.

¿Qué es GitLab? es otra alternativa a GitHub, otro sitio de forja con un servicio Web y un sistema de control de versiones también basado en Git. Por supuesto, se ideó para el alojamiento de proyectos de código abierto y para facilitar la vida de los desarrolladores, pero existen algunas diferencias con el anterior.

Esta Web, además de la gestión de repositorios y control de versiones, también ofrece alojamiento para Wikis, y sistema de seguimiento de errores. Una completa suite para crear y gestionar los proyectos de todo tipo, ya que, al igual que GitHub, actualmente se alojan proyectos que van más allá del código fuente.

Fue escrito por unos desarrolladores ucranianos, Dmitry Zaporozhets y Valery Sizov, usando lenguaje de programación Ruby y algunas partes en Go. Después se mejoró su arquitectura con Go, Vue.js, y Ruby on Rails, como en el caso de GitHub.

A pesar de ser muy conocida y ser la gran alternativa a GitHub, no cuenta con tantos proyectos. Eso no quiere decir que la cantidad de código alojado sea muy grande, con organizaciones que confían en ella de la talla del CERN, la NASA, IBM, Sony, etc.

Personalmente, te diría que no existe un claro ganador en la batalla GitHub vs GitLab. No es tan sencillo elegir una plataforma que sea infinitamente superior a la otra, de hecho, cada una tiene sus puntos fuertes y sus puntos débiles. Y todo dependerá de lo que realmente busques para que te tengas que decantar por una u otra.

Diferencias GitHub vs GitLab A pesar de todas las similitudes, una de las claves a la hora de decantarse en la comparativa GitHub vs GitLab pueden ser las diferencias entre ambas:

Niveles de autenticación: GitLab puede establecer y modificar permisos a los diferentes colaboradores según su función. En el caso de GitHub, puede

decidir quién tiene derechos de lectura y escritura en un repositorio, pero es más limitado en ese sentido.

- Alojamiento: aunque ambas plataformas permiten alojar el contenido de los proyectos en las propias plataformas, en el caso de GitLab también te puede permitir autoalojar tus repositorios, lo que puede ser una ventaja en algunos casos. GitHub ha agregado esa característica también, pero solo con ciertos planes de pago. GitHub ofrece máximo 3 colaboradores gratis y hasta 500 MB por repositorio, en cambio en GitLab no hay límite al número de colaboradores y hasta 10 GB por repositorio.

- Importación y exportación: GitLab contiene información muy detallada de cómo poder importar proyectos para moverlos de una plataforma a otra, como GitHub, Bitbucket, o traerlos a GitLab. Además, a la hora de exportar, GitLab ofrece un trabajo muy sólido. En el caso de GitHub no se ofrece documentación detallada, aunque se puede usar GitHub Importer como herramienta, aunque puede ser algo más restrictivo cuando se trata de exportar.

- Comunidad: ambos tienen una buena comunidad tras de sí, aunque GitHub parece haber ganado la batalla en popularidad. Actualmente aglutina millones de desarrolladores. Por eso, será más sencillo encontrar ayuda al respecto.

- Versiones Enterprise: ambos las ofrecen si pagas la cuota, por lo que se podría pensar que la comparativa GitHub vs GitLab no tiene sentido en este punto, pero lo cierto es que GitLab ofrece unas prestaciones muy interesantes, y se ha hecho popular entre los equipos de desarrollo muy grandes.

Ventajas y Desventajas de GitLab una vez conocidas las diferencias y semejanzas entre GitHub vs GitLab, las ventajas y desventajas de estas plataformas te pueden ayudar a decidirte.

Ventajas

- Plan gratuito y sin limitaciones, aunque tiene planes de pago.

- Es de licencia de código abierto.

- Permite el autohospedaje en cualquier plan.
- Está muy bien integrado con Git.

Desventajas

- Su interfaz puede ser algo más lenta con respecto a la competencia.
- Existen algunos problemas habituales con los repositorios.

Ventajas y Desventajas de GitHub Por otro lado, GitHub también tiene sus pros y contras, entre los que destacan los siguientes:

Ventajas

- Servicio gratuito, aunque también tiene servicios de pago.
- Búsqueda muy rápida en las estructura de los repos.
- Amplia comunidad y fácil encontrar ayuda.
- Ofrece prácticas herramientas de cooperación y buena integración con Git.
- Fácil integración con otros servicios de terceros.
- Trabaja también con TFS, HG y SVN.

Desventajas

- No es absolutamente abierto.
- Tiene limitaciones de espacio, ya que no puedes exceder de 100MB en un solo archivo, mientras que los repositorios están limitados a 1GB en la versión gratis.

Como ves, no existe un claro ganador. La elección no es fácil y, como comenté, deberías vigilar muy bien las ventajas, desventajas y diferencias de cada uno para poder identificar cuál se adapta mejor a tus necesidades.

Personalmente te diría que si quieres disponer de un entorno totalmente abierto, mejor usa GitLab. En cambio, si prefieres más facilidades y usar el servicio Web con más presencia, entonces ve a por GitHub. Incluso incluiría un tercero en discordia y te diría que si buscas trabajar con servicios Atlassian deberías mirar del lado de Bitbucket.

Uso GitLab (https://about.gitlab.com/)

Para configurar:

```
git config –global user.name "Antonio Carrillo Ledesma"
git config –global user.email "antoniocarrillo@ciencias.unam.mx"
```

Para crear nuevo repositorio:

```
git clone https://gitlab.com/antoniocarrillo69/MDF.git
cd MDF
touch README.md
git add README.md
git commit -m "add README"
git push -u origin master
```

Para usar una carpeta existente:

```
cd existing_folder
git init
git remote add origin https://gitlab.com/antoniocarrillo69/MDF.git
git add .
git commit -m "Initial commit"
git push -u origin master
```

Para usar un repositorio existente:

```
cd existing_repo
git remote rename origin old-origin
git remote add origin https://gitlab.com/antoniocarrillo69/MDF.git
git push -u origin –all
git push -u origin –tags
```

Uso Github (https://github.com/)

Para configurar:

```
git config –global user.name "Antonio Carrillo Ledesma"
git config –global user.email "antoniocarrillo@ciencias.unam.mx"
```

Para configurar un nuevo repositorio:

```
$ touch README.md
$ git init
$ git add .
$ git commit -m "mi primer commit"
$ git remote add origin https://github.com/antoniocarrillo69/ejemploPruebas.git
$ git pull origin master
$ git push origin master
```

7.7.10 Otras opciones

Herramientas para convertir código fuentes en HTML, usando:

```
$ code2html Fuente Salida.html
```

o

```
$ c2html Fuente
```

Para java, usamos:

```
$ java2html Fuentes
```

También podemos convertir código fuente en PDF, usando:

```
$ nl test.cpp | a2ps -1 -l100 -otest.ps ; ps2pdf test.ps
```

el primer comando numera las líneas del fuente, el segundo comando generá del fuente numerado un .PS y el último comando convierte .PS a .PDF

Si se tiene que ejecutar múltiples programas que son independientes uno de otro se puede usar el programa *parallel* para correr N (número de cores del equipo) de ellos al mismo tiempo, por ejemplo si tenemos un archivo Bash con el nombre *mi-Bash* y cuyo contenido es:

```
./a.out 4 5 4 > a1.txt
./a.out 4 5 3 > a2.txt
./a.out 4 5 6 > a3.txt
./a.out 4 5 4 > a4.txt
./a.out 3 5 4 > a5.txt
./a.out 4 6 4 > a6.txt
```

entonces podemos ejecutarlo usando *parallel*, el programa usará el número máximo de cores disponibles:

```
$ parallel -v < mi-Bash
```

si solo se desea usar una determinada cantidad de cores (por ejemplo 3) entonces usamos:

```
$ parallel -v -j 3 < mi-Bash
```

7.8 Programando Desde la Nube

Existen diferentes servicios Web[135] que permiten editar, compilar y ejecutar código de diversos lenguajes y paquetes desde el navegador, esto en aras de que los estudiantes y profesores que cuenten con algún sistema de acceso a red y un navegador puedan programar en los más diversos lenguajes, IDEs y terminales sin hacer instalación alguna en su equipo de cómputo, tableta o teléfono celular.

Algunos ejemplos de estos servicios son:

- https://www.jdoodle.com/
- https://repl.it/
- http://browxy.com
- https://jupyter.org/try
- https://tio.run/
- https://www.compilejava.net/
- http://codepad.org/
- https://code.hackerearth.com/
- https://www.remoteinterview.io/online-c-compiler
- https://ideone.com/
- https://hackide.herokuapp.com/
- https://www.codechef.com/ide
- http://cpp.sh/

[135] Cuando se trabaja desde la Web es recomendable usar el modo Privado o Incógnito para no guardar el historial de navegación, información introducida en los formularios y borrar al cerrar el navegador los datos de los sitios visitados. Pero recuerda que los sitios Web que visitamos sí guardan información de nuestra visita, nuestro proveedor de Internet también guarda constancia de nuestra visita y si descargamos algo, esto no se borra al igual que el historial de descargas, además de las marcas de páginas o favoritos se conservarán al cerrar el navegador.

- https://codebunk.com/
- https://rextester.com/
- https://www.tutorialspoint.com/codingground.htm
- https://www.compileonline.com
- http://pythonfiddle.com/
- https://trinket.io/python
- https://www.pythonanywhere.com/try-ipython/
- https://www.rollapp.com/
- https://godbolt.org/
- https://www.codiva.io/
- https://paiza.io/en
- https://wandbox.org/
- http://coliru.stacked-crooked.com/
- http://quick-bench.com/
- https://cppinsights.io/
- https://ideone.com/
- http://cpp.sh/
- https://ide.geeksforgeeks.org/
- https://www.codechef.com/ide
- https://visualstudio.microsoft.com/services/visual-studio-online/

`Usando Editores Colaborativos`

La escritura colaborativa es una escritura de códigos de programación en la Web hecha por más de una persona simultáneamente.

Algunos ejemplos de estos servicios son:

- http://collabedit.com (edita código, tiene chat, no compila)
- https://gitduck.com/
- https://codeshare.io/
- https://www.tutorialspoint.com/codingground.htm
- http://ideone.com
- https://codebunk.com
- https://visualstudio.microsoft.com/services/visual-studio-online/
- https://ace.c9.io/build/kitchen-sink.html
- https://coderpad.io/
- https://peerpad.net/
- https://aws.amazon.com/cloud9/
- https://codeanywhere.com/
- https://stekpad.com/home/

Algunas de las terminales soportados son:

CentOS, IPython, Lua, MemCached, Mongo DB, MySQL, Node.js, Numpy, Oracle, Octave, PowerShell, PHP, R Programming, Redis, Ruby, SciPy, SymPy, etc.

Algunos de los IDEs soportados son:

Ada (GNAT), Algol68, Angular JS, Assembly, AsciiDoc, AWK, Bash Shell, Befunge, Bootstrap, Brainf**k, C, CSS3, Chipmunk BASIC, Clojure, Cobol, CoffeeScript, ColdFusion, C99 Strict, C++, C++ 0x, C++ 11, C#, Dart, D Programming Language, Embedded C, Erlang, Elixir, Factor, Fantom, Falcon, Fortran-95, Forth,F#, Free Basic, Groovy, GO, Haxe, Haskell, HTML, ilasm, Intercal, Icon, Java, Java 8, Java MySQL, JavaScript, JSP, JQuery, Julia, Korn Shell (ksh), Latex, Lisp, LOLCODE, Lua, Matlab/Octave, Malbolge, Markdown, MathML, Mozart-Oz, Nimrod, Node.JS, Objective-C, OCaml, Pascal, PARI/GP, Pawn, Perl, Perl MySQL, PHP, PHP MySQL, WebView, Pike, Processing.js, p5.js, Prolog, Python-2, Python-3, Python MySQL, Jupyter Notebook, Rexx, reStructure, Ruby, Rust, Scala, R Programming, Scheme, Smalltalk,SML/NJ, Simula, SQLite SQL, Tcl, TeX, Unlambda, VB.NET, Verilog, Whitespace, Ya Basic, etc.

Google Colaboratory Integrante de la G Suite for Education de Google permite a los usuarios que pertenezcan a esta Suite (como gran parte de los estudiantes de la UNAM) tener acceso desde el navegador para escribir y ejecutar código de Python (Jupyter), es posible elegir correr nuestro Notebook en una CPU, GPU o en una TPU de forma gratuita. Tiene algunas restricciones, como por ejemplo que una sesión dura 12 hrs, pasado ese tiempo se limpia nuestro ambiente y perdemos las variables y archivos que tengamos almacenados allí.

Es conveniente para principiantes que requieran experimentar con Machine Learning y Deep Learning pero sin recurrir en costos de procesamiento Cloud. Además el ambiente de trabajo ya viene con muchas librerías instaladas y listas para utilizar (como por ejemplo *Tensorflow, Sclkit-learn, Pytorch, Keras* y *OpenCV*), ahorrándonos el trabajo de configurar nuestro ambiente de trabajo. Podemos importar nuestros archivos y datos desde *Google Drive, GitHub*, etc.

Más información sobre Google Colaboratory en:

https://colab.research.google.com/notebooks/intro.ipynb

8 Bases de Datos

¿Qué son las bases de datos? una base de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su uso posterior (véase [64] y [65]). En este sentido; una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta. Actualmente la mayoría de las bases de datos están en formato digital, se han desarrollado y se ofrece un amplio rango de soluciones al problema del almacenamiento de datos.

8.1 Clasificación

Las bases de datos se pueden clasificar de diferentes formas: acuerdo a la ubicación, de acuerdo a la forma de establecer relaciones entre los datos, según la orientación, entre otros.

De acuerdo a la ubicación

- Base de datos centralizada: Se ubica, almacena y mantiene en una sola ubicación. Eso no significa que el usuario tenga que estar en el mismo lugar para acceder.
- Base de datos distribuida: Se trata en realidad de diferentes bases de datos en distintas locaciones físicas unidas por un gestor que las hace funcionar como si fueran una sola.

De acuerdo a la forma de establecer relaciones entre los datos

- Relacionales: En este tipo de base de datos estos se organizan en forma de filas y columnas.
- Orientadas a objetos: Aquí los datos se almacenan en forma de objetos. Estos objetos se guardan en la base de datos asignándoles atributos y métodos que son los que definen qué hacer con los datos.
- Orientadas a grafos: Utiliza la teoría de grafos para almacenar, mapear y consultar las relaciones entre datos.
- NoSQL: Permite almacenar datos no estructurados o semiestructurados.

- Orientada a documentos: Es un subtipo de la anterior. En lugar de almacenar los datos en filas y columnas utiliza documentos para almacenar y recuperar los datos. Estos documentos organizan los datos utilizando estándares como JSON o XML.

Según la orientación

- OLTP: Son bases de datos orientadas al procesamiento de transacciones e incluye funciones de introducción, modificación y borrado de datos.
- OLAP: Estas bases de datos están orientadas al análisis de los datos para permitir extraer conclusiones.

Otros tipos

- Autónomas: Están basadas en la nube y utilizan el aprendizaje automático para automatizar el trabajo la base de datos, la seguridad, las copias de seguridad, las actualizaciones y otras tareas de gestión rutinarias que en las bases de datos tradicionales realiza un administrador.
- Almacén de datos: Es una base de datos enfocada en el sector corporativo que integra y depura información de varias fuentes distintas para procesarla y analizarla desde diferentes puntos de vista a gran velocidad.

Hay programas denominados sistemas gestores de bases de datos SGBD (Database Management System o DBMS), que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada. Actualmente se tienen dos importantes tipos de bases de datos:

- Structured Query Language SQL (lenguaje de consulta estructurada) es un lenguaje de dominio específico, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales.
- NoSQL "no solo SQL" es una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico de SGBDR (Sistema de Gestión de Bases de Datos Relacionales) en aspectos importantes, siendo el más destacado que no usan SQL como lenguaje principal de consultas. Los datos almacenados no requieren estructuras fijas como tablas.

8.2 ¿Qué son las bases de datos SQL?

el lenguaje de consulta estructurada SQL (Structured Query Language), es un lenguaje de dominio específico, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales. Una de sus principales características es el manejo del álgebra y el cálculo relacional para efectuar consultas con el fin de recuperar, de forma sencilla, información de bases de datos, así como realizar cambios en ellas.

Originalmente basado en el álgebra relacional y en el cálculo relacional, SQL consiste en un lenguaje de definición de datos, un lenguaje de manipulación de datos y un lenguaje de control de datos. El alcance de SQL incluye la inserción de datos, consultas, actualizaciones y borrado, la creación y modificación de esquemas y el control de acceso a los datos. También en SQL a veces se describe como un lenguaje declarativo, también incluye elementos procesales.

Características generales de SQL SQL es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales y permite así gran variedad de operaciones.

Es un lenguaje declarativo de "alto nivel" o "de no procedimiento" que, gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros –y no a registros individuales– permite una alta productividad en codificación y la orientación a objetos. De esta forma, una sola sentencia puede equivaler a uno o más programas que se utilizarían en un lenguaje de bajo nivel orientado a registros. SQL también tiene las siguientes características:

- Lenguaje de definición de datos: El LDD de SQL proporciona comandos para la definición de esquemas de relación, borrado de relaciones y modificaciones de los esquemas de relación.
- Lenguaje interactivo de manipulación de datos: El LMD de SQL incluye lenguajes de consultas basado tanto en álgebra relacional como en cálculo relacional de tuplas.
- Integridad: El LDD de SQL incluye comandos para especificar las restricciones de integridad que deben cumplir los datos almacenados en la base de datos.

- Definición de vistas: El LDD incluye comandos para definir las vistas.
- Control de transacciones: SQL tiene comandos para especificar el comienzo y el final de una transacción.
- SQL incorporado y dinámico: Esto quiere decir que se pueden incorporar instrucciones de SQL en lenguajes de programación como: C++, C, Java, Python, PHP, COBOL, Pascal y Fortran.
- Autorización: El LDD incluye comandos para especificar los derechos de acceso a las relaciones y a las vistas.

Tipos de datos Algunos de los tipos de datos básicos de SQL son:

- BINARY (1 Byte), para consultas sobre tabla adjunta de productos de bases de datos que definen un tipo de datos Binario.
- BIT (1 Byte), valores Si/No ó True/False
- BYTE (1 Byte), un valor entero entre 0 y 255.
- COUNTER (4 Bytes), un número incrementado automáticamente (de tipo Long)
- CURRENCY (8 Bytes), un entero escalable entre -922,337,203,685,477.5808 y 922,337,203,685,477.5807.
- DATETIME (8 Bytes), un valor de fecha u hora entre los años 100 y 9999.
- SINGLE (4 Bytes), un valor en punto flotante de precisión simple con un rango de $-3.402823*10^{38}$ a $-1.401298*10^{-45}$ para valores negativos, $1.401298 * 10^{-45}$ a $3.402823 * 10^{38}$ para valores positivos, y 0.
- DOUBLE (8 Bytes), un valor en punto flotante de doble precisión con un rango de $-1.79769313486232 * 10^{308}$ a $-4.94065645841247 * 10^{-324}$ para valores negativos, $4.94065645841247 * 10^{-324}$ a $1.79769313486232 * 10^{308}$ para valores positivos, y 0.
- SHORT (2 Bytes), un entero corto entre -32,768 y 32,767.
- LONG (4 Bytes), un entero largo entre -2,147,483,648 y 2,147,483,647.

- LONGTEXT (1 Byte por carácter), de cero a un máximo de 1.2 gigabytes.
- LONGBINARY (según se necesite), de cero 1 gigabyte. Utilizado para objetos OLE.
- TEXT (1 Byte por carácter), de cero a 255 caracteres.

También se tienen definidos múltiples sinónimos de los tipos de datos definidos:

- BINARY: VARBINARY
- BIT: BOOLEAN, LOGICAL, LOGICAL1, YESNO
- BYTE: INTEGER1
- COUNTER: AUTOINCREMENT
- CURRENCY: MONEY
- DATETIME: DATE, TIME, TIMESTAMP
- SINGLE: FLOAT4, IEEESINGLE, REAL
- DOUBLE: FLOAT, FLOAT8, IEEEDOUBLE, NUMBER, NUMERIC
- SHORT: INTEGER2, SMALLINT
- LONG: INT, INTEGER, INTEGER4
- LONGBINARY: GENERAL, OLEOBJECT
- LONGTEXT: LONGCHAR, MEMO, NOTE
- TEXT: ALPHANUMERIC, CHAR - CHARACTER, STRING - VARCHAR

SQL no requiere que escribamos todas las palabras clave en mayúscula, pero por convención ayuda a las personas a distinguir las palabras clave de SQL de los nombres de las columnas y las tablas.

Optimización Como ya se dijo antes, y suele ser común en los lenguajes de acceso a bases de datos de alto nivel, SQL es un lenguaje declarativo. O sea, que especifica qué es lo que se quiere y no cómo conseguirlo, por lo que una sentencia no establece explícitamente un orden de ejecución.

El orden de ejecución interno de una sentencia puede afectar seriamente a la eficiencia del SGBD, por lo que se hace necesario que este lleve a cabo una optimización antes de su ejecución. Muchas veces, el uso de índices acelera una instrucción de consulta, pero ralentiza la actualización de los datos. Dependiendo del uso de la aplicación, se priorizará el acceso indexado o una rápida actualización de la información. La optimización difiere sensiblemente en cada motor de base de datos y depende de muchos factores.

Los sistemas de bases de datos modernos poseen un componente llamado optimizador de consultas. Este realiza un análisis detallado de los posibles planes de ejecución de una consulta SQL y elige aquel que sea más eficiente para llevar adelante la misma.

Existe una ampliación de SQL conocida como FSQL (Fuzzy SQL, SQL difuso) que permite el acceso a bases de datos difusas, usando la lógica difusa.

8.2.1 Lenguaje de definición de datos (DDL)

El lenguaje de definición de datos (en inglés Data Definition Language, o DDL), es el que se encarga de la modificación de la estructura de los objetos de la base de datos. Incluye órdenes para modificar, borrar o definir las tablas en las que se almacenan los datos de la base de datos. Existen cuatro operaciones básicas: CREATE, ALTER, DROP y TRUNCATE.

CREATE (Crear) este comando permite crear objetos de datos, como nuevas bases de datos, tablas, vistas y procedimientos almacenados.

Ejemplos (crear una tabla):

```
CREATE TABLE clientes;
CREATE TABLE personas(
    IDPersonas INT NOT NULL,
    Nombre VARCHAR(50) NOT NULL,
    Apellido VARCHAR(60) NOT NULL,
UNIQUE(IDPersonas)
);
CREATE TABLE pedidos(
```

```
        OrdenID INT NOT NULL,
        NumeroOrden INT NOT NULL,
        IDPersona INT,
        PRIMARY KEY(OrdenID),
        FOREING KEY(IDPersona) REFERENCES personas(IDPersonas)
);
```

ALTER (Alterar) este comando permite modificar la estructura de una tabla u objeto. Se pueden agregar/quitar campos a una tabla, modificar el tipo de un campo, agregar/quitar índices a una tabla, modificar un *trigger*, etc.

Ejemplo (agregar columna a una tabla):

```
ALTER TABLE personas ADD edad INT UNSIGNED;
```

DROP (Eliminar) este comando elimina un objeto de la base de datos. Puede ser una tabla, vista, índice, trigger, función, procedimiento o cualquier objeto que el motor de la base de datos soporte. Se puede combinar con la sentencia ALTER.

Ejemplo (elimina la tabla personas):

```
DROP TABLE personas;
```

TRUNCATE (Truncar) este comando solo aplica a tablas y su función es borrar el contenido completo de la tabla especificada. La ventaja sobre el comando DELETE, es que si se quiere borrar todo el contenido de la tabla, es mucho más rápido, especialmente si la tabla es muy grande. La desventaja es que TRUNCATE solo sirve cuando se quiere eliminar absolutamente todos los registros, ya que no se permite la cláusula WHERE. Si bien, en un principio, esta sentencia parecería ser DML (Lenguaje de Mani-pulación de Datos), es en realidad una DDL, ya que internamente, el comando TRUNCATE borra la tabla y la vuelve a crear y no ejecuta ninguna transacción.

Ejemplo (borra el contenido de la tabla personas):

```
TRUNCATE TABLE personas;
```

CREATE INDEX permite crear índices (ya sea únicos o permite valores duplicados) en una o más columnas.

Ejemplo (crea un índice admitiendo valores duplicados en la columna)

```
CREATE INDEX indicePersonas ON personas(personas);
```

Ejemplo (crear un índice sin que puedan existir valores duplicados)

```
CREATE UNIQUE INDEX indicePersonas ON personas(personas);
```

8.2.2 Lenguaje de manipulación de datos DML

Un lenguaje de manipulación de datos (Data Manipulation Language, o DML en inglés) es un lenguaje proporcionado por el sistema de gestión de base de datos que permite a los usuarios llevar a cabo las tareas de consulta o manipulación de los datos, organizados por el modelo de datos adecuado.

El lenguaje de manipulación de datos más popular hoy en día es SQL, es usado para recuperar y manipular datos en una base de datos relacional.

SELECT (Seleccionar) La sentencia SELECT nos permite consultar los datos almacenados en una tabla de la base de datos.

Forma básica:

```
SELECT [{ALL|DISTINCT}]
<nombre_campo>[, <nombre_campo>...]
FROM {<nombre_tabla>|<nombre_vista>}[,
{<nombre_tabla>|<nombre_vista>}...]
[WHERE <condición> [{AND|OR} <condición>...]]
[GROUP BY <nombre_campo>[, <nombre_campo>...]]
[HAVING <condición> [{AND|OR} <condición>...]]
[ORDER BY {<nombre_campo>|<indice_campo>}
[{ASC|DESC}][,{<nombre_campo>|<indice_campo>}
[{ASC|DESC}]]];
```

SELECT palabra clave que indica que la sentencia de SQL que queremos ejecutar es de selección.

ALL indica que queremos seleccionar todos los valores. Es el valor por defecto y no suele especificarse casi nunca.

DISTINCT indica que queremos seleccionar solo los valores distintos.

FROM indica la tabla (o tablas) desde la que queremos recuperar los datos. En el caso de que exista más de una tabla se denomina a la consulta "consulta combinada" o "join". En las consultas combinadas es necesario aplicar una condición de combinación a través de una cláusula WHERE.

WHERE especifica una condición que debe cumplirse para que los datos sean devueltos por la consulta. Admite los operadores lógicos AND y OR.

GROUP BY especifica la agrupación que se da a los datos. Se usa siempre en combinación con funciones agregadas.

HAVING especifica una condición que debe cumplirse para que los datos sean devueltos por la consulta. Su funcionamiento es similar al de WHERE pero aplicado al conjunto de resultados devueltos por la consulta. Debe aplicarse siempre junto a GROUP BY y la condición debe estar referida a los campos contenidos en ella.

ORDER BY presenta el resultado ordenado por las columnas indicadas. El orden puede expresarse con ASC (orden ascendente) y DESC (orden descendente). El valor predeterminado es ASC.

Ejemplo:

Para formular una consulta a la tabla coches, usamos la palabra clave FROM y con SELECT le indicamos que deseamos recuperar los campos *matrícula, marca, modelo, color, número_kilómetros, num_plazas*, para ello debemos ejecutar la siguiente consulta:

```
SELECT
    matricula, marca, modelo, color, numero_kilometros, num_plazas
FROM
    coches
ORDER BY
    marca, modelo;
```

los datos serán devueltos ordenados por marca y por modelo en orden ascendente, de menor a mayor.

Ejemplo de consulta simplificada a través de un comodín de campos (*): El uso del asterisco indica que queremos que la consulta devuelva todos los campos que existen en la tabla y los datos serán devueltos ordenados por marca y por modelo.

```
SELECT *
FROM
    coches
ORDER BY
    marca, modelo;
```

Ejemplo de consulta simplificada a través de un comodín de campos (*): El uso del asterisco indica que queremos que la consulta devuelva todos los campos que existen en la tabla y los datos serán devueltos ordenados por marca en forma descendente y por modelo en forma ascendente.

```
SELECT *
FROM
    coches
ORDER BY
    marca DESC, modelo ASC;
```

Cláusula WHERE (Donde) la cláusula WHERE es la instrucción que nos permite filtrar el resultado de una sentencia SELECT. Habitualmente no deseamos obtener toda la información existente en la tabla, sino que queremos obtener sólo la información que nos resulte útil en ese momento. La cláusula WHERE filtra los datos antes de ser devueltos por la consulta. Cuando en la cláusula WHERE queremos incluir un tipo texto, debemos incluir el valor entre comillas simples.

Ejemplos:

En nuestro ejemplo, se desea consultar un coche en concreto, para esto se agregó una cláusula WHERE. Esta cláusula especifica una o varias condiciones que deben cumplirse para que la sentencia SELECT devuelva los datos. En este caso la consulta devolverá sólo los datos del coche con matrícula para que la consulta devuelva sólo los datos del coche con matrícula MF-234-ZD o bien la matrícula FK-938-ZL . Se puede utilizar la cláusula WHERE solamente, ó en combinación con tantas condiciones como queramos.

```
SELECT
    matricula, marca, modelo, color, numero_kilometros, num_plazas
FROM
    coches
WHERE
matricula = 'MF-234-ZD' OR matricula = 'FK-938-ZL';
```

Una condición WHERE puede ser negada a través del Operador Lógico NOT. La siguiente consulta devolverá todos los datos de la tabla coches, menos el que tenga la matrícula MF-234-ZD.

```
SELECT
    matricula, marca, modelo, color, numero_kilometros, num_plazas
FROM
    coches
WHERE
    NOT matricula = 'MF-234-ZD';
```

La siguiente consulta utiliza la condicional DISTINCT, la cual nos devolverá todos los valores distintos formados por los campos *marca* y *modelo* de la tabla coches.

```
SELECT DISTINCT marca, modelo FROM coches;
```

Cláusula ORDER BY (Ordernar Por) la cláusula ORDER BY es la instrucción que nos permite especificar el orden en el que serán devueltos los datos. Podemos especificar el orden de forma ascendente o descendente a través de las palabras clave ASC y DESC. El orden depende del tipo de datos que estén definidos en la columna, de forma que un campo numérico será ordenado como tal, y un alfanumérico se ordenará de la A a la Z, aunque su contenido sea numérico. El valor predeterminado es ASC si no se especifica al hacer la consulta.

Ejemplos:

```
SELECT
    matricula, marca, modelo, color, numero_kilometros, num_plazas
FROM
    coches
ORDER BY
    marca ASC, modelo DESC;
```

Este ejemplo, selecciona todos los campos *matricula, marca, modelo, color, numero_kilometros y num_plazas* de la tabla coches, ordenándolos por los campos *marca* y *modelo*, *marca* en forma ascendente y *modelo* en forma descendente.

```
SELECT
    matricula, marca, modelo, color, numero_kilometros, num_plazas
FROM
    coches
ORDER BY 2;
```

Este ejemplo, selecciona todos los campos *matrícula, marca, modelo, color, numero_kilometros y num_plazas* de la tabla coches, ordenándolos por el campo *marca*, ya que aparece en segundo lugar dentro de la lista de campos que componen la cláusula SELECT.

Subconsultas una subconsulta es una sentencia SELECT que está embebida en una cláusula de otra sentencia SQL. También pueden utilizarse subconsultas en los comandos INSERT, UPDATE, DELETE y en la cláusula FROM.

Las subconsultas pueden resultar útiles si necesitas seleccionar filas de una tabla con una condición que depende de los datos de la propia tabla o de otra tabla.

La subconsulta (consulta interna), se ejecuta antes de la consulta principal; el resultado de la subconsulta es utilizado por la consulta principal (consulta externa).

```
SELECT
    c.matricula, c.modelo
FROM
    coches AS c
WHERE c.matricula IN
(
    SELECT m.matricula
    FROM multas AS m
    WHERE m.importe > 100
);
```

En este ejemplo, se seleccionan las matrículas y los modelos de los coches cuyas multas superan los USD 100.

INSERT (Insertar) una sentencia INSERT de SQL agrega uno o más registros a una (y solo una) tabla en una base de datos relacional.

Forma básica:

```
INSERT INTO
tablatura(columnaA, [columnaB, ...])
VALUES
('valor1', ['valor2', ...]);
```

o también se puede utilizar como:

```
INSERT INTO tablatura VALUES ('valor1', 'valor2');
```

Las cantidades de columnas y valores deben ser iguales. Si una columna no se especifica, le será asignado el valor por omisión. Los valores especificados (o implícitos) por la sentencia INSERT deberán satisfacer todas las restricciones aplicables. Si ocurre un error de sintaxis o si alguna de las restricciones es violada, no se agrega la fila y se devuelve un error.

Ejemplo:

```
INSERT INTO agenda_telefonica (nombre, numero)
VALUES ('Roberto Jeldrez', 4886850);
```

Cuando se especifican todos los valores de una tabla, se puede utilizar la sentencia acortada:

```
INSERT INTO nombre_tabla VALUES ('valor1', ['valor2',
...]);
```

Ejemplo (asumiendo que 'nombre' y 'número' son las únicas columnas de la tabla 'agenda_telefonica'):

```
INSERT INTO agenda_telefonica
VALUES ('Jhonny Aguilar', 080473968);
```

Formas avanzadas una característica de SQL (desde SQL-92) es el uso de constructores de filas para insertar múltiples filas a la vez, con una sola sentencia SQL:

```
INSERT INTO
tabla(columna1[, columna2, ...])
VALUES
('valor1A', ['valor1B', ...]),
('value2A', ['value2B', ...]), ...;
```

Esta característica es soportada por DB2, PostgreSQL (desde la versión 8.2), MySQL, y H2.

Ejemplo (asumiendo que nombre y número son las únicas columnas en la tabla agenda_telefonica):

```
INSERT INTO
    agenda_telefonica
VALUES
    ('Roberto Fernández', '4886850'),
    ('Alejandro Sosa', '4556550');
```

Que podía haber sido realizado por las sentencias:

```
INSERT INTO agenda_telefonica VALUES ('Roberto Fernández', '4886850');
INSERT INTO agenda_telefonica VALUES ('Alejandro Sosa', '4556550');
```

Notar que las sentencias separadas pueden tener semántica diferente (especialmente con respecto a los triggers), y puede tener diferente rendimiento que la sentencia de inserción múltiple.

Para insertar varias filas en MS SQL puede utilizar esta construcción:

```
INSERT INTO phone_book
SELECT 'John Doe', '555-1212'
UNION ALL
SELECT 'Peter Doe', '555-2323';
```

Tenga en cuenta que no se trata de una sentencia SQL válida de acuerdo con el estándar SQL (SQL: 2003), debido a la cláusula subselect incompleta.

Para hacer lo mismo en Oracle se usa la Tabla DUAL, siempre que se trate de solo una fila simple:

```
INSERT INTO phone_book
SELECT 'John Doe', '555-1212' FROM DUAL
UNION ALL
SELECT 'Peter Doe','555-2323' FROM DUAL
```

Una implementación conforme al estándar de esta lógica se muestra en el siguiente ejemplo, o como se muestra arriba (no aplica en Oracle):

```
INSERT INTO phone_book
SELECT 'John Doe', '555-1212' FROM LATERAL ( VAL-
UES (1) ) AS t(c)
UNION ALL
SELECT 'Peter Doe','555-2323' FROM LATERAL ( VAL-
UES (1) ) AS t(c)
```

Copia de filas de otras tablas un INSERT también puede utilizarse para recuperar datos de otros, modificarla si es necesario e insertarla directamente en la tabla. Todo esto se hace en una sola sentencia SQL que no implica ningún procesamiento intermedio en la aplicación cliente. Un SUBSELECT se utiliza en lugar de la cláusula VALUES. El SUBSELECT puede contener la sentencia JOIN, llamadas a funciones, y puede incluso consultar en la misma TABLA los datos que se insertan. Lógicamente, el SELECT se evalúa antes de que la operación INSERT esté iniciada. Un ejemplo se da a continuación:

```
INSERT INTO phone_book2
SELECT *
FROM phone_book
WHERE name IN ('John Doe', 'Peter Doe');
```

Una variación es necesaria cuando algunos de los datos de la tabla fuente se está insertando en la nueva tabla, pero no todo el registro. (O cuando los esquemas de las tablas no son iguales.)

```
INSERT INTO phone_book2 ([name], [phoneNumber])
SELECT [name], [phoneNumber]
FROM phone_book
WHERE name IN ('John Doe', 'Peter Doe');
```

El SELECT produce una tabla (temporal), y el esquema de la tabla temporal debe coincidir con el esquema de la tabla donde los datos son insertados.

UPDATE (Actualizar) una sentencia UPDATE de SQL es utilizada para modificar los valores de un conjunto de registros existentes en una tabla.
Ejemplo:

```
UPDATE My_table SET field1 = 'updated value' WHERE
field2 = 'N';
```

DELETE (Borrar) una sentencia DELETE de SQL borra uno o más registros existentes en una tabla.
Forma básica:
DELETE FROM tabla WHERE columna1 = 'valor1';
Ejemplo:

```
DELETE FROM mi_tabla WHERE columna2 = 'N';
```

Funciones Disponemos de varias funciones que podemos usar, por ejemplo la función MIN(), devuelve el valor más pequeño de la columna seleccionada:

```
SELECT MIN(columna)
FROM tabla
WHERE condición;
```

La función MAX() devuelve el valor más grande de la columna seleccionada:

```
SELECT MAX(columna)
FROM tabla
WHERE condición;
```

La función COUNT() devuelve el número de filas de la consulta, es decir, el número de registros que cumplen una determinada condición:

```
SELECT COUNT(columna)
FROM tabla
WHERE condición;
```

La función AVG() devuelve el valor promedio de una columna especifica (numérica):

```
SELECT AVG(columna)
FROM tabla
WHERE condición;
```

La función SUM() devuelve la suma total de una columna especifica (numérica):

```
SELECT SUM(columna)
FROM tabla
WHERE condición;
```

Caracteres Comodín Los comodines son caracteres especiales que se utilizan con las palabras clave LIKE y NOT LIKE. Esto permite buscar datos con patrones sofisticados de manera bastante eficiente.

- % Equivale a cero o más caracteres

```
SELECT * FROM clientes
WHERE nombre LIKE '%ory';
```

- _ Equivale a cualquier carácter individual

```
SELECT * FROM clientes
WHERE ciudad LIKE '___vale';
```

- [listacaracteres] Equivale a cualquier carácter en la lista

```
SELECT * clientes
WHERE nombre LIKE '[JKT]%';
```

Claves En las bases de datos relacionales, existe un concepto de claves primarias y externas . En las tablas SQL, estos se incluyen como restricciones, donde una tabla puede tener una clave principal, una clave externa o ambas.

Claves Primarias una clave primaria permite que cada registro de una tabla se identifique de forma única. Solo puede tener una clave principal por tabla y puede asignar esta restricción a cualquier columna única o combinación. Sin embargo, esto significa que cada valor dentro de esta (s) columna (s) debe ser único.

Por lo general, en una tabla, la columna de ID es una clave principal y, por lo general, está emparejada con la palabra clave AUTO_INCREMENT. Esto significa que el valor aumenta automáticamente a medida que se crean nuevos registros.

Ejemplo: Cree una nueva tabla y establezca la clave principal en la columna ID.

```
CREATE TABLE clientes (
    id int NOT NULL AUTO_INCREMENT,
    Nombre varchar(255),
    Apellido varchar(255) NOT NULL,
    Direccion varchar(255),
    Email varchar(255),
    PRIMARY KEY (id)
);
```

Claves Externas se puede aplicar una clave externa a una columna o a muchas. Se usa para vincular 2 tablas juntas en una base de datos relacional.

- La tabla que contiene la clave externa se llama clave secundaria ,
- La tabla que contiene la clave a la que se hace referencia (o candidata) se denomina tabla principal .

Básicamente, esto significa que los datos de la columna se comparten entre 2 tablas, porque una clave externa también evita que se inserten datos no válidos que no están presentes en la tabla principal.

Ejemplo: Cree una nueva tabla y convierta cualquier columna que haga referencia a ID en otras tablas en claves externas.

```
CREATE TABLE ordenes (
id int NOT NULL,
usuario_id int,
```

```
producto_id int,
PRIMARY KEY (id),
FOREIGN KEY (usuario_id) REFERENCES users(id),
FOREIGN KEY (producto_id) REFERENCES products(id)
);
```

Índices Los índices son atributos que se pueden asignar a columnas en las que se buscan con frecuencia para que la recuperación de datos sea un proceso más rápido y eficiente.

```
CREATE INDEX idx_test
ON users (first_name, surname);
CREATE UNIQUE INDEX idx_test
ON users (first_name, surname);
DROP INDEX
ALTER TABLE users
DROP INDEX idx_test;
```

Uniones En SQL, JOINse utiliza una cláusula para devolver un resultado que combina datos de varias tablas, basándose en una columna común que aparece en ambas.

Hay varias combinaciones diferentes disponibles para su uso:

- Inner Join (predeterminado): Devuelve cualquier registro que tenga valores coincidentes en ambas tablas.
- Left Join: Devuelve todos los registros de la primera tabla, junto con los registros coincidentes de la segunda tabla.
- Right Join: Devuelve todos los registros de la segunda tabla, junto con los registros coincidentes de la primera.
- Full Join: Devuelve todos los registros de ambas tablas cuando hay una coincidencia.

```
SELECT orders.id, users.FirstName, users.Surname, products.name
as 'product name'
FROM orders
INNER JOIN users on orders.user_id = users.id
INNER JOIN products on orders.product_id = products.id;
```

Vistas en SQL Una vista es esencialmente un conjunto de resultados de SQL que se almacena en la base de datos bajo una etiqueta, por lo que puede volver a él más tarde sin tener que volver a ejecutar la consulta. Estos son especialmente útiles cuando tiene una consulta SQL costosa que puede necesitar varias veces. Entonces, en lugar de ejecutarlo una y otra vez para generar el mismo conjunto de resultados, puede hacerlo una vez y guardarlo como una vista.

Cómo crear vistas para crear una vista, puede hacerlo así:

```
CREATE VIEW priority_users AS
SELECT * FROM users
WHERE country = 'United Kingdom';
```

Luego, en el futuro, si necesita acceder al conjunto de resultados almacenado, puede hacerlo así:

```
SELECT * FROM [priority_users];
```

Cómo reemplazar vistas con el CREATE OR REPLACE comando, podemos actualizar una vista como esta:

```
CREATE OR REPLACE VIEW [priority_users] AS
SELECT * FROM users
WHERE country = 'United Kingdom' OR country='USA';
```

Cómo eliminar vistas para eliminar una vista, simplemente usamos DROP VIEW comando.

```
DROP VIEW priority_users;
```

Recuperación de Clave Los diseñadores de base de datos que usan una clave suplente como la clave principal para cada tabla, se ejecutará en el escenario ocasional en el que es necesario recuperar automáticamente la base de datos, generando una clave primaria de una sentencia SQL INSERT para su uso en otras sentencias SQL. La mayoría de los sistemas no permiten sentencias SQL INSERT para retornar la fila de datos. Por lo tanto, se hace necesario aplicar una solución en tales escenarios.

Implementaciones comunes incluyen:

- Utilizando un procedimiento almacenado específico de base de datos que genera la clave suplente, realice la operación INSERT y finalmente devuelva la clave generada.

- Utilizando una sentencia SELECT específica de base de datos, sobre una tabla temporal que contiene la última fila insertada. DB2 implementa esta característica de la siguiente manera:

```
SELECT *
FROM NEW TABLE (
INSERT INTO phone_book
VALUES ('Cristobal Jeldrez','0426.817.10.30')
) AS t
```

- Utilizando una sentencia SELECT después de la sentencia INSERT con función específica de base de datos, que devuelve la clave primaria generada por el registro insertado más recientemente.

- Utilizando una combinación única de elementos del original SQL INSERT en una sentencia posterior SELECT.

- Utilizando un GUID en la sentencia SQL INSERT y la recupera en una sentencia SELECT.

- Utilizando la función de PHP mysql_insert_id() de MySQL después de la sentencia INSERT.

- Utilizando un INSERT con la cláusula RETURNING para Oracle, que solo se puede utilizar dentro de un bloque PL/SQL, en el caso de PostgreSQL se puede usar tanto con SQL como con PL/SQL.

```
INSERT INTO phone_book VALUES ('Cristobal Jeldrez',
'0426.817.10.30')
RETURNING phone_book_id INTO v_pb_id
```

- En el caso de MS SQL se puede utilizar la siguiente instrucción:

```
Set NoCount On;
INSERT INTO phone_book VALUES ('Cristobal Jeldrez',
'0426.817.10.30');
Select @@Identity as id
```

8.3 ¿Qué son las bases de datos NoSQL?

A veces llamado "no solo SQL" es una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico de SGBDR (Sistema de Gestión de Bases de Datos Relacionales) en aspectos importantes, siendo el más destacado que no usan SQL como lenguaje principal de consultas. Los datos almacenados no requieren estructuras fijas como tablas, normalmente no soportan operaciones JOIN, ni garantizan completamente ACID (atomicidad, consistencia, aislamiento y durabilidad) y habitualmente escalan bien horizontalmente. Los sistemas NoSQL se denominan a veces "no solo SQL" para subrayar el hecho de que también pueden soportar lenguajes de consulta de tipo SQL.

Por lo general, los investigadores académicos se refieren a este tipo de bases de datos como almacenamiento estructurado, término que abarca también las bases de datos relacionales clásicas. A menudo, las bases de datos NoSQL se clasifican según su forma de almacenar los datos, y comprenden categorías como clave-valor, las implementaciones de BigTable, bases de datos documentales, y bases de datos orientadas a grafos.

Las bases de datos NoSQL son ampliamente reconocidas porque son fáciles de desarrollar, por su funcionalidad y el rendimiento a escala. Bastantes sistemas NoSQL emplean una arquitectura distribuida, manteniendo los datos de forma redundante en varios servidores, usando frecuentemente una tabla Hash distribuida. De esta forma, el sistema puede realmente escalar añadiendo más servidores, y el fallo en un servidor puede ser tolerado.

¿Cómo funciona una base de datos NoSQL (no relacionales)? Las bases de datos NoSQL utilizan una variedad de modelos de datos para acceder y administrar datos. Estos tipos de bases de datos están optimizados específicamente para aplicaciones que requieren grandes volúmenes de datos, baja latencia y modelos de datos flexibles, lo que se logra mediante la flexibilización de algunas de las restricciones de coherencia de datos en otras bases de datos.

Considere el ejemplo de modelado del esquema para una base de datos simple de libros:

- En una base de datos relacional, un registro de libros a menudo se enmascara (o "normaliza") y se almacena en tablas separadas, y las relaciones se definen mediante restricciones de claves primarias y externas. En este ejemplo, la tabla *Libros* tiene las columnas *ISBN, Título*

del libro y Número de edición, la tabla *Autores* tiene las columnas *IDAutor y Nombre de autor* y, finalmente, la tabla *Autor-ISBN* tiene las columnas *IDAutor e ISBN*. El modelo relacional está diseñado para permitir que la base de datos aplique la integridad referencial entre tablas en la base de datos, normalizada para reducir la redundancia y, generalmente, está optimizada para el almacenamiento.

- En una base de datos NoSQL, el registro de un libro generalmente se almacena como un documento JSON. Para cada libro, el elemento *ISBN, Título del libro, Número de edición, Nombre autor y IDAutor* se almacenan como atributos en un solo documento. En este modelo, los datos están optimizados para un desarrollo intuitivo y escalabilidad horizontal.

¿Por qué debería usar una base de datos NoSQL? Las bases de datos NoSQL se adaptan perfectamente a muchas aplicaciones modernas, como dispositivos móviles, Web y juegos, que requieren bases de datos flexibles, escalables, de alto rendimiento y altamente funcionales para proporcionar excelentes experiencias de usuario.

- Flexibilidad: las bases de datos NoSQL generalmente ofrecen esquemas flexibles que permiten un desarrollo más rápido y más iterativo. El modelo de datos flexible hace que las bases de datos NoSQL sean ideales para datos semiestructurados y no estructurados.

- Escalabilidad: las bases de datos NoSQL generalmente están diseñadas para escalar usando clústeres distribuidos de Hardware en lugar de escalar añadiendo servidores caros y sólidos. Algunos proveedores de la nube manejan estas operaciones en segundo plano, como un servicio completamente administrado.

- Alto rendimiento: la base de datos NoSQL está optimizada para modelos de datos específicos y patrones de acceso que permiten un mayor rendimiento que el intento de lograr una funcionalidad similar con bases de datos relacionales.

- Altamente funcional: las bases de datos NoSQL proporcionan APIs altamente funcionales y tipos de datos que están diseñados específicamente para cada uno de sus respectivos modelos de datos.

Algunos ejemplos de bases de datos NoSQL son: DynamoDB, Cassandra, Redis, CouchDB, RethinkDB, RavenDB, Elasticsearch y MongoDB.

Tipos de bases de datos NoSQL

- Clave-valor: las bases de datos clave-valor son altamente divisibles y permiten escalado horizontal a escalas que otros tipos de bases de datos no pueden alcanzar. Los casos de uso como juegos, tecnología publicitaria e IoT se prestan particularmente bien con el modelo de datos clave-valor. Amazon DynamoDB está diseñado para proporcionar una latencia de milisegundos constante de un solo dígito para cualquier escala de cargas de trabajo. Este rendimiento sistemático es uno de los principales elementos que explican por qué la característica de historias de Snapchat, que incluye la carga de trabajo de escritura de almacenamiento más grande de Snapchat, se trasladó a DynamoDB.

- Documentos: en el código de aplicación, los datos se representan a menudo como un objeto o un documento de tipo JSON porque es un modelo de datos eficiente e intuitivo para los desarrolladores. Las bases de datos de documentos facilitan a los desarrolladores el almacenamiento y la consulta de datos en una base de datos mediante el uso del mismo formato de modelo de documento que emplean en el código de aplicación. La naturaleza flexible, semiestructurada y jerárquica de los documentos y las bases de datos de documentos permiten que evolucionen según las necesidades de las aplicaciones. El modelo de documentos funciona bien con catálogos, perfiles de usuario y sistemas de administración de contenido en los que cada documento es único y evoluciona con el tiempo. Amazon DocumentDB (con compatibilidad para MongoDB) y MongoDB son bases de datos de documentos conocidas que proporcionan APIs poderosas e intuitivas para un desarrollo flexible e iterativo.

- Gráficos: el propósito de una base de datos de gráficos es facilitar la creación y la ejecución de aplicaciones que funcionan con conjuntos de datos altamente conectados. Los casos de uso típicos para una base de datos de gráficos incluyen redes sociales, motores de recomendaciones, detección de fraude y gráficos de conocimiento. Amazon Neptune es un servicio de base de datos de gráficos completamente administrado. Neptune admite tanto el modelo de Property Graph como el Resource

Description Framework (RDF), que ofrece la opción de dos API de gráficos: TinkerPop y RDF/SPARQL. Las bases de datos de gráficos populares incluyen Neo4j y Giraph.

- En memoria: las aplicaciones de juegos y tecnología publicitaria tienen casos de uso como tablas de clasificación, tiendas de sesión y análisis en tiempo real que requieren tiempos de respuesta de microsegundos y pueden tener grandes picos de tráfico en cualquier momento. Amazon ElastiCache ofrece MemCachéd y Redis, para servir cargas de trabajo de baja latencia y alto rendimiento, como McDonald's, en las que no se pueden servirse con almacenes de datos basados en disco. Amazon DynamoDB Accelerator (DAX) es otro ejemplo de un almacén de datos especialmente diseñado. DAX hace que DynamoDB lea una orden de magnitud más rápida.

- Buscar: muchas aplicaciones generan registros para ayudar a los desarrolladores a solucionar problemas. Amazon Elasticsearch Service (Amazon ES) está diseñado para proporcionar visualizaciones en tiempo real y análisis de datos generados por máquinas al indexar, agregar y buscar registros y métricas semiestructuradas. Amazon ES también es un poderoso motor de búsqueda de alto rendimiento para casos de uso de búsqueda de texto completo. Expedia está utilizando más de 150 dominios de Amazon ES, 30 TB de datos y 30 mil millones de documentos para una variedad de casos de uso críticos, que van desde el monitoreo operativo y la resolución de problemas, hasta el seguimiento de la pila de aplicaciones distribuidas y la optimización de precios.

8.3.1 SQL en comparación con NoSQL

Durante décadas, el modelo de datos predominante utilizado para el desarrollo de aplicaciones era el modelo de datos relacional empleado por bases de datos relacionales como Oracle, DB2, SQL Server, MySQL y PostgreSQL. No fue sino hasta mediados y finales de la década del 2000 que otros modelos de datos comenzaron a adoptarse y aumentó su uso significativamente. Para diferenciar y categorizar estas nuevas clases de bases de datos y modelos de datos, se acuñó el término "NoSQL". Con frecuencia, los términos "NoSQL" y "no relacional" se usan indistintamente.

Cargas de trabajo óptimas las bases de datos relacionales están diseñadas para aplicaciones de procesamiento de transacciones online (OLTP) altamente coherentes y transaccionales, y son buenas para el procesamiento analítico online (OLAP), en cambio las bases de datos NoSQL están diseñadas para varios patrones de acceso a datos que incluyen aplicaciones de baja latencia. Las bases de datos de búsqueda NoSQL están diseñadas para hacer análisis sobre datos semiestructurados.

Modelo de datos el modelo relacional normaliza los datos en tablas conformadas por filas y columnas. Un esquema define estrictamente las tablas, las filas, las columnas, los índices, las relaciones entre las tablas y otros elementos de las bases de datos. La base de datos impone la integridad referencial en las relaciones entre tablas, en cambio las bases de datos NoSQL proporcionan una variedad de modelos de datos, como clave-valor, documentos y gráficos, que están optimizados para el rendimiento y la escala.

Propiedades ACID las bases de datos relacionales ofrecen propiedades de atomicidad, coherencia, aislamiento y durabilidad (ACID):

- La atomicidad requiere que una transacción se ejecute por completo o no se ejecute en absoluto.
- La coherencia requiere que una vez confirmada una transacción, los datos deben acoplarse al esquema de la base de datos.
- El aislamiento requiere que las transacciones simultáneas se ejecuten por separado.
- La durabilidad requiere la capacidad de recuperarse de un error inesperado del sistema o de un corte de energía y volver al último estado conocido.

Las bases de datos NoSQL a menudo hacen concesiones al flexibilizar algunas de las propiedades ACID de las bases de datos relacionales para un modelo de datos más flexible que puede escalar horizontalmente. Esto hace que las bases de datos NoSQL sean una excelente opción para casos de uso de baja latencia y alto rendimiento que necesitan escalar horizontalmente más allá de las limitaciones de una sola instancia.

Rendimiento en las bases de datos relacionales normalmente, el rendimiento depende del subsistema de disco. Se necesita la optimización de consultas, índices y estructura de tabla para lograr el máximo rendimiento, en cambio en las bases de datos NoSQL el rendimiento depende del tamaño del clúster de Hardware subyacente, la latencia de red y la aplicación que efectúa la llamada.

Escalado las bases de datos relacionales generalmente escalan en forma ascendente las capacidades de computación del Hardware o la ampliación mediante la adición de réplicas para cargas de trabajo de solo lectura, en cambio en las bases de datos NoSQL normalmente se pueden particionar porque los patrones de acceso son escalables mediante el uso de arquitectura distribuida para aumentar el rendimiento que proporciona un rendimiento constante a una escala casi ilimitada.

API en las bases de datos relacionales se solicita almacenar y recuperar datos que están comunicados mediante consultas que se ajustan a un lenguaje de consulta estructurado (SQL). Estas consultas son analizadas y ejecutadas por la base de datos relacional, en cambio en las bases de datos NoSQL las APIs basadas en objetos permiten a los desarrolladores almacenar y recuperar fácilmente estructuras de datos. Las claves de partición permiten que las aplicaciones busquen pares de clave-valor, conjuntos de columnas o documentos semiestructurados que contengan atributos y objetos de aplicación serializados.

En los últimos años, las bases de datos SQL y NoSQL incluso han comenzado a fusionarse. Por ejemplo, los sistemas de bases de datos, como PostgreSQL, MySQL y Microsoft SQL Server ahora admiten el almacenamiento y la consulta de datos JSON, al igual que las bases de datos NoSQL. Con esto, ahora puede lograr muchos de los mismos resultados con ambas tecnologías. Pero aún no obtiene muchas de las funciones NoSQL, como el escalado horizontal y la interfaz fácil de usar.

8.3.2 SQL en comparación con Terminología NoSQL

La siguiente tabla compara la terminología utilizada por las bases de datos NoSQL seleccionadas con la terminología utilizada por las bases de datos SQL.

SQL	MongoDB	DynamoDB	Cassandra	Couchbase
Tabla	Conjunto	Tabla	Tabla	Bucketdedatos
Fila	Documento	Elemento	Fila	Documento
Columna	Campo	Atributo	Columna	Campo
Claveprincipal	ObjectId	Clave principal	Clave principal	ID del documento
Índice	Índice	Índice secundario	Índice	Índice
Ver	Ver	Índice secundario global	Vista materializada	Ver
Tabla u objeto anidado	Documento incrustado	Mapa	Mapa	Mapa
Matriz	Matriz	Lista	Lista	Lista

Un conjunto de *Campos* formarán un *Documento*, que en caso de asociarse con otros formará una *Colección*. Las *Bases de Datos* estarán formadas por *Colecciones*, y a su vez, cada *Servidor* puede tener tantas bases de datos como el equipo lo permita.

Los tipos de datos soportados son: Numéricos:

- Double: punto flotante de 64 bits.
- Decimal: punto flotante de 128 bits.
- Int: enteros de hasta 32 bits.
- Long: enteros de hasta 64 bits.

Texto:

- String: Cadenas UTF-8.
- Regex: Almacena expresiones regulares.

Fecha:

- Date: entero de 64 bits representa en número de milisegundos desde el primero de enero de 1970.
- Timestamp: entero de 64 bits, en el que los primeros 32 bits representan los segundos pasados desde el primero de enero de 1970 y los otros 32 bits son ordinales incrementales.

Especiales:

- Array: almacena un conjunto de elementos de cualquier tipo.
- Objectld: tipo de dato único, principalmente utilizados para dar valor al campo _id de los documentos.
- Javascript: código de Javascript
- Null: valor nulo
- Boolean: valor booleano.

Consulta y modificación de datos Los datos pueden ser accedidos mediante operaciones CRUD (Create, Read, Update, Delete), la ejecución de cualquiera de estas instrucciones sigue el siguiente esquema:

db.<nombre_colección>.<nombre_Operación>(<condiciones_de_la_operación>)

Operaciones:

- Find: lectura (Read) de datos
- Insert: escritura (Create) de datos
- Update: actualización de valores de datos
- Remove: borrado (Delete) de datos

A modo de ejemplo utilizaremos un JSON como el que se usa en mongoDB. Supongamos que vamos a registrar diferentes personas en una colección perteneciente a una BBDD NoSQL con algunos campos especiales. Estos no necesariamente tienen que seguir un patrón específico como veremos a continuación:

```
item1 = {
    Nombre: "Luís",
    Apellidos: "Martinez",
    Edad: 18,
    Aficiones: ["fútbol","senderismo","tenis"],
    Amigos: [
        {
```

```
            Nombre:"Monica",
            Edad:20
        },
        {
            Nombre:"Andrés",
            Edad:24
        }
    ]
}
db.hi.insert(item1)
db.amigos.find().pretty()
```

Ahora bien, si queremos añadir otros datos con algunas características diferentes la podemos hacer sin mayor problema introduciendo lo siguiente:

```
item2 = {
    Nombre: "Luís",
    Apellidos: "Martinez",
    Edad: 18,
}
db.hi.insert(item2)
db.amigos.find().pretty()
```

En un modelo relacional o SQL clásico esto sería imposible de hacer. Esta es una de las tantas ventajas que hemos comentado

Operación Find ejemplo del comando Find, para encontrar a alguien de nombre Luis:

```
db.amigos.find({Nombre:"Luís"})
```

ejemplo del comando Find, para encontrar a alguien de nombre Luis y apellidos Martinez:

```
db.amigos.find({$and:[{Nombre:"Luís"},{Apellidos:"Martinez"}]})
```

ejemplo del comando Find, para encontrar a alguien de nombre Luis y apellidos Martinez o edad mayor a 18:

```
db.amigos.find({
    $or:[
        {$and:[{Nombre:"Luís"},{Apellidos:"Martinez"}]},
        {Edad:{gte:18}}
    ]
})
```

ejemplo en el que el campo Amigos exista:

```
db.amigos.find({Amigos:{$exists:true}})
```

Lista de comando ejecutables contra la base de datos:

db.adminCommand(nameOrDocument) - switches to 'admin' db, and runs command [just calls db.runCommand(...)]
db.auth(username, password)
db.cloneDatabase(fromhost)
db.commandHelp(name) returns the help for the command
db.copyDatabase(fromdb, todb, fromhost)
db.createCollection(name, { size : ..., capped : ..., max : ... })
db.createUser(userDocument)
db.currentOp() displays currently executing operations in the db
db.dropDatabase()
db.fsyncLock() flush data to disk and lock server for backups
db.fsyncUnlock() unlocks server following a db.fsyncLock()
db.getCollection(cname) same as db['cname'] or db.cname
db.getCollectionInfos([filter]) - returns a list that contains the names and options of the db's collections
db.getCollectionNames()
db.getLastError() - just returns the err msg string
db.getLastErrorObj() - return full status object
db.getLogComponents()
db.getMongo() get the server connection object
db.getMongo().setSlaveOk() allow queries on a replication slave server
db.getName()

db.getPrevError()
db.getProfilingStatus() - returns if profiling is on and slow threshold
db.getReplicationInfo()
db.getSiblingDB(name) get the db at the same server as this one
db.getWriteConcern() - returns the write concern used for any operations on this db, inherited from server object if set
db.hostInfo() get details about the server's host
db.isMaster() check replica primary status
db.killOp(opid) kills the current operation in the db
db.listCommands() lists all the db commands
db.loadServerScripts() loads all the scripts in db.system.js
db.logout()
db.printCollectionStats()
db.printReplicationInfo()
db.printShardingStatus()
db.printSlaveReplicationInfo()
db.dropUser(username)
db.repairDatabase()
db.resetError()
db.runCommand(cmdObj) run a database command. if cmdObj is a string, turns it into { cmdObj : 1 }
db.serverStatus()
db.setLogLevel(level,<component>)
db.setProfilingLevel(level,<slowms>) 0=off 1=slow 2=all
db.setWriteConcern(<write concern doc>) - sets the write concern for writes to the db
db.unsetWriteConcern(<write concern doc>) - unsets the write concern for writes to the db
db.setVerboseShell(flag) display extra information in shell output
db.shutdownServer()
db.stats()
db.version() current version of the server

8.4 Bases de Datos de Código Abierto y sus Ventajas Frente a las Propietarias

Elegir la base de datos más adecuada para un proyecto supone enfrentarse a todo tipo de posibilidades de elección. Quizá la primera, y la más importante, es elegir una base de datos de código abierto o una propietaria. Es decir, una de código abierto, generalmente gratuita, o una propiedad de una empresa. A la vista de esto, la primera ventaja de las bases de datos de código abierto salta a la vista: evitan tener que hacer frente a un desembolso económico fuerte, puesto que no hace falta comprar su licencia a un fabricante. A no ser que tengas necesidades muy concretas y específicas, pero la proliferación de bases de datos de código abierto en los últimos años, así como de diversos complementos para ellas, hace que resulten cada vez más atractivas incluso para fines muy concretos.

Mientras tanto, la popularidad de los sistemas de bases de datos con licencia comercial, es decir, de las propietarias, ha ido cayendo a lo largo de los últimos años. Y según la lista de sistemas de gestión de bases de datos de DB-Engines, en muchos casos han sido superadas por las bases de datos de código abierto en lo que respecta a la adopción en empresas.

Ventajas de las Bases de Datos de Código Abierto Una base de datos de código abierto es un sistema de gestión de bases de datos abierto y de uso gratuito, y quizá una de las pocas desventajas que tiene frente a las bases de datos propietarias es que no cuenta con un servicio de soporte técnico por parte de un proveedor, algo con lo que sí cuentan las bases de datos propietarias. Pero precisamente de esta desventaja nace una de sus ventajas.

Las bases de datos de código abierto cuentan con una nutrida comunidad de usuarios que, en muchos casos no solo comparten sus conocimientos en foros y grupos de usuarios especializados, sino que en muchos casos están dispuestos a ayudar a otros usuarios con problemas y necesidades concretas con ellas.

Además, la información que se guarda en una base de datos de código abierto es de quien la inserta en ella. Sin ningún tipo de restricción. También puedes modificarla para adaptarla a tus necesidades, aunque para ello necesitas contar con ciertos conocimientos de desarrollo. Puedes crear aplicaciones y complementos con base en el Software de código abierto que elijas, de una manera muy parecida a como sucede por ejemplo con el CMS Word-

Press. Por lo tanto, las bases de datos de código abierto te ofrecen un sinfín de posibilidades de personalización y ampliación.

Las Principales Bases de Datos de Código Abierto El número de bases de datos de código abierto crece cada vez más a medida que pasan los años. No obstante, entre todas las posibilidades, hay varias que, según TechTarget, o bien por su sencillez y robustez, o bien por sus posibilidades y facilidades de ampliación y personalización, son las que consiguen una mayor adopción y despliegue en las empresas. Entre ellas están las seis siguientes, quizá las más populares y utilizadas.

MySQL Sin duda esta es la más extendida de las bases de datos de código abierto, y durante muchos años, la opción por defecto para muchos desarrolladores a la hora de optar por una base de datos. Oracle se hizo con ella hace ya bastantes años cuando compró Sun Microsystems, y es un sistema de gestión de bases de datos relacional. Esto quiere decir que utiliza tablas para almacenar los datos, y los tipos de datos almacenados deben estar relacionados entre sí de alguna manera. Cuenta con una edición de código abierto y algunas versiones de pago, y debido a su popularidad hay muchas herramientas de terceros para ella, así como una cantidad importante de documentación centrada en facilitar su aprendizaje.

La base de MySQL se originó a partir de un sistema de base de datos propietario antiguo, mantener la mayor parte de su compatibilidad y convertir los resultados en código abierto. Sus creadores se centraron en la velocidad, y en la actualidad sigue estando considerada entre las opciones más rápidas de bases de datos. Además, de la velocidad, cuenta con un Script que contribuye a la mejora de la seguridad de las bases de datos de los usuarios. Esta base de datos es compatible, entre otros, con los lenguajes de programación C, C++, Java, Python y Ruby.

No obstante, se trata de una base de datos con licencia dual. Esto implica que algunas de sus funciones y Plugins solo son accesibles en ediciones propietarias de pago. En cuanto a su curva de aprendizaje, no es muy elevada. No es necesario saber mucho SQL para utilizarla, y puedes trabajar con la base de datos desde la línea de comandos. MySQL tiene una compatibilidad elevada prácticamente con cualquier sistema operativo. Se trata de una base de datos sólida, rápida y versátil, por lo que es adecuada para multitud de casos de uso.

MariaDB Cuando Oracle compró Sun, y con ella MySQL, hubo quien no se lo tomó muy bien, porque aunque Oracle ha mantenido MySQL como código abierto, la compañía no tiene mucha reputación que digamos como defensora del Software abierto. Uno de los que no se creyó a Oracle fue uno de los propios fundadores de MySQL, que dejó la compañía para crear MariaDB, un fork de MySQL con una compatibilidad casi completa con ella.

Se puede utilizar como un sustituto prácticamente para todo de MySQL, aunque cuenta con varias funciones únicas que la distinguen de ella. Para empezar, MariaDB usa el motor de almacenamiento Aria para gestionar querys de SQL complejas. Esto hace que la base de datos sea todavía más veloz que MySQL. Además, puedes utilizar filas dinámicas como columnas de tabla, lo que la hace más flexible y adaptable a diversas situaciones.

Además, cuenta con motores de almacenamiento especializados para casos de uso concretos que no tiene MySQL. Por ejemplo, permite implementar el almacenamiento distribuido o las transacciones distribuidas. Como hemos mencionado, puedes utilizar MariaDB en los mismos supuestos y casos que MySQL, pero no hay compatibilidad entre las dos, por lo que tendrás que elegir cuál de las dos quieres utilizar.

PostgreSQL Esta base de datos relacional y de código abierto es muy utilizada en ciencia de datos y creación de gráficos, y también en el sector de la Inteligencia Artificial. Esto último se debe a que PostgreSQL está especialmente pensada para aplicaciones de Python y Ruby. No obstante, también es compatible con PHP. Se ha desarrollado pensando en la elegancia, y cuenta con ciertas funciones bastante diferenciadas. Entre ellas están la posibilidad de implementar la replicación asíncrona, y el soporte nativo para el almacenamiento de documentos de estilo JSON o XML.

Además, PostgreSQL permite hacer búsquedas completas de texto en la base de datos, y cuenta con varios tipos de datos integrados de gran valor para algunas aplicaciones, como los rangos, los arrays y la geolocalización. Eso sí, no es excesivamente adecuada para aplicaciones que requieran operaciones intensivas de lectura, y la creación de informes de datos con regularidad puede hacer que el almacenamiento de documentos no esté muy fino si tiene que almacenar un conjunto de datos muy grande.

SQLite También relacional, SQLite es una base de datos de código abierto compuesta por una librería ligera y pequeña que ofrece un motor de

base de datos. A pesar de su ligereza, permite generar bases de datos de cientos de Terabytes, y un tamaño máximo de filas de un Gigabyte. Incluso con archivos de este tamaño, SQLite sigue siendo rápida. Además, es compatible con Java, Python, C o C++, entre otros lenguajes.

Cuenta con decenas de casos de uso, y por ejemplo, la encontrarán muy útil los desarrolladores de aplicaciones sencillas. Está integrada en smartphones y en muchos ordenadores. También es muy valorada en aplicaciones de Internet de las Cosas y Webs de poco tráfico. Eso sí, por su estructura, no está indicada para sitios con mucho tráfico, ya que su rendimiento se resiente. Además, cuenta con ciertas limitaciones que en algunos casos pueden ser importantes.

Estas hacen que, entre otras cosas, no esté indicada en aplicaciones a las que tienen que acceder varios usuarios con permisos de acceso especiales. SQLite lee y escribe en un archivo de disco común y corriente, por lo que los únicos permisos de acceso aplicables para esta base de datos son los incluidos habitualmente en el sistema operativo.

MongoDB Es un sistema de base de datos NoSQL orientado a documentos de código abierto y escrito en C++, que en lugar de guardar los datos en tablas lo hace en estructuras de datos BSON (similar a JSON) con un esquema dinámico. Al ser un proyecto de código abierto, sus binarios están disponibles para los sistemas operativos Windows, GNU/Linux, OS X y Solaris y es usado en múltiples proyectos o implementaciones en empresas como MTV Network, Craigslist, BCI o Foursquare.

La razón de esto es que MongoDB, al estar escrito en C++, cuenta con una más que notoria capacidad para aprovechar los recursos de la máquina y, al estar licenciado bajo una licencia GNU AGPL 3.0, es posible adaptarlo a nuestras necesidades. Si tuviéramos que resumir a una la principal característica a destacar de MongoDB, sin duda esta sería la velocidad, que alcanza un balance perfecto entre rendimiento y funcionalidad gracias a su sistema de consulta de contenidos. Pero sus características principales no se limitan solo a esto, MongoDB cuenta, además, con otras que lo posicionan como el preferido de muchos desarrolladores.

Redis Esta base de datos de código abierto es distinta a las más utilizadas, ya que se utiliza sobre todo en situaciones completamente distintas a ellas. Se trata de una base de datos que permite estructurar datos como

pares de valor clave. Por tanto, se trata de un sistema parecido a los arrays asociativos de PHP o los diccionarios de Python. En realidad, Redis es un sistema para enlazar datos para su referencia, más adelante, con rapidez. Está considerada como la mejor elección como base de datos para cacheado, y también para el trabajo con datos distribuidos.

Para empezar, es una solución de almacenamiento en memoria, enteramente en RAM. Esto quiere decir que sus velocidades de lectura y escritura son muy rápidas. Para aprender sus bases tampoco necesitas mucho tiempo, ya que puedes conocerlas en minutos y empezar con rapidez con el almacenamiento de objetos en ella. Eso sí, puede que no sea muy útil para aplicaciones complejas, pero basta con utilizarlas junto con otras bases de datos, como MariaDB, para el resto de la aplicación que se esté desarrollando.

8.5 Programando en Diferentes Manejadores de Bases de Datos

Existen diferentes paquetes para que permiten trabajar con bases de datos, en esta sección mostraremos cómo trabajar con algunos de ellos:

- SQLite
- MySQL/MariaDB/Microsoft SQL Server
- PostgreSQL
- MongoDB
- Redis

A continuación, mostraremos el mismo ejemplo en cada uno de estos manejadores:

8.5.1 SQLite

es una herramienta de Software libre, que permite almacenar información en dispositivos empotrados de una forma sencilla, eficaz, potente, rápida y en equipos con pocas capacidades de hardware, como puede ser una PDA o un teléfono celular. Está presente en todos los dispositivos Android, iPhone, Mac y Windows 10, además en los navegadores Firefox, Chromium, Chrome y Safari entre otros.

SQLite implementa el estándar SQL92 y también agrega extensiones que facilitan su uso en cualquier ambiente de desarrollo. Esto permite que SQLite soporte desde las consultas más básicas hasta las más complejas del lenguaje SQL, y lo más importante es que se puede usar tanto en dispositivos móviles como en sistemas de escritorio, sin necesidad de realizar procesos complejos de importación y exportación de datos, ya que existe compatibilidad al 100% entre las diversas plataformas disponibles, haciendo que la portabilidad entre dispositivos y plataformas sea transparente.

Lenguajes de Programación de SQLite gracias a que SQLite[136] es Software libre, es posible encontrar una gran cantidad de componentes, librerías y Drivers para interactuar con SQLite desde una gran diversidad de lenguajes y plataformas de programación. Ya sea que estemos utilizando lenguajes modernos como Java, Perl, Python, PHP, Ruby, C#, lenguajes más antiguos como Pascal, SmallTalk, Clipper, o lenguajes poco conocidos como Suneido, REXX, S-Lang, para todos podemos encontrar librerías y ejemplos de código para SQLite.

http://www.sqlite.org/cvstrac/wiki?p=SqliteWrappers ofrece más información sobre "wrappers" para SQLite sobre diferentes plataformas y lenguajes.

Instalación en Debian GNU/Linux Para instalar el paquete usamos:

```
# apt install sqlite3 sqlite3-doc sqlitebrowser
```

y ya podemos iniciar a trabajar, usando

```
# sqlite3
```

SQLite y Python SQLite es probablemente la base de datos más sencilla para conectarse con una aplicación Python, ya que no necesita instalar ningún módulo SQL externo de Python para hacerlo. De forma predeterminada, su instalación de Python contiene una biblioteca SQL de Python llamada *sqlite3* que puede usar para interactuar con una base de datos SQLite.

[136] También existe Rqlite que es una base de datos relacional distribuida y liviana de código abierto para configuraciones de cluster que está construido sobr SQLite y tiene como objetivo ser fácil de usar y tolerante a fallas.

Además, las bases de datos SQLite no tienen servidor y son autónomas, ya que leen y escriben datos en un archivo. Esto significa que, a diferencia de MySQL y PostgreSQL, ni siquiera necesita instalar y ejecutar un servidor SQLite para realizar operaciones de base de datos.

Así es como se usa *sqlite3* para conectarse a una base de datos SQLite en Python:

```
import sqlite3
from sqlite3 import Error

def create_connection(path):
    connection = None
    try:
        connection = sqlite3.connect(path)
        print("Connection to SQLite DB successful")
    except Error as e:
        print(f"The error '{e}' occurred")
    return connection

connection = create_connection("/trayectoria/basedatos")
```

Ejemplo de uso Una vez que hemos ingresado al sistema de base de datos con nuestro usuario, entonces podemos pedirle al sistema que cree o abra una base de datos:

```
sqlite3 prueba.db
```

y estamos en posibilidad de crear nuestras tablas, por ejemplo:

```
create table Productos(
    id_prod int,
    descripcion text,
    precio float(9,2),
    precioVenta float(9,2)
);
```

e ingresamos valores:

```
insert into Productos
values (1, 'Cigarros', 25.5, 30.0), (2, 'Refresco', 4.5, 6.0);
```

y los visualizamos:

```
select * from Productos;
```

ahora procedemos hacer lo pertinente para nuestra otra tabla, usando:

```
create table Ventas(
    id_prod int,
    cantidad int,
    fecha timestamp
);
insert into Ventas
values (1, 2, '2020/12/01 8:01:00'), (2, 5, '2020/12/01 10:15:00'),
(2, 4, '2020/12/01 13:34:00'), (1, 3, '2020/12/01 21:56:00');
select * from Ventas;
```

de esta forma, ahora podemos usar SELECT para hacer un JOIN de nuestras tablas y mostrar el resultado de la solicitud, mediante:

```
    select Ventas.fecha, Ventas.cantidad, Productos.descripcion,
Productos.precioVenta, (Ventas.cantidad * Productos.precioVenta)
    from Ventas
    INNER JOIN Productos ON Ventas.id_prod=Productos.id_prod;
```

8.5.2 MySQL/MariaDB

MySQL/MariaDB es un sistema de administración de bases de datos (Database Management System, DBMS) para bases de datos relacionales. Así, estos manejadores de bases de datos, no son más que una aplicación que permite gestionar archivos llamados de bases de datos.

Existen muchos tipos de bases de datos, desde un simple archivo hasta sistemas relacionales orientados a objetos. MySQL/MariaDB, como base de datos relacional, utiliza múltiples tablas para almacenar y organizar la información. MySQL/MariaDB fue escrito en C y C++ y destaca por su gran adaptación a diferentes entornos de desarrollo, permitiendo su interacción con los lenguajes de programación más utilizados como PHP, Perl y Java y su integración en distintos sistemas operativos.

También es muy destacable, la condición de open source de MySQL/MariaDB, que hace que su utilización sea gratuita e incluso se pueda modificar con total

libertad, pudiendo descargar su código fuente. Esto ha favorecido muy positivamente en su desarrollo y continuas actualizaciones, para hacer de MySQL una de las herramientas más utilizadas por los programadores orientados a Internet.

MariaDB este sistema de gestión de base de datos es una bifurcación de MySQL, aunque parecidos, estos dos sistemas de gestión de bases de datos siguen siendo bastante diferentes:

- MariaDB tiene licencia GPL mientras que MySQL tiene un enfoque de doble licencia
- Cada mango se acumula de una manera diferente
- MariaDB soporta muchos motores de almacenamiento diferente
- En muchos escenarios, MariaDB ofrece un mejor rendimiento

Instalación en Debian GNU/Linux En este caso tenemos dos opciones, instalar MariaDB que ya es parte integral de múltiples aplicaciones dentro de GNU/Linux o usar MySQL, en cada caso el uso es prácticamente el mismo.

Para instalar el paquete MariaDB usamos:

```
# apt install mariadb-server mariadb-client
```

y para configurarlo, usamos:

```
# mysql
> use mysql
> update user set plugin=” where user='root';
> alter user root@localhost identified by 'XXXXXXX';
> flush privilegies;
> exit
```

y para usarlo

```
$ mysql -u root -p
```

Para instalar el paquete MySQL usamos:

```
# apt install mysql-common mysql-client mysql-server mytop
mysql-admin
```

y para configurarlo (las bases de datos se guardan en /var/lib/mysql), usamos:

```
# mysql_secure_installation
```

o

```
# mysql
```

en caso de no crear contraseña, podemos ingresar así:

```
$ mysql -u root -p
```

una vez en la consola del cliente de MySQL creamos un nuevo usuario

```
> create user nombre identified by 'XXXXX';
```

y a continuación le damos privilegios totales

```
> gran all privileges on *.* to nombre with grant option;
> flush privileges;
> exit;
```

otras opciones a instalar son:

```
# apt install ferret mysql-workbench mysql-workbench-data
mycli
# snap install squirrelsql mysql-workbench-community data-
grip dbeaver-ce
```

Crear un Nuevo Usuario Para esto usamos la declaración CREATE USER. La sintaxis básica de la declaración es:

> CREATE USER 'username'@'hostname' IDENTIFIED BY 'password';

En la sintaxis anterior, claramente debes de reemplazar el nombre de usuario y la contraseña con el nombre de usuario y contraseña deseados.

Establezca el hostname en localhost si desea que el usuario pueda conectarse a MySQL Server solo desde localhost. Si desea que el usuario pueda conectarse desde cualquier host, use el comodín % como hostname.

Por ejemplo, crearemos un usuario con el nombre Javier y la contraseña 1234 usando el siguiente comando:

> CREATE USER 'Javier'@'localhost' IDENTIFIED BY '1234';

Sin embargo, este usuario no podrá trabajar con ninguna de las bases de datos MySQL hasta que se le otorguen privilegios adicionales. Ahí el detalle.

Como otorgar privilegios adicionales al usuario inmediatamente después de crear con éxito el nuevo usuario, podemos otorgar privilegios a este nuevo usuario. En la mayoría de los casos, otorgará privilegios a los usuarios de MySQL en función de la base de datos particular a la que la cuenta debería tener acceso. Hay varios tipos de privilegios que se pueden otorgar a una cuenta de usuario, como:

- ALL PRIVILEGES: otorga todos los privilegios a una cuenta de usuario.
- ALTER: el usuario puede cambiar la estructura de una tabla o base de datos.
- CREATE: la cuenta de usuario puede crear bases de datos y tablas.
- DROP: la cuenta de usuario puede eliminar bases de datos y tablas.
- DELETE: la cuenta de usuario puede eliminar filas de una tabla específica.
- INSERT: la cuenta de usuario puede insertar filas en una tabla específica.

- SELECT: la cuenta de usuario puede leer una base de datos.
- UPDATE: la cuenta de usuario puede actualizar las filas de la tabla.

Para proporcionar acceso y otorgar permisos a un $user, generalmente se debe usar la siguiente declaración GRANT:

> GRANT permission_type ON privilege_level TO 'username'@'hostname';

Sí retomamos el ejemplo de arriba, al usuario Javier y su base de datos Javierdb, se le puede otorgar permisos, así:

> GRANT ALL PRIVILEGES ON Javierdb.* TO 'Javier'@'localhost';

Oh claro otorgarle permisos específicos, solo cambiando el ALL PRIVILEGES, por SELECT, CREATE, INSERT.... todos separados por una coma (,).

En algunos casos, es posible que desee crear otro "superusuario". Para otorgar a un usuario los mismos privilegios que el usuario raíz de MySQL, usamos el siguiente comando, que otorga privilegios globales al usuario james que se conecta a través de localhost:

> GRANT ALL ON *.* TO 'Javier'@'localhost' WITH GRANT OPTION;

Como revocar privilegios y eliminar un usuario si necesitas revocar privilegios del usuario Javier en una base de datos Javierdb, sería suficiente con la sintaxis siguiente, que es muy similar

> REVOKE ALL PRIVILEGES ON Javierdb.* TO 'Javier'@'localhost';

Mientras que para eliminar a ese usuario. Basta con:

> DROP USER 'Javier'@'localhost';

Para que cualquier cambio surta efecto, es necesario usar el comando:

> FLUSH PRIVILEGES;

Mantenimiento En caso de requerir mantenimiento a las tablas (analizar, verificar, optimizar y reparar las tablas de una base de datos) podemos usar *mysqlcheck*, por ejemplo para verificar y reparar todas las bases de datos:

```
$ mysqlcheck -A –auto-repair -u root -p
```

para forzar la optimización y autoreparación de todas las tablas:

```
$ mysqlcheck -A –auto-repair –f -o -u root -p
```

para verificar todas las bases de datos:

```
$ mysqlcheck –all-databases -u root -p -c
```

para verificar las tablas de todas las bases de datos:

```
$ mysqlcheck –all-databases -u root -p -a
```

para reparar las tablas de todas las bases de datos:

```
$ mysqlcheck –all-databases -u root -p -r
```

para optimizar las tablas de todas las bases de datos:

```
$ mysqlcheck –all-databases -u root -p -o
```

Exportación e Importación Para exportar una base de datos, debemos utilizar una utilidad de cliente *mysqldump* que crea la copia de seguridad lógica de las bases de datos en archivos de texto SQL, lo que facilita la transferencia de archivos de un servidor a otro, para ello hacemos:

```
$ mysqldump -u usuario -p baseDatos > archivo.sql
```

para restaurarla usamos:

```
$ mysqldump -u usuario -p baseDatos < archivo.sql
```

Exportar como CSV Podemos exportar el contenido de una o más tablas al formato CSV, por ejemplo:

```
> SELECT * FROM test_proyect
INTO OUTFILE '/var/tmp/get_all_queries.csv'
FIELDS ENCLOSED BY '"'
TERMINATED BY ';'
ESCAPED BY '"'
LINES TERMINATED BY '\r\n';
```

MySQL y Python A diferencia de SQLite, no existe un módulo SQL de Python predeterminado que pueda usar para conectarse a una base de datos MySQL. En su lugar, deberá instalar un controlador Python SQL para MySQL a fin de interactuar con una base de datos MySQL desde una aplicación Python. Uno de estos controladores es *mysql-connector-python*. Puede descargar este módulo SQL de Python con pip:

```
$ pip install mysql-connector-python
```

Tenga en cuenta que MySQL es un sistema de gestión de bases de datos basado en servidor. Un servidor MySQL puede tener varias bases de datos. A diferencia de SQLite, donde crear una conexión equivale a crear una base de datos, una base de datos MySQL tiene un proceso de dos pasos para la creación de la base de datos:

- Establezca una conexión a un servidor MySQL.
- Ejecute una consulta separada para crear la base de datos.

Defina una función que se conecte al servidor de la base de datos MySQL y devuelva el objeto de conexión:

```
import mysql.connector
from mysql.connector import Error

def create_connection(host_name, user_name, user_password):
    connection = None
    try:
        connection = mysql.connector.connect(
```

```
                host=host_name,
                user=user_name,
                passwd=user_password
            )
            print("Connection to MySQL DB successful")
        except Error as e:
            print(f"The error '{e}' occurred")
        return connection

connection = create_connection("localhost", "root", "")
```

Ejemplo de uso Una vez que hemos ingresado al sistema de base de datos con nuestro usuario, entonces podemos pedirle al sistema que nos muestre las bases de datos existentes:

```
show databases
```

si necesitamos crear una, por ejemplo prueba, usamos:

```
create databases prueba
```

ahora, le indicamos que usaremos dicha base de datos, mediante:

```
use prueba
```

y ahora podremos, por ejemplo ver las tablas creadas usando:

```
show tables
```

y estamos en posibilidad de crear nuestras tablas, por ejemplo:

```
create table Productos(
    id_prod int,
    descripcion text,
    precio float(9,2),
    precioVenta float(9,2)
);
```

e ingresamos valores:

```
insert into Productos
values (1, 'Cigarros', 25.5, 30.0), (2, 'Refresco', 4.5, 6.0);
```

y los visualizamos:

```
select * from Productos;
```

ahora procedemos hacer lo pertinente para nuestra otra tabla, usando:

```
create table Ventas(
    id_prod int,
    cantidad int,
    fecha timestamp
);
insert into Ventas
values (1, 2, '2020/12/01 8:01:00'), (2, 5, '2020/12/01 10:15:00'),
(2, 4, '2020/12/01 13:34:00'), (1, 3, '2020/12/01 21:56:00');
select * from Ventas;
```

de esta forma, ahora podemos usar SELECT para hacer un JOIN de nuestras tablas y mostrar el resultado de la solicitud, mediante:

```
    Select Ventas.fecha, Ventas.cantidad, Productos.descripcion,
  Productos.precioVenta, (Ventas.cantidad * Productos.precioVenta)
    From Ventas
    INNER JOIN Productos ON Ventas.id_prod=Productos.id_prod;
```

8.5.3 PostgreSQL

Las bases de datos relacionales son uno de los tipos más populares que existen, y su uso nos permite crear sitios web de toda clase. Entre los sistemas de bases de datos relacionales más utilizados se encuentra uno que es conocido como PostgreSQL, que dicho sea de paso está entre los más usados a nivel mundial.

PostgreSQL, o simplemente Postgres para darle un nombre más pintoresco, es un sistema de código abierto de administración de bases de datos del tipo relacional, aunque también es posible ejecutar consultas que sean no relaciones. En este sistema, las consultas relacionales se basan en SQL, mientras que las no relacionales hacen uso de JSON.

Como decíamos, se trata de un sistema de código abierto y además gratuito, y su desarrollo es llevado adelante por una gran comunidad de colaboradores de todo el mundo que día a día ponen su granito de arena para hacer de este sistema una de las opciones más sólidas a nivel de bases de datos.

Dos detalles a destacar de PostgreSQL es que posee data types (tipos de datos) avanzados y permite ejecutar optimizaciones de rendimiento avanzadas, que son características que por lo general solo se ven en sistemas de bases de datos comerciales, como por ejemplo SQL Server de Microsoft u Oracle de la compañía homónima.

Instalación en Debian GNU/Linux Para instalar el paquete usamos:

```
# apt install postgresql postgresql-client postgresql-doc postgresql-contrib
# snap install postgresql10
```

y ya podemos conectarnos al servidor mediante:

```
# postgres psql
```

PostgreSQL y Python Al igual que MySQL, no existe una biblioteca Python SQL predeterminada que pueda usar para interactuar con una base de datos PostgreSQL. En su lugar, debe instalar un controlador Python SQL de terceros para interactuar con PostgreSQL. Uno de esos controladores Python SQL para PostgreSQL es *psycopg2*. Ejecute el siguiente comando en su terminal para instalar el módulo Python SQL psycopg2:

```
$ pip install psycopg2
```

Al igual que con las bases de datos SQLite y MySQL, definirás create_connection () para hacer una conexión con tu base de datos PostgreSQL:

```
import psycopg2
from psycopg2 import OperationalError

def create_connection(db_name, db_user, db_password, db_host, db_port):
    connection = None
    try:
```

```
            connection = psycopg2.connect(
                database=db_name,
                user=db_user,
                password=db_password,
                host=db_host,
                port=db_port,
            )
            print("Connection to PostgreSQL DB successful")
        except OperationalError as e:
            print(f"The error '{e}' occurred")
        return connection

connection = create_connection("postgres", "postgres", "abc123",
"127.0.0.1", "5432")
```

Ejemplo de uso Una vez que hemos ingresado al sistema de base de datos con nuestro usuario, entonces podemos pedirle al sistema que nos muestre las bases de datos existentes:

```
\l
```

si necesitamos crear una, por ejemplo prueba, usamos:

```
create databases prueba
```

ahora, le indicamos que usaremos dicha base de datos, mediante:

```
\c prueba
```

y ahors podremos, por ejemplo ver las tablas creadas usando:

```
\d
```

y estamos en posibilidad de crear nuestras tablas, por ejemplo:

```
create table Productos(
    id_prod int,
    descripcion text not null,
    precio money,
    precioVenta money,
    primary key (id_prod)
);
```

e ingresamos valores:

```
insert into Productos
values (1, 'Cigarros', 25.5, 30.0), (2, 'Refresco', 4.5, 6.0);
```

y los visualizamos:

```
select * from Productos;
```

ahora procedemos hacer lo pertinente para nuestra otra tabla, usando:

```
create table Ventas(
    id_prod int,
    cantidad int,
    fecha timestamp
);
insert into Ventas
values (1, 2, '12/01/2020 8:01:00.00'), (2, 5, '12/01/2020 10:15:00.00'),
(2, 4, '12/01/2020 13:34:00.00'), (1, 3, '12/01/2020 21:56:00.00');
select * from Ventas;
```

de esta forma, ahora podemos usar SELECT para hacer un JOIN de nuestras tablas y mostrar el resultado de la solicitud, mediante:

```
    select Ventas.fecha, Ventas.cantidad, Productos.descripcion,
Productos.precioVenta, (Ventas.cantidad * Productos.precioVenta)
    from Ventas
    INNER JOIN Productos ON Ventas.id_prod=Productos.id_prod;
```

8.5.4 MongoDB

MongoDB (del inglés humongous, "enorme") es un sistema de base de datos NoSQL orientado a documentos de código abierto y escrito en C++, que en lugar de guardar los datos en tablas lo hace en estructuras de datos BSON (similar a JSON) con un esquema dinámico. Al ser un proyecto de código abierto, sus binarios están disponibles para los sistemas operativos Windows, GNU/Linux, OS X y Solaris y es usado en múltiples proyectos o implementaciones en empresas como MTV Network, Craigslist, BCI o Foursquare.

La razón de esto es que MongoDB, al estar escrito en C++, cuenta con una más que notoria capacidad para aprovechar los recursos de la máquina

y, al estar licenciado bajo una licencia GNU AGPL 3.0, es posible adaptarlo a nuestras necesidades.

Si no conocemos a MongoDB, al principio podemos sentirnos un poco perdidos. Al no tener tablas ni nada que se parezca a SQL como referencia, tendremos que estudiar un poco su filosofía y características para entender cómo manejar los datos. Aun así, MongoDB es una seria candidata para almacenar los datos de nuestras aplicaciones.

Características de MongoDB si tuviéramos que resumir a una la principal característica a destacar de MongoDB, sin duda esta sería la velocidad, que alcanza un balance perfecto entre rendimiento y funcionalidad gracias a su sistema de consulta de contenidos. Pero sus características principales no se limitan solo a esto, MongoDB cuenta, además, con otras que lo posicionan como el preferido de muchos desarrolladores.

Características principales:

- Consultas ad hoc. Con MongoDb podemos realizar todo tipo de consultas. Podemos hacer búsqueda por campos, consultas de rangos y expresiones regulares. Además, estas consultas pueden devolver un campo específico del documento, pero también puede ser una función JavaScript definida por el usuario.

- Indexación. El concepto de índices en MongoDB es similar al empleado en bases de datos relacionales, con la diferencia de que cualquier campo documentado puede ser indexado y añadir múltiples índices secundarios.

- Replicación. Del mismo modo, la replicación es un proceso básico en la gestión de bases de datos. MongoDB soporta el tipo de replicación primario-secundario. De este modo, mientras podemos realizar consultas con el primario, el secundario actúa como réplica de datos en solo lectura a modo copia de seguridad con la particularidad de que los nodos secundarios tienen la habilidad de poder elegir un nuevo primario en caso de que el primario actual deje de responder.

- Balanceo de carga. Resulta muy interesante cómo MongoDB puede escalar la carga de trabajo. MongoDB tiene la capacidad de ejecutarse de manera simultánea en múltiples servidores, ofreciendo un balanceo

de carga o servicio de replicación de datos, de modo que podemos mantener el sistema funcionando en caso de un fallo del hardware.

- Almacenamiento de archivos. Aprovechando la capacidad de MongoDB para el balanceo de carga y la replicación de datos, Mongo puede ser utilizado también como un sistema de archivos. Esta funcionalidad, llamada GridFS e incluida en la distribución oficial, permite manipular archivos y contenido.
- Ejecución de JavaScript del lado del servidor. MongoDB tiene la capacidad de realizar consultas utilizando JavaScript, haciendo que estas sean enviadas directamente a la base de datos para ser ejecutadas.

Instalación en Debian GNU/Linux Para instalar el paquete usamos:

```
# apt install mongodb mongodb-clients mongodb-server
# snap install mongo44-configurable
```

y ya podemos iniciar su uso mediante:

```
$ mongo
```

MongoDB y Python A diferencia de SQLite, no existe un módulo noSQL de Python predeterminado que pueda usar para conectarse a una base de datos MongoDB. En su lugar, deberá instalar un controlador Python noSQL para MongoDB a fin de interactuar con una base de datos MongoDB desde una aplicación Python. Uno de esos controladores es *pymongo*. Puede descargar este módulo noSQL de Python con pip:

```
$ pip install pymongo
```

Defina una función que se conecte al servidor de la base de datos MongoDB y devuelva el objeto de conexión:

```
from pymongo import MongoClient
import pymongo

def get_database():
    # Provide the mongodb atlas url to connect python to
mongodb using pymongo
```

```
        CONNECTION_STRING = "mongodb+srv:
    //<username>:<password>@<cluster-name>.mongodb.net/myFirstDatabase"
        # Create a connection using MongoClient. You can im-
  port MongoClient or use pymongo.MongoClient
        client = MongoClient(CONNECTION_STRING)
        # Create the database for our example
        return client['user_shopping_list']

    dbname = get_database()
```

Ejemplo de uso Una vez que hemos ingresado al sistema de base de datos con nuestro usuario, entonces podemos pedirle al sistema nos muestre las colecciones existentes (bases de datos), mediante:

```
    db
```

podemos crear una usando:

```
    db.createCollection(prueba)
```

podemos ver las existentes mediante:

```
    show collections
```

y le pedimos usar alguna, mediante:

```
    use prueba
```

y estamos en posibilidad de crear nuestras documentos, por ejemplo:

```
    var miProductos =
    [
        {"_id" : 1, "descripcion" : "Cigarros", "precio" : 25.5,
  "precioVenta" : 30.0},
        {"_id" : 2, "descripcion" : "Refresco", "precio" : 4.5,
  "precioVenta" : 6.0},
    ];
    db.Productos.insert(miProductos);
```

y los visualizamos:

```
db.Productos.find()
```

ahora procedemos hacer lo pertinente para nuestra otro documento, usando:

```
var miVentas =
[
    {"_id" : 1, "id_prod" : 1, "Cantidad" : 2, "Fecha" :
"2020/12/01 8:01:00"},
    {"_id" : 2, "id_prod" : 2, "Cantidad" : 5, "Fecha" :
"2020/12/01 10:15:00"},
    {"_id" : 3, "id_prod" : 2, "Cantidad" : 4, "Fecha" :
"2020/12/01 13:34:00"},
    {"_id" : 4, "id_prod" : 1, "Cantidad" : 3, "Fecha" :
"2020/12/01 21:56:00"},
];
db.Ventas.insert(miVentas);
db.Ventas.find()
```

de esta forma, ahora podemos usar Find para hacer el equivalente a un JOIN como en los ejemplos anteriores y mostrar el resultado de la solicitud, mediante:

```
db.Ventas.find().forEach(
    function (object) {
        var commonInBoth=db.Productos.findOne({ "_id":
object.id_prod} );
        if (commonInBoth != null) {
            print(object.Fecha, object.Cantidad, common-
InBoth.descripcion, commonInBoth.precioVenta, (commonInBoth.precioVenta
* object.Cantidad));
        }
    }
);
```

Exportar Base de Datos en MongoDB En MongoDB, cuando se exporta información, se puede obtener un archivo de texto legible por el ser humano con sus datos. Por defecto, la información se exporta en formato *json*, pero también se puede exportar en formato *csv*.

Para exportar información desde MongoDB, utilizaremos el comando mongoexport. Este comando nos permite exportar de forma detallada, por lo que podemos especificar una base de datos, una colección, un campo e incluso utilizar una consulta para la exportación.

Si queremos exportar debemos de ejecutar lo siguiente:

```
# mongoexport –db mibasededatos -c restaurants –out Restaurants.json
```

En el comando anterior, se utilizó –db para especificar la base de datos, -c para la colección y –out, para el archivo en el que se guardarán los datos.

Importar Base de Datos en MongoDB Importaremos los datos de Restaurante.json a una nueva base de datos llamada mibasededatos y a una colección llamada restaurantes. Utilizaremos el comando mongoimport de la siguiente forma:

```
# mongoimport –db mibasededatos –collection restaurantes –file Restaurants.json
```

Como podemos observar, la base de datos no la hemos tenido que crear, si no que al importar los datos se ha creado de forma automática.

Comprobar los datos importados Nos conectaremos a la base de datos:

```
$ mongo mibasededatos
```

Se nos habrá cambiado el prompt, por lo que estaremos conectados a la base de datos, contaremos los documentos de la colección importada:

```
> db.restaurantes.count()
```

y debe coincidir con los datos exportados. Y con esto veremos que hemos importado los mismos registros que hemos exportado.

8.5.5 Redis

El servidor de diccionarios remotos o Redis (Remote Dictionary Server) es un almacén de datos de valores clave que destaca por su gran velocidad y su facilidad de uso. Un almacén de valores clave es una base de datos NoSQL que utiliza un sistema de clave/valor para guardar la información. El proyecto se inició cuando Salvatore Sanfilippo, el desarrollador original de Redis, trataba de mejorar la escalabilidad de su empresa emergente italiana. A partir de ahí, desarrolló Redis, que ahora se utiliza como base de datos, caché, agente de mensajes y cola.

En Redis se almacenan los datos como un grupo de clave-valor, donde la clave es un identificador único y el valor puede contener cualquier tipo de dato como un texto, un archivo, un documento, o una imagen, entre otros. Redis utiliza la memoria para almacenar de forma rápida datos que pueden rescatarse de manera inmediata cuando son necesarios.

Redis ofrece tiempos de respuesta inferiores al milisegundo, lo que permite que se realicen millones de solicitudes por segundo para aplicaciones en tiempo real de la industria, como videojuegos, tecnología publicitaria, servicios financieros, sanidad e IoT. Hoy en día, Redis es uno de los motores de código abierto más populares en la actualidad, denominado la base de datos "preferida" por Stack Overflow durante cinco años consecutivos. Por su rápido rendimiento, Redis es una opción muy habitual en aplicaciones de almacenamiento en caché, administración de sesiones, videojuegos, tablas de clasificación, análisis en tiempo real, datos geoespaciales, servicios de vehículos compartidos, Chat/mensajería, Streaming de contenido multimedia y publicación/suscripción.

Para qué se Usa y sus Funcionalidades Redis tiene muchos usos debido a sus características especiales. Se trata de una gran alternativa para implementar una caché y reducir la latencia a la hora de acceder a los datos de un servidor, en salas de chat para ofrecer un alto rendimiento, en el desarrollo de videojuegos para ofrecer tablas de clasificación en tiempo real, como almacenamiento de sesiones de usuarios, como complemento en Streamings multimedia para guardar metadatos, o en Machine Learning para procesar grandes cantidades de información y ofrecerla en tiempo real.

Los principales beneficios que aporta Redis son:

- Guarda la información en memoria en lugar de en discos HDD o SSD como las bases de datos SQL.

- Cuenta con una amplia variedad de estructuras de datos que permite adaptarse a cualquier aplicación.
- El código para utilizar Redis es sencillo y simple, haciendo que sean necesarias menos líneas de código.
- Permite replicar los datos de forma asíncrona en distintos servidores, favoreciendo un mejor rendimiento, acelerando los tiempos de recuperación e incrementando el nivel de seguridad.
- Aporta un alto nivel de disponibilidad y escalabilidad.
- Es un proyecto de código abierto que se encuentra en constante evolución.

Instalación en Debian GNU/Linux de Redis Para instalar el paquete usamos:

```
# apt install redis-server
```

Una vez instalado el servicio de Redis es necesario acceder al mismo, iniciamos Redis usando:

```
$ redis-server
```

Para comprobar si Redis funciona correctamente, iniciamos la interfaz de comunicación con la base de datos:

```
$ redis-cli
```

La interfaz debería mostrar entonces la dirección IP y el puerto a través del cual se ejecuta Redis, a los que se puede enviar un ping de comprobación.

```
127.0.0.1:6397> ping
PONG
```

Si Redis responde, queda demostrado que el sistema de bases de datos se ha instalado correctamente. Ahora también se puede comprobar si se puede escribir texto.

```
127.0.0.1:6397> set test "OK!"
127.0.0.1:6397> get test
"OK!"
```

Configurar Redis Redis se instala, en un principio, con la configuración estándar, que luego puede modificarse con los comandos correspondientes.

```
127.0.0.1:6397> config get *
```

En la lista de ajustes para la configuración, las parejas de elementos se descomponen en dos posiciones, una debajo de la otra: al elemento *dbfilename* le corresponde, entonces, el valor *dump.rdb*. El asterisco que hemos usado para abrir la lista actúa de marcador de posición para un ajuste determinado de la lista. Cuando solo se quiere examinar un ajuste, el asterisco se reemplaza por el nombre del elemento, utilizando siempre para ello el que se encuentra en primera posición, que es la llave para el valor de configuración correspondiente.

```
127.0.0.1:6397> config get dbfilename
1) "dbfilename"
2) "dump.rdb"
```

Para cambiar una entrada en el archivo de configuración[137], se usa el comando *set*. Puede usarse, por ejemplo, para definir una contraseña:

```
127.0.0.1:6397> config set requirepass "password"
OK
```

Tras hacerlo, si solicitamos la contraseña con el comando get, se nos pedirá que la introduzcamos nosotros primero, al fin y al cabo, por algo le hemos puesto una. Para introducirla, usamos el comando authy, a continuación, consultamos la entrada en el archivo de configuración como acabamos de hacer.

```
127.0.0.1:6397> auth "password"
127.0.0.1:6397> config get requirepass
1) "requirepass"
2) "password"
```

[137] Existen más formas de hacer más segura tu base de datos. En la página web oficial de Redis, los desarrolladores resumen diversos puntos al respecto.

En realidad, Redis guarda todos los datos en la memoria principal. Teniendo esto en cuenta, para lograr la persistencia de los datos, se puede almacenar una copia (*snapchot*) de la base de datos en el disco duro, que se ubicará en el archivo *dump.rdb*.

```
127.0.0.1:6397> save
```

Con el comando *save*, se crea manualmente una copia, pero también pueden programarse para que se realicen de forma automática.

```
127.0.0.1:6397> save 60 10
```

En este ejemplo, hemos asignado dos parámetros al comando: ahora, se creará una copia cada 60 segundos si ya se han producido 10 cambios en dicho intervalo de tiempo.

Sin embargo, el comando *save* no se recomienda mientras el sistema esté en funcionamiento, ya que impide que los clientes accedan a la base de datos. En estos casos es más conveniente *bgsave*, que realiza el proceso en un segundo plano.

Además de la posibilidad de hacer una copia de la base de datos, también existe el modo *append only file (AOF)*, en el que Redis guarda en un archivo cada acción realizada. Gracias a este archivo, si el servidor se averiará inesperadamente, se podría averiguar qué fue lo último que se hizo. Para activar el modo AOF[138], debe modificarse el archivo de configuración.

```
127.0.0.1:6397> config set appendonly yes
```

Crear Entradas Una vez hayas configurado Redis, ya puedes trabajar con la base de datos. Dispones para ello de varios tipos distintos de datos y de comandos.

[138] Si tus datos requieren la máxima seguridad, deberías activar el modo AOF y, además, hacer copias regulares de la base de datos: así será prácticamente imposible que pierdas tus datos. Estos procesos, no obstante, ralentizan en cierta medida el funcionamiento de la base de datos.

Strings Lo más fácil es crear una String[139], es decir, una cadena o secuencia de elementos. Para ello, utiliza el comando *set*.

```
127.0.0.1:6397> set foo "bar"
127.0.0.1:6397> set value 1
```

Si se solicitan ahora las entradas foo y value mediante el comando *get*, se mostrarán los valores correspondientes.

```
127.0.0.1:6397> get foo
"bar"
127.0.0.1:6397> get value
"1"
```

El comando para borrar una entrada es *del*.

```
127.0.0.1:6397> del foo
(integer) 1
127.0.0.1:6397> get foo
(nil)
```

Si no quieres crear muchas entradas usando una fila nueva cada vez, puedes usar la función avanzada *mset*. Para solicitar los valores de varios campos a la vez, también existe el comando *mget*.

```
127.0.0.1:6397> mset foo1 "bar1" foo2 "bar2" foo3 "bar3"
OK
127.0.0.1:6397> mget foo1 foo2 foo3
1) "bar1"
2) "bar2"
3) "bar3"
```

[139] No importa si se introducen los valores entre comillas o no. Para hacer el código más legible, puede ponerse el texto entre comillas y los valores numéricos sin ellas.

Listas Con Redis se pueden usar, además, otros tipos de datos. Algunos de los más populares para trabajar con la base de datos son, por ejemplo, las listas y los sets. Ambos son conjuntos de valores, pero, mientras que los sets no tienen un orden concreto, los valores de las listas están numerados. En una lista se pueden añadir, solicitar o borrar entradas.

```
127.0.0.1:6397> lpush mylist foo
(integer) 1
127.0.0.1:6397> lpush mylist bar
(integer) 2
127.0.0.1:6397> lrange mylist 0 10
1) "foo"
2) "bar"
127.0.0.1:6397> linsert mylist before "bar" "test"
(integer) 3
127.0.0.1:6397> lrange mylist 0 10
1) "foo"
2) "test"
3) "bar"
127.0.0.1:6397> lrem mylist 0 foo
(integer) 1
127.0.0.1:6397> lrange mylist 0 10
1) "test"
2) "bar"
```

En este ejemplo hemos añadido en primer lugar dos elementos a una lista (*lpush*) y luego hemos solicitado que se muestren. Con el comando *lrange* se indica qué segmento debe mostrarse (aquí del 0 al 10, pero pueden usarse también números negativos). A continuación, mediante el comando *linsert* hemos añadido un valor nuevo delante de uno que ya existía (también podría usarse *after*), con lo cual hemos cambiado la numeración. El comando *lrem* permite borrar de la lista entradas con un valor específico.

Sets Para los sets, Redis utiliza otros comandos, pero con resultados muy similares:

```
127.0.0.1:6397> sadd myset "foo"
(integer) 1
```

```
127.0.0.1:6397> sadd myset "bar"
(integer) 1
127.0.0.1:6397> smembers myset
1) "bar"
2) "foo"
127.0.0.1:6397> sismember myset "bar"
(integer) 1
127.0.0.1:6397> srem myset "bar"
(integer) 1
127.0.0.1:6397> smembers myset
1) "foo"
```

Con el comando *sadd* también se pueden integrar varios elementos en el set si se introducen en el comando uno detrás de otro. Para visualizar el set, basta con usar el comando *smembers* y el nombre del set en cuestión. El comando *sismember* permite, además, buscar una entrada concreta. De manera análoga a la lista, con *srem* se pueden borrar entradas sueltas.

Sin embargo, Redis también ofrece a los usuarios la posibilidad de utilizar sets en un formato ordenado.

```
127.0.0.1:6397> zadd mysortedset 1 "foo"
(integer) 1
127.0.0.1:6397> zadd mysortedset 2 "bar"
(integer) 1
127.0.0.1:6397> zadd mysortedset 2 "foobar"
(integer) 1
127.0.0.1:6397> zrange mysortedset 0 10
1) "foo"
2) "bar"
3) "foobar"
```

Para añadir elementos se utiliza, en este caso, el comando *zadd* y un *score* o puntaje. Mientras que los valores propiamente dichos no pueden aparecer más de una vez, con un *score* se puede indicar un mismo valor varias veces. El score no es, por lo tanto, una numeración directa dentro del set, sino una ponderación, de manera que todas las entradas con el puntaje o score2 aparecerán tras los que tengan el score1. Con el comando *zrange* se pueden visualizar todos los elementos o los que se seleccionen.

Hashes Un tipo especial de datos son los hashes: entradas individuales compuestas de varios valores, de manera similar a los sets y las listas, pero en los que cada valor va acompañado de una clave, formando así los llamados pares clave-valor o key-value.

```
    127.0.0.1:6397> hset user1 name "bob" email "bob@example.com"
password "rK87_x"
    OK
    127.0.0.1:6397> hget user1 name
    1) "bob"
    127.0.0.1:6397> hgetall user1
    1) "name"
    2) "bob"
    3) "email"
    4) "bob@example.com"
    5) "password"
    6) "rK87_x"
    127.0.0.1:6397> hvals user1
    1) "bob"
    2) "bob@example.com"
    3) "rK87_x"
    127.0.0.1:6397> hkeys user1
    1) "name"
    2) "email"
    3) "password"
    > hdel user1 password
    (integer) 1
    127.0.0.1:6397> hgetall user1
    1) "name"
    2) "bob"
    3) "email"
    4) "bob@example.com"
    127.0.0.1:6397> del user1
    (integer) 1
    127.0.0.1:6397> hgetall user1
    (empty list or set)
```

En este ejemplo, hemos usado *hset* para crear un hash con el nombre user1 y tres campos. Mediante el comando *hget* podemos solicitar el valor de cada campo. Para que se muestren todos, se puede usar *hgetall*. Otras opciones para visualizar valores son *hvals* (muestra todos los valores guardados en el hash) y *hkeys* (muestra todas las claves guardadas en el hash). Con *hdel* se pueden borrar valores sueltos, mientras que con *del*, como ya hemos visto, se borra el hash entero[140].

Otras opciones Naturalmente, con Redis no solo se pueden crear entradas en una base de datos, sino que también pueden asignarse propiedades concretas a los datos. En este sentido, pueden resultar muy útiles, por ejemplo, los comandos de incremento y decremento.

```
127.0.0.1:6397> set foo 1
OK
127.0.0.1:6397> get foo
"1"
127.0.0.1:6397> incr foo
(integer) 2
127.0.0.1:6397> incr foo
(integer) 3
127.0.0.1:6397> get foo
"3"
127.0.0.1:6397> decr foo
(integer) 2
127.0.0.1:6397> get foo
"2"
```

Con la ayuda de estas funciones, se pueden incrementar o reducir los valores en una unidad. A veces, en cambio, se quieren introducir valores que solo permanezcan en la base de datos durante cierto tiempo: para ello existe la función expire.

```
127.0.0.1:6397> set foo "bar"
OK
127.0.0.1:6397> expire foo 100
(integer) 1
```

[140] El comando flushall sirve para borrar todas las entradas de la base de datos.

```
127.0.0.1:6397> ttl foo
(integer) 50
127.0.0.1:6397> ttl foo
(integer) -50
127.0.0.1:6397> get foo
(nil)
```

El comando expire requiere una indicación de tiempo en segundos. En este ejemplo, hemos decidido que la entrada debe durar 100 segundos. Una vez transcurrida la mitad del tiempo, hemos usado el comando *ttl* para solicitar el time-to-live, es decir, el tiempo restante. Si esperamos aún más, el TTL pasará a ser negativo a partir del momento en el que la entrada ya haya desaparecido.

El comando *setex* permite asignar un TTL a una entrada de la base de datos ya desde su creación.

```
127.0.0.1:6397> setex foo 100 "bar"
OK
```

Una vez se ha creado una entrada, esta se puede ampliar: el comando *append* añade otro valor al que ya existía.

```
127.0.0.1:6397> set foo "Hello"
OK
127.0.0.1:6397> append foo " World"
(integer) 11
127.0.0.1:6397> get foo
"Hello World"
127.0.0.1:6397> set bar 5
OK
127.0.0.1:6397> append bar 10
(integer) 3
127.0.0.1:6397> get bar
"510"
```

Como se puede ver, al solicitar los valores correspondientes, aparecen los nuevos elementos simplemente tras los que ya estaban. Si la entrada en cuestión aún no existe, *append* realiza entonces la misma función que *set*.

Además, también se puede cambiar el nombre de las entradas usando el comando *rename*.

```
127.0.0.1:6397> set foo 100
OK
127.0.0.1:6397> rename foo bar
OK
127.0.0.1:6397> get foo
(nil)
127.0.0.1:6397> get bar
"100"
```

Existen muchos otros comandos para trabajar con Redis. En la descripción oficial se pueden consultar los detalles de todos los comandos disponibles.

8.6 LAMP y LEMP

LAMP y LEMP son acrónimos usados para describir un sistema de infraestructura de internet que usa las siguientes herramientas:

Para LAMP:

- Linux, el sistema operativo
- Apache, el servidor de Web
- MySQL/MariaDB, el gestor de bases de datos
- PHP, el lenguaje de programación

Para LEMP

- Linux, el sistema operativo
- Nginx, el servidor de Web
- MySQL/MariaDB, el gestor de bases de datos
- PHP (Hypertext Preprocessor), el lenguaje de programación

La combinación de estas tecnologías es usada principalmente para definir la infraestructura de un servidor Web, utilizando un paradigma de programación para el desarrollo del sistema. A pesar de que el origen de estos programas de código abierto no fue específicamente diseñado para trabajar entre sí, la combinación se popularizó debido al bajo costo de adquisición y ubicuidad de sus componentes.

Para instalar apache2 usamos:

```
# apt install apache2 libapache2-mod-evasive apachetop
```

Para instalar Nginx usamos:

```
# apt install nginx
```

En este caso tenemos dos opciones, instalar MariaDB que ya es parte integral de múltiples aplicaciones dentro de GNU/Linux o usar MySQL, en cada caso el uso es prácticamente el mismo.

Para instalar el paquete MariaDB usamos:

```
# apt install mariadb-server mariadb-client
```

y para configurarlo, usamos:

```
# mysql
> use mysql
> update user set plugin=" where user='root';
> alter user root@localhost identified by 'XXXXXXX';
> flush privilegies;
> exit
```

y para usarlo

```
$ mysql -u root -p
```

Para instalar el paquete MySQL usamos:

```
# apt install mysql-common mysql-client mysql-server mytop
mysql-admin
```

y para configurarlo (las bases de datos se guardan en /var/lib/mysql), usamos:

```
# mysql_secure_installation
```

o

```
# mysql
```

en caso de no crear contraseña, podemos ingresar así:

```
$ mysql -u root -p
```

una vez en la consola del cliente de MySQL creamos un nuevo usuario

```
> create user nombre identified by 'XXXXX';
```

y a continuación le damos privilegios totales

```
> gran all privileges on *.* to nombre with grant option;
> flush privileges;
> exit;
```

si requerimos las pruebas de eficiencia, podemos instalar:

```
# apt install ferret mysql-workbench mysql-workbench-data
mycli
```

Para instalar la interface para PHP, usamos:

```
# apt install php libapache2-mod-php php-mysql php-gd ph-
pmyadmin
```

8.7 Lenguajes de Programación y Bases de Datos SQL

Quizás te estés preguntando, ¿por qué debería preocuparme por conectar un lenguaje de programación como Java o Python y una base de datos SQL?. Hay muchos casos de uso en los que alguien querría conectar Java o Python a una base de datos SQL. Es posible que se esté trabajando en una aplicación Web. En este caso, necesitaríamos conectar una base de datos SQL para poder almacenar los datos provenientes de la aplicación Web. Quizás estemos trabajando en ingeniería de datos y necesitemos crear una canalización ETL automatizada. Conectar Java o Python a una base de datos SQL nos permitirá utilizar las capacidades de automatización del lenguaje de programación. También podremos comunicar entre diferentes fuentes de datos, no tendremos que cambiar entre diferentes lenguajes de programación.

Conectar Java o Python y una base de datos SQL también hará que nuestro trabajo de ciencia de datos sea más conveniente. Podremos utilizar nuestras habilidades en el lenguaje de programación para manipular datos de una base de datos SQL. No necesitaremos un archivo *CSV*.

Bases de Datos Relacionales las bases de datos de Software libre más utilizadas son PostgreSQL, MariaDB y MySQL que rivalizan con la base de datos Oracle comercial. PostgreSQL es adecuada incluso para organizaciones y proyectos de gran tamaño.

Otras bases de datos relevantes son H2 una base de datos implementada en Java con características avanzadas que es posible utilizar para los tests de integración al no requerir de un servidor y ser posible ejecutarla en memoria. Para realizar las pruebas de integración utilizando la misma base de datos que en producción es posible utilizar *Testcontainers* que utiliza Docker para iniciar una instancia de la base de datos en un contenedor. La conexión a la base de datos por seguridad requiere de un usuario y contraseña que la aplicación ha de conocer, para aún mayor seguridad es posible generar las credenciales de conexión a la base de datos de forma dinámica por *Vault* en una aplicación de *Spring*.

El problema de Seguridad de SQL Injection es un grave problema de seguridad que afecta a las aplicaciones que construyen sentencias de forma dinámica a partir de datos provenientes de origen no confiable. Un origen no confiable es cualquier dato proveniente de forma externa a la aplicación, en el caso de las aplicaciones Web o servicios *REST* es un parámetro de la petición o un dato de un *JSON*.

El SQL injection es un grave problema de seguridad ya que permite al atacante tener acceso a datos de la base de datos, obtener acceso con una cuenta de otro usuario o realizar operaciones sobre los datos de forma no autorizada. El problema se produce en la construcción de forma dinámica de la sentencia SQL mediante la concatenación de cadenas y datos de origen no confiable.

8.7.1 Java y el Acceso a Base de Datos Relacionales

Java ofrece soporte para las bases de datos relacionales desde prácticamente las primeras versiones del JDK hasta día de hoy incorporando un conjunto de clases en el paquete *java.sql* en la denominada *API* en Java de *Java Database Connectivity* o *JDBC*.

Las principales clases de la *API* de *JDBC* son las clases *Statement* y *PreparedStatement* que representan la sentencia SQL ya sea de inserción, actualización, eliminación o consulta así como las sentencias *DDL* para la

creación de tablas, índices, campos o procedimientos almacenados. Normalmente se utiliza la clase *PreparedStatement* ya que tiene beneficios en cuanto a rendimiento al ejecutarse de forma repetida y utilizada correctamente permite evitar el grave problema de seguridad de SQL injection común en las aplicaciones que construyen de forma dinámica con concatenaciones sentencias SQL utilizando datos procedentes de fuentes no confiables como parámetros de una petición *HTTP*, de *JSON* u otras fuentes externas a la aplicación.

La clase *ResultSet* es la clase que proporciona el acceso a los datos cuando se ejecuta una sentencia SQL de consulta, la clase se itera en los resultados y se obtienen los datos según los nombres o índices asignados en la consulta para las columnas.

La clase *Connection* representa una conexión a una base de datos, hay que crear una conexión ya que las bases de datos trabajan con una arquitectura de cliente y servidor, la aplicación actúa de cliente y la base de datos actúa de servidor. A través de la clase *Connection* se inicia una transacción y finaliza con el *commit* o el *rollback*. Para crear la conexión de la base de datos se proporcionan las credenciales de usuario y contraseña.

Las bases de datos relacionales soportan transacciones para proporcionar las propiedades *ACID* para lo que se utilizan las transacciones. Atomicidad donde un grupo de operaciones individuales se ejecutan todas o ninguna, consistencia mediante la cual los cambios son válidos según las reglas incluyendo restricciones, cambios en cascada, y disparadores, aislamiento donde los cambios de una transacción no se ven afectados por los cambios realizados en otras transacciones concurrentes y finalmente durabilidad que garantiza que en caso de completarse la transacción perdura en el tiempo aún cuando el sistema sufra un fallo posterior.

Crear una conexión a una base de datos es costoso, para evitar incurrir en este tiempo de creación y destrucción de conexiones o limitar el número de conexiones que una aplicación utiliza como máximo las aplicaciones utilizan un pool de conexiones. Las conexiones se crean al iniciar la aplicación o bajo demanda según se van necesitando más hasta el límite máximo definido. Cuando la aplicación necesita una conexión la obtiene de forma rápida del pool de conexiones y cuando termina de utilizarla la devuelve al pool de conexiones para que sea reutilizada en posteriores usos.

Cada base de datos utiliza un protocolo diferente de comunicación con los clientes por lo que es necesario un componente que abstrae de las peculiaridades de cada base de datos y proporcione un marco común de trabajo

independiente de cada base de datos. Cada base de datos requiere de un *Driver* compatible también con la versión de la base de datos. Generalmente, son los desarrolladores de la propia base de datos los que proporcionan un *Driver* específico adecuado para el acceso a la base de datos desde Java que cumple con las *APIs* de *JDBC*.

Ejemplo de Conexión y Consulta a un Base de Datos Relacional con la API de Java En este ejemplo de código se muestra el uso de las clases fundamentales de Java para usar una base de datos relacional. El primer paso es establecer una conexión con la base de datos, en este caso usando la base de datos H2 en memoria.

Posteriormente se ejecuta una sentencia *DDL* para crear una tabla, se insertan varias filas con la sentencia *insert* y se obtienen los datos de la tabla con una sentencia *select*. Después de las inserciones se realiza un commit que completa una transacción en la base de datos, si en vez del *commit* se hiciese un *rollback* al obtener los datos con la consulta posterior la tabla aparecería vacía.

Las clases *Connection*, *Statemente*, *PreparedStatement* y *ResultSet* al finalizar su uso hay que invocar su método *close* para liberar los recursos que tienen reservados, especialmente en el caso de las conexiones ya que son un recurso limitado. Estas clases implementan la interfaz *AutoCloseable* con lo que son adecuadas para las sentencias *try-with-resources* de Java.

Establecer la Conexión a la Base de Datos por defecto después de cada sentencia Java emite un *commit*, esto no es lo deseado en el caso de querer agrupar la ejecución de varias sentencias en una transacción, para evitarlo hay que usar la opción *setAutoCommit* a *false*.

```
package io.github.picodotdev.blgbitix.javasql;

import java.math.BigDecimal;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.Statement;

import java.text.DecimalFormat;

import java.util.Locale;

public class Main {
```

```
        public static void main(String[] args) {
            DriverManager.drivers().forEach(d -> {
                System.out.printf("Driver: %s%n", d.getClass().getName());
            });
            try (Connection connection = DriverManager.getConnection("jdbc:h2:mem:database",
"sa", "")) {
                connection.setAutoCommit(false);
            ...
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
```

Ejecutar una Sentencia con Statement

```
    Statement statement = connection.createStatement();
    statement.execute("CREATE TABLE product(id INT IDENTITY NOT NULL PRIMARY KEY,
name VARCHAR(255), price DECIMAL(20, 2))");
```

Ejecutar una Sentencia con PreparedStatement

```
    PreparedStatement preparedStatement = connection.prepareStatement("INSERT INTO product
(name, price) values (?, ?)", new String[] { "id" });
    preparedStatement.setString(1, "PlayStation 5");
    preparedStatement.setBigDecimal(2, new BigDecimal("499.99"));
    preparedStatement.executeUpdate();
    ResultSet resultSet1 = preparedStatement.getGeneratedKeys();
    while (resultSet1.next()) {
        System.out.printf("Primary key: %s%n", resultSet1.getLong(1));
    }
    resultSet1.close();
    preparedStatement.setString(1, "Xbox Series X");
    preparedStatement.setBigDecimal(2, new BigDecimal("499.99"));
    preparedStatement.executeUpdate();
    ResultSet resultSet2 = preparedStatement.getGeneratedKeys();
    while (resultSet2.next()) {
        System.out.printf("Primary key: %s%n", resultSet2.getLong(1));
```

```
        }
        resultSet2.close();
        connection.commit();
```

Ejecutar una Consulta

```
        PreparedStatement preparedStatement = connection.prepareStatement("SELECT id, name, price
    FROM product");
        ResultSet resultSet = preparedStatement.executeQuery();
        while (resultSet.next()) {
            System.out.printf("Product (id: %s, name: %s, price: %s)%n", resultSet.getLong(1), result-
    Set.getString(2),
            DecimalFormat.getCurrencyInstance(new Locale("es", "ES")).format(resultSet.getBigDecimal(3)));
        }
        resultSet.close();
```

Cerrar la Conexión de Forma Explícita

```
        connection.close();
```

Resultado el resultado del programa en la terminal es el siguiente.

```
        Driver: org.h2.Driver
        Primary key: 1
        Primary key: 2
        Product (id: 1, name: PlayStation 5, price: 499,99 EUR)
        Product (id: 2, name: Xbox Series X, price: 499,99 EUR)
```

Dependencia con el Driver de la Base de Datos en el archivo de construcción hay que añadir la dependencia que contiene el Driver para la base de datos a conectarse.

```
        plugins {
            id 'application'
        }
        repositories {
            mavenCentral()
        }
```

```
dependencies {
    implementation 'com.h2database:h2:1.4.200'
}
application {
    mainClass = 'io.github.picodotdev.blgbitix.javasql.Main'
}
```

El problema de Seguridad de SQL Injection Las siguientes sentencias SQL sufren del problema de SQL injection, en la primera un atacante puede obtener un dato de cualquier columna de la tabla y con la segunda ejecutar una sentencia en este caso para eliminar todas las filas de cualquier tabla.

En la primera cambiando el valor de column se obtiene el dato de cualquier columna de la tabla, por ejemplo el campo password.

```
"SELECT id, " + column + " FROM users WHERE user_id = " + userId
"SELECT id, password FROM users WHERE user_id = 1
```

En esta sentencia se finaliza la sentencia original con ; y se inicia otra lo que provoca la eliminación de una tabla, con un valor para user_id especialmente construido para ejecutar la sentencia maliciosa de eliminación de la tabla.

```
"SELECT * FROM users WHERE user_id = " + userId
"SELECT * FROM users WHERE user_id = 105; DROP TABLE users;"
```

La solución al problema de seguridad de SQL *injection* en Java es no construir la sentencia de forma dinámica mediante concatenación de cadenas utilizando la clase *PreparedStatement* con argumentos para los datos que se inserten en la sentencia SQL.

```
    PreparedStatement preparedStatement = connection.prepareStatement("SELECT * FROM users
WHERE user_id = ?"});
    preparedStatement.setInt(1, 1);
    preparedStatement.executeUpdate();
```

Librerías de Persistencia en Java Habitualmente no se utilizan directamente las clases de la API de Java sino que se utilizan otras librerías de más alto nivel. Una de las más conocidas es *Hibernate*, es un ORM que proporciona acceso a los datos con una correspondencia entre el modelo relacional de las bases de datos y el modelo de objetos de Java. La aplicación trabaja con objetos y relaciones entre los objetos e *Hibernate* se encarga de transformar esos objetos en el modelo relacional de la base de datos, la aplicación no ejecuta sentencias SQL de forma directa sino que es *Hibernate* el encargado de emitir las sentencias adecuadas según los cambios realizados en los datos. Es la implementación más utilizada de ORM en Java para la especificación JPA.

- Aplicación de ejemplo con Hibernate, jOOQ y Liquibase

Spring Data es una capa de abstracción para el acceso a datos ya sean de un modelo relacional, de bases de datos NoSQL o algunos otros sistemas de datos. Spring Data hace más sencillo el acceso a los datos utilizando de forma subyacente JDBC o JPA. Spring Data proporciona algunas clases e interfaces que la aplicación implementa.

jOOQ también es otra librería de acceso a bases de datos relacionales, proporciona DSL para la construcción de sentencias SQL mediante un API Java. A diferencia de JPA con su lenguaje JPQL, jOOQ soporta características avanzadas del lenguaje SQL como windows functions. Otra ventaja de jOOQ es que al utilizar su DSL el compilador de Java realiza validación de tipos y comprobaciones en la sintaxis de la construcción de la SQL.

- Alternativa a Hibernate u ORM y ejemplo de jOOQ

Algunas aplicaciones combinan el uso de varias de estas librerías en la misma aplicación según el caso, por ejemplo utilizando Hibernate para el modelo de escritura y jOOQ o Spring Data para el modelo de lectura. También es posible utilizar jOOQ para generar las sentencias SQL y posteriormente ejecutarlas con Hibernate o Spring Data.

Liquibase es otra librería de utilidad que permite lanzar scripts de migración con sentencias SQL para realizar cambios en la base de datos como modificar el esquema de tablas, insertar, actualizar o eliminar datos. Es necesario si los cambios en el código en una nueva versión de la aplicación requiere cambios en el esquema de la base de datos. Estas no son las únicas librerías existentes pero sí son de las más conocidas y utilizadas.

8.7.2 Python y el Acceso a Base de Datos Relacionales

Las bases de datos de Python y SQL se conectan a través de bibliotecas personalizadas de Python. Podemos importar estas bibliotecas en su secuencia de comandos de Python. Las bibliotecas de Python específicas de la base de datos sirven como instrucciones complementarias. Estas instrucciones guían a nuestra computadora sobre cómo puede interactuar con un base de datos SQL.

Cómo Configurar el Proyecto Tomemos una base de datos PostgreSQL, *AWS Redshift*, por ejemplo. Primero, necesitamos importar la biblioteca *psycopg*. Es una biblioteca de Python universal para bases de datos de por ejemplo PostgreSQL:

```
#Library for connecting to AWS Redshift
import psycopg
#Library for reading the config file, which is in JSON
import json
#Data manipulation library
import pandas as pd
```

Notarás que también importamos las bibliotecas *JSON* y *pandas*. Importamos *JSON* porque crear un archivo de configuración *JSON* es una forma segura de almacenar las credenciales de su base de datos. ¡No queremos que nadie más los mire!.

La biblioteca de *pandas* nos permitirá utilizar todas las capacidades estadísticas de *pandas* para su secuencia de comandos de Python. En este caso, la biblioteca permitirá que Python almacene los datos que devuelve su consulta SQL en un marco de datos.

A continuación, necesitamos acceder al archivo de configuración. La función *json.load()* lee el archivo *JSON* para que pueda acceder a las credenciales de la base de datos en el siguiente paso:

```
config_file = open(r"C:\Users\yourname\config.json")
config = json.load(config_file)
```

Ahora que la secuencia de comandos de Python puede acceder a el archivo de configuración *JSON*, queremos crear una conexión de base de datos. Deberá leer y usar las credenciales de su archivo de configuración:

```
    con = psycopg2.connect(dbname= "db_name", host=config[hostname], port = config["port"],user=config["user_id"],
password=config["password_key"])
    cur = con.cursor()
```

¡Acabamos de crear una conexión a la base de datos! Cuando importamos la biblioteca *psycopg*, tradujo el código Python que escribió anteriormente para hablar con la base de datos PostgreSQL (AWS Redshift).

En sí mismo, AWS Redshift no entendería el código anterior. Pero debido a que importó la biblioteca *psycopg*, ahora habla un idioma que AWS Redshift puede entender. Lo bueno de Python es que tiene bibliotecas para SQLite, MySQL y PostgreSQL. Podremos integrar las tecnologías con facilidad.

Cómo Escribir una Consulta SQL siéntase libre de descargar European Soccer Data a su base de datos PostgreSQL. Usaré sus datos para este ejemplo.

La conexión de base de datos que creó en el último paso le permite escribir SQL para luego almacenar los datos en una estructura de datos compatible con Python. Ahora que hemos establecido una conexión a la base de datos, podemos escribir una consulta SQL para comenzar a extraer datos:

```
query = "SELECT *
FROM League
JOIN Country ON Country.id = League.country_id;"
```

Sin embargo, el trabajo aún no está terminado. Necesitamos escribir código Python adicional que ejecute la consulta SQL:

```
#Runs your SQL query
execute1 = cur.execute(query)
result = cur.fetchall()
```

Luego, debemos almacenar los datos devueltos en un marco de datos de pandas:

```
#Create initial dataframe from SQL data
raw_initial_df = pd.read_sql_query(query, con)
print(raw_initial_df)
```

Ahora deberíamos obtener un marco de datos de pandas (raw_initial_df).

8.8 Seguridad en Bases de Datos

Las fugas de datos son extremadamente prominentes en el mundo cibernético debido a la falta de una implementación de seguridad adecuada. Proteger las bases de datos es una práctica esencial para garantizar que se eviten consecuencias como la pérdida de datos incluso por acceso no autorizado o el tiempo de inactividad del sistema.

Hay muchos desafíos cuando se trata de la seguridad de la base de datos, ya que cuanto más accesible es una base de datos, menos segura es. Algunos de los desafíos de seguridad comunes cuando se trata de la seguridad de la base de datos incluyen errores humanos, Malware, seguridad de la ubicación física y vulnerabilidades de Software. Sin embargo, hay muchos pasos que se pueden tomar para garantizar que una base de datos esté protegida y, por lo tanto, evitar cualquiera de los problemas mencionados anteriormente. Además, el cifrado de la base de datos es un paso esencial para proteger todos los datos.

El Software libre tiene una variedad de herramientas que pueden usarse para garantizar la seguridad de la base de datos. Brevemente presentaremos algunas de estas herramientas que se pueden usar para la evaluación de bases de datos y otras que ayudarán a proteger estas bases de datos.

Prácticas Recomendadas para Bases de Datos Algunas de las amenazas de seguridad comunes incluyen ataques DDoS, ataques de inyección SQL, ataques Man in the Middle, contraseñas débiles, corrupción de datos, condición de carrera, mala gestión del acceso a la cuenta, así como vulnerabilidades ocasionales. Afortunadamente, hay muchas prácticas que se pueden adoptar para poder evitar estos problemas de seguridad.

8.8.1 Cifrado

El cifrado es una de las prácticas más efectivas e importantes que siempre debe usarse al almacenar datos. Algunas bases de datos pueden cifrar los datos directamente o cifrar el contenedor en el que se encuentran. Para MySQL, algunas de las herramientas que se pueden usar para el cifrado son el cifrado asimétrico, el cifrado simétrico, la generación de claves públicas/privadas, las firmas digitales y los datos transparentes. En cuanto a MariaDB, hay cifrado de datos en reposo, cifrado de datos en tránsito, certificados TLS/SSL y Cipher Block Chaining.

Además, para garantizar la seguridad de la base de datos, otras prácticas útiles a tener en cuenta son el acceso condicional y la auditoría. Para el acceso condicional, siempre es útil aplicar el privilegio mínimo. Es muy importante otorgar sólo la menor cantidad de privilegios necesarios para que un usuario pueda completar el trabajo. En cuanto a la auditoría, existe una variedad de herramientas que se pueden usar para auditar bases de datos como ClusterControl y Cloud SQL para MySQL, mientras que MariaDB tiene su propio complemento de auditoría que se puede usar.

8.8.2 Restringir el Acceso y Personalizar la Configuración Predeterminada

Los próximos pasos a seguir para proteger una base de datos MySQL deben incluir la eliminación de cuentas predeterminadas, asignaciones de puertos y la personalización de la configuración predeterminada. Es muy importante eliminar la base de datos de prueba ya que todos los usuarios tienen acceso completo a ella.

El acceso remoto también debe estar restringido. El acceso a la red solo debe permitir el mínimo requerido, como es el caso de muchas prácticas de seguridad, incluido el enfoque de privilegio mínimo mencionado. Todo acceso remoto debe ser monitoreado y controlado.

8.8.3 Asegúrese de que los Servidores sean Físicamente Seguros

Un paso importante que con frecuencia se ignora es asegurarse de que los servidores estén físicamente seguros. Si está utilizando proveedores de la nube como AWS, GCP o Azure, esto está cubierto automáticamente.

Uno de los pasos más esenciales para asegurar cualquier cosa, no solo las bases de datos, es garantizar el uso de credenciales de autenticación sólidas. Las contraseñas débiles pueden ser fácilmente forzadas otorgando a los atacantes acceso a los servidores. Por lo tanto, debemos asegurarnos de que solo se utilicen contraseñas seguras que contengan una combinación de caracteres en mayúsculas y minúsculas, números y caracteres especiales.

Además, authn y authz, que se refieren a autenticación y autorización, nos permiten controlar el acceso de los usuarios a los elementos dentro de la base de datos.

8.8.4 Usar Herramientas de Evaluación de la Seguridad de la Base de Datos

Un paso fundamental para proteger una base de datos es utilizar herramientas de evaluación de la seguridad de la base de datos. Estas herramientas revisan el entorno de una base de datos y reconocen las amenazas que están presentes en el entorno. Además de resaltar las vulnerabilidades existentes presentes en el entorno, estas herramientas también evalúan las medidas de seguridad implementadas para verificar su efectividad. Presentamos cinco de las mejores herramientas de evaluación de seguridad de bases de datos de Linux para garantizar que tenga una base de datos segura.

SQLMap es una herramienta de inyección SQL de código abierto basada en Python. Esta es una gran herramienta para usar, ya que ofrece una variedad de características. Estas características incluyen la detección y explotación de diferentes tipos de ataques de inyección, Hash de contraseñas de fuerza bruta, carga o descarga de archivos de la base de datos, se pueden usar para conectarse directamente a la base de datos a través de la inyección de SQL y se pueden personalizar.

SQLNinja es una herramienta basada en Perl que se puede utilizar con aplicaciones que utilizan servidores Microsoft SQL como Backend para explotar las vulnerabilidades de inyección de SQL. Las características de SQLNinja incluyen la capacidad de tomar huellas dactilares de servidores SQL remotos, extracción de datos, carga de ejecutables, explotación de vulnerabilidades de inyección para obtener acceso a la base de datos, contraseñas de fuerza bruta mediante ataques de diccionario y ataques de escalada de privilegios si la fuerza bruta tiene éxito.

BBQSQL como SQLMap, es una herramienta de código abierto basada en Python. BBQSQL se usa para explotar las vulnerabilidades de inyección de SQL y se enfoca en la vulnerabilidad ciega de SQL. Esta herramienta requiere cierta información para ser utilizada, que incluye URL, método HTTP, Cookies, etc. Sin embargo, es una gran herramienta ya que es muy rápida para encontrar y explotar vulnerabilidades, es personalizable, realiza la validación de entrada en todas las opciones de configuración e incluso puede parchear las vulnerabilidades detectadas

JSQL Injection como su nombre lo indica, también es una herramienta de inyección SQL basada en Java. Se puede utilizar para encontrar y explotar vulnerabilidades de inyección. Se puede utilizar en 33 motores de bases de datos, realizar varios tipos de ataques de inyección, crear un Shell Web y un Shell SQL en un servidor remoto, utilizar Hashes de fuerza bruta y mucho más.

OScanner es un marco de evaluación de Oracle basado en Java. Se puede utilizar para enumeración de SID, contraseñas de fuerza bruta, versión de enumeración, roles de cuenta, privilegios y políticas de contraseña, información de auditoría y enlaces a bases de datos. OScanner da los resultados en un formato de árbol java gráfico.

Estas herramientas que hemos mencionado son fundamentales para garantizar que su base de datos esté segura. Se pueden usar para verificar qué pasos se deben tomar para garantizar que las bases de datos sean seguras, y también se pueden usar después de tomar estos pasos para evaluar la efectividad del procedimiento seguido.

8.8.5 Pruebas de Penetración

Cuando se trabaja con bases de datos que envían y reciben peticiones desde páginas Web o aplicaciones en internet, es necesario hacer pruebas de penetración (o Pentest) que nos permiten analizar los riesgos de los programas desarrollados y del sistema completo antes de entrar en la fase de producción.

Hay tres tipos de pruebas de penetración:

- Prueba de caja negra (Blackbox): Tratamos de atacar el sistema sin el conocimiento de la infraestructura y el funcionamiento del mismo. Esta es la forma menos efectiva de las pruebas Pentest.

- Pruebas de caja gris (Greybox): Tratamos de atacar el sistema con poco conocimiento de la infraestructura y el funcionamiento interno del mismo. Las pruebas de caja gris son más efectivas que las de caja negra.

- Pruebas de caja blanca (Whitebox): Tratamos de atacar el sistema con toda la información sobre la infraestructura y código fuente. Esta es la forma más efectiva de las pruebas Pentest.

Las pruebas de penetración tienen como objetivo prevenir ciberataques. ¿Qué es un ataque cibernético?

- Ataque cibernético: Es el que se ejecuta a través del espacio cibernético con el objetivo de dislocar, dañar o apropiarse de la gestión de la infraestructura de una empresa o dependencia gubernamental.
- Superficie del ataque cibernético: Es el ámbito en el cual se ataca a un sistema de cómputo. Esto puede ser muy amplio y abarca las irrupciones en las redes, los protocolos, el sistema operativo a las aplicaciones.
- Exploit: Es un ataque contra los elementos vulnerables de un sistema llevado a cabo mediante el uso de Software, un comando o una metodología específica. Un Exploit puede ser un elemento Malware; para el Hackeo ético un Exploit es una forma de identificar vulnerabilidades del sistema.

Debido a que se practicará con aplicaciones inseguras, es importante crear un ambiente seguro, pero siempre es deseable hacer pruebas con toda la infraestructura computacional disponible sobre la que se montará el sistema desarrollado, pero en muchas ocasiones esto no es posible o viable, por ello se opta por generar un sistema de prueba aislado y seguro usando máquinas virtuales[141] con la misma configuración que el sistema de interés.

Dependiendo del tamaño del sistema, es posible usar uno o más equipos anfitriones (Hosts) que alberguen nuestras máquinas virtuales y usar la red física o virtual para interconectar las aplicaciones que se montarán en las máquinas virtuales para implementar el sistema de pruebas de penetración.

Por ejemplo, si todo nuestro sistema desarrollado cabe en un solo equipo de cómputo (no importa que sistema operativo tenga el anfitrión que usemos para las pruebas), entonces podríamos usar una primer máquina virtual[142] para instalar el ambiente de pruebas como por ejemplo Kali Linux[143], dentro

[141] Entendamos por una máquina virtual a un programa de cómputo que simula a una computadora, en la cual se puede instalar y usar otros sistemas operativos de forma simultánea como si fuese una computadora real sobre nuestro sistema operativo huésped.

[142] Existen una gran variedad de manejadores de máquinas virtuales como: QEMU/KVM, VirtualBox, Vmware, Parallels, Windows Virtual PC, Gnome Box, Xen, LXC, Docker, etc.

[143] Kali Linux (https://www.kali.org), es una distribución para pruebas de penetración estándar de la industria. Es una de las distribuciones más populares entre Pentesters,

de esta instalación instalamos otro manejador de máquinas virtuales para soportar el sistema operativo y la paquetería necesaria[144] que soporte el sistema a probar, dentro de esta máquina virtual además instalaremos el proyecto abierto para la seguridad de aplicaciones OWASP-BWA[145] (Web Open Web Application Security Project), que es un proyecto de código abierto para profesionales de la seguridad, el cual permite crear un ambiente de prueba seguro donde podemos identificar vulnerabilidades en las aplicaciones desarrolladas.

De esta forma podemos crear y usar un ambiente seguro para evitar los controles y detectar vulnerabilidades en los sistemas desarrollados e informar de los hallazgos de las pruebas de penetración antes de que estos se conviertan en brechas de seguridad y se generen daños por ellas.

8.9 Programando desde la Web

Existen diferentes servicios WEB que permiten editar, compilar y ejecutar código de diversos lenguajes y paquetes desde el navegador, esto en aras de que los estudiantes y profesores que cuenten con algún sistema de acceso a red y un navegador puedan programar en los más diversos lenguajes, IDEs y Terminales sin hacer instalación alguna en su equipo de cómputo, tableta o teléfono celular.

Algunos ejemplos de estos servicios son:

8.9.1 SQLite

Se puede trabajar este manejador de bases de datos desde la red, ingresando a la dirección: https://www.codechef.com/ide

Por ser un servicio en red, no es necesario conocer qué bases de datos hay, ni seleccionar alguna de las existentes para iniciar el trabajo en la base de datos. Por lo tanto, sólo creamos las tablas necesarias, agregamos valores y hacemos las operaciones requeridas con la base de datos creada, por ejemplo:

Hackers éticos e investigadores de seguridad en todo el mundo y contiene cientos de herramientas para el trabajo forense, esta distribución está basada en Debian.

[144] Puede ser por ejemplo Linux, Apache, MySQL, PHP, Perl, Python, etc.

[145] Es un proyecto comunitario que ayuda a organizaciones e instituciones educativas a diseñar, implementar y administrar aplicaciones seguras. Encontrando amenazas como: Inyección (SQL, OS, XXE, LDAP), irrupción en la autentificación y la sesión, Cross Site Scripting, irrupción en el control de acceso, errores en la configuración de la seguridad, exposición de datos sensibles, protección insuficiente contra ataques, uso de componentes con vulnerabilidades conocidos, API con protección insuficiente, etc.

Creamos la tabla de Productos e insertamos los valores mediante:

```
create table Productos(
    id_prod int,
    descripcion text,
    precio float(9,2),
    precioVenta float(9,2)
);
insert into Productos
values (1, 'Cigarros', 25.5, 30.0), (2, 'Refresco', 4.5, 6.0);
select * from Productos;
```

ahora, creamos la tabla de ventas e insertamos los valores mediante:

```
create table Ventas(
    id_prod int,
    cantidad int,
    fecha timestamp
);
insert into Ventas
values (1, 2, '2020/12/01 8:01:00'), (2, 5, '2020/12/01 10:15:00'),
(2, 4, '2020/12/01 13:34:00'), (1, 3, '2020/12/01 21:56:00');
select * from Ventas;
```

de esta forma, ahora podemos usar SELECT para hacer un JOIN de nuestras tablas y mostrar el resultado de la solicitud, mediante:

```
    select Ventas.fecha, Ventas.cantidad, Productos.descripcion,
Productos.precioVenta, (Ventas.cantidad * Productos.precioVenta)
    from Ventas
    INNER JOIN Productos ON Ventas.id_prod=Productos.id_prod;
```

8.9.2 MySQL y MariaDB

Se puede trabajar este manejador de bases de datos desde la red, ingresando a la dirección: https://www.jdoodle.com/

Por ser un servicio en red, no es necesario conocer qué bases de datos hay, ni seleccionar alguna de las existentes para iniciar el trabajo en la base de datos. Por lo tanto, sólo creamos las tablas necesarias, agregamos valores y hacemos las operaciones requeridas con la base de datos creada, por ejemplo:

Creamos la tabla de Productos e insertamos los valores mediante:

```
create table Productos(
    id_prod int,
    descripcion text,
    precio float(9,2),
    precioVenta float(9,2)
);
insert into Productos
values (1, 'Cigarros', 25.5, 30.0), (2, 'Refresco', 4.5, 6.0);
select * from Productos;
```

ahora, creamos la tabla de ventas e insertamos los valores mediante:

```
create table Ventas(
    id_prod int,
    cantidad int,
    fecha timestamp
);
insert into Ventas
values (1, 2, '2020/12/01 8:01:00'), (2, 5, '2020/12/01 10:15:00'),
(2, 4, '2020/12/01 13:34:00'), (1, 3, '2020/12/01 21:56:00');
select * from Ventas;
```

de esta forma, ahora podemos usar SELECT para hacer un JOIN de nuestras tablas y mostrar el resultado de la solicitud, mediante:

```
    select Ventas.fecha, Ventas.cantidad, Productos.descripcion,
  Productos.precioVenta, (Ventas.cantidad * Productos.precioVenta)
    from Ventas
    INNER JOIN Productos ON Ventas.id_prod=Productos.id_prod;
```

8.9.3 PostgreSQL

Se puede trabajar este manejador de bases de datos desde la red, ingresando a la dirección: https://rextester.com/l/postgresql_online_compiler

Por ser un servicio en red, no es necesario conocer qué bases de datos hay, ni seleccionar alguna de las existentes para iniciar el trabajo en la base de datos. Por lo tanto, sólo creamos las tablas necesarias, agregamos valores y hacemos las operaciones requeridas con la base de datos creada, por ejemplo:

Creamos la tabla de Productos e insertamos los valores mediante:

```
create table Productos(
    id_prod int,
    descripcion text not null,
    precio money,
    precioVenta money,
    primary key (id_prod)
);
insert into Productos
values (1, 'Cigarros', 25.5, 30.0), (2, 'Refresco', 4.5, 6.0);
select * from Productos;
```

ahora, creamos la tabla de ventas e insertamos los valores mediante:

```
create table Ventas(
    id_prod int,
    cantidad int,
    fecha timestamp
);
insert into Ventas
values (1, 2, '12/01/2020 8:01:00.00'), (2, 5, '12/01/2020 10:15:00.00'),
(2, 4, '12/01/2020 13:34:00.00'), (1, 3, '12/01/2020 21:56:00.00');
select * from Ventas;
```

de esta forma, ahora podemos usar SELECT para hacer un JOIN de nuestras tablas y mostrar el resultado de la solicitud, mediante:

```
    select Ventas.fecha, Ventas.cantidad, Productos.descripcion,
  Productos.precioVenta, (Ventas.cantidad * Productos.precioVenta)
    from Ventas
    INNER JOIN Productos ON Ventas.id_prod=Productos.id_prod;
```

8.9.4 MongoDB

Se puede trabajar este manejador de bases de datos desde la red, ingresando a la dirección: https://www.jdoodle.com/online-mongodb-terminal/

Por ser un servicio en red, no es necesario conocer que bases de datos hay, ni seleccionar alguna de las existentes para iniciar el trabajo en la base de datos. Por lo tanto, sólo creamos las colecciones necesarias, agregamos

valores y hacemos las operaciones requeridas con la base de datos creada, por ejemplo:

Creamos el document de Productos e insertamos los valores mediante:

```
var miProductos =
[
    {"_id" : 1, "descripcion" : "Cigarros", "precio" : 25.5,
"precioVenta" : 30.0},
    {"_id" : 2, "descripcion" : "Refresco", "precio" : 4.5,
"precioVenta" : 6.0},
];
db.Productos.insert(miProductos);
db.Productos.find()
```

ahora, creamos documento de ventas e insertamos los valores mediante:

```
var miVentas =
[
    {"_id" : 1, "id_prod" : 1, "Cantidad" : 2, "Fecha" :
"2020/12/01 8:01:00"},
    {"_id" : 2, "id_prod" : 2, "Cantidad" : 5, "Fecha" :
"2020/12/01 10:15:00"},
    {"_id" : 3, "id_prod" : 2, "Cantidad" : 4, "Fecha" :
"2020/12/01 13:34:00"},
    {"_id" : 4, "id_prod" : 1, "Cantidad" : 3, "Fecha" :
"2020/12/01 21:56:00"},
];
db.Ventas.insert(miVentas);
db.Ventas.find()
```

de esta forma, ahora podemos usar Find para hacer el equivalente a un JOIN como en los ejemplos anteriores y mostrar el resultado de la solicitud, mediante:

```
db.Ventas.find().forEach(
    function (object) {
        var commonInBoth=db.Productos.findOne({ "_id":
object.id_prod} );
```

```
                if (commonInBoth != null) {
                    print(object.Fecha, object.Cantidad, common-
InBoth.descripcion, commonInBoth.precioVenta, (commonInBoth.precioVenta
* object.Cantidad));
                }
            }
        );
```

9 El Cómputo en Paralelo

La computación paralela es el uso de múltiples recursos computacionales para resolver un problema. Se distingue de la computación secuencial en que varias operaciones pueden ocurrir simultáneamente. Los sistemas de cómputo con procesamiento en paralelo surgen de la necesidad de resolver problemas complejos en un tiempo razonable, utilizando las ventajas de memoria, velocidad de los procesadores, formas de interconexión de estos y distribución de la tarea, a los que en su conjunto denominamos arquitectura en paralelo. Entenderemos por una arquitectura en paralelo a un conjunto de procesadores interconectados capaces de cooperar en la solución de un problema.

Así, para resolver un problema en particular, se usa una arquitectura o combinación de múltiples arquitecturas (topologías), ya que cada una ofrece ventajas y desventajas que tienen que ser sopesadas antes de implementar la solución del problema en una arquitectura en particular. También es necesario conocer los problemas a los que se enfrenta un programador que desea que su programa corra en paralelo, como son: el partir eficientemente[146] un problema en múltiples subtareas y cómo distribuir eficazmente[147] estas según la arquitectura en particular con que se trabaje. La eficacia difiere de la eficiencia en el sentido de que la eficiencia hace referencia en la mejor utilización de los recursos computacionales (procesadores), en tanto que la eficacia hace referencia en la capacidad para alcanzar un objetivo, aunque en el proceso no se haya hecho el mejor uso de los recursos computacionales.

El cómputo de alto rendimiento (HPC por High-Performance Computing) está formado por un conjunto de computadoras unidas entre sí en forma de Cluster[148], para aumentar su potencia de trabajo y rendimiento. En 2019 la supercomputadora *SUMMIT* de IBM funcionaban a más de 148 peta-Flops[149] (cada uno de ellos equivale a la realización de más de 1000 billones

[146] Es un indicador de la utilización efectiva de los diferentes recursos computacionales (principalmente del uso de los procesadores) en relación al algoritmo dado. Se entiende que la eficacia se da cuando se utilizan menos recursos para lograr un mismo objetivo. O al contrario, cuando se logran más objetivos con los mismos o menos recursos.

[147] La podemos definir como el nivel de consecución de metas y objetivos. La eficacia hace referencia a nuestra capacidad de lograr lo que no proponemos.

[148] Existe el Ranking de las 500 supercomputadoras más poderosas del mundo (esta se actualiza cada seis meses en junio y noviembre) y puede ser consultada en:
https://top500.org

[149] "FLOPS" operaciones de punto flotante por segundo. Describe una velocidad de procesamiento teórica: para hacer posible esa velocidad es necesario enviar datos a los

de operaciones por segundo de las pruebas de rendimiento del HPC-AI), mientras que en junio de 2020 *FUGAKU* de Japón superaba los 415.5 peta-Flops y en noviembre del mismo año alcanzó los 2.0 exa-Flops, siendo este el primer equipo en alcanzar valores por arriba de un exa-Flops para cualquier precisión sobre cualquier tipo de Hardware, manteniéndose en el primer lugar del top 500 durante el 2021

El clásico uso del paralelismo, es el de diseño de programas eficientes en el ámbito científico. La simulación de problemas científicos es un área de gran importancia, los cuales requieren de una gran capacidad de procesamiento y de espacio de memoria, debido a las complejas operaciones que se deben realizar.

Otro uso clásico es el de las gráficas y vídeos de simulaciones generadas por computadora. La generación de fotogramas y videos requiere de una gran cantidad de cálculos matemáticos. Esto supone una tarea muy compleja para un solo procesador, luego es necesario que haya algún tipo de paralelismo, para distribuir la tarea para que esta sea realizada eficiente y eficazmente.

Tradicionalmente, los programas informáticos se han escrito para el cómputo en secuencial. Para resolver un problema, se construye un algoritmo y se implementa como un flujo en serie de instrucciones. Estas instrucciones se ejecutan en una unidad central de procesamiento en un ordenador. Sólo puede ejecutarse una instrucción a la vez y un tiempo después de que la instrucción ha terminado, se ejecuta la siguiente.

Actualmente, es una práctica común usar directivas de compilación en equipos paralelos sobre programas escritos de forma secuencial, con la esperanza que sean puestos por el compilador como programas paralelos. Esto en la gran mayoría de los casos genera códigos poco eficientes, pese a que corren en equipos paralelos y pueden usar toda la memoria compartida de dichos equipos, el algoritmo ejecutado continúa siendo secuencial en una gran parte del código.

La computación en paralelo, por el contrario, utiliza simultáneamente múltiples elementos de procesamiento para resolver un problema. Esto se logra mediante la división del problema en partes independientes de modo que cada elemento de procesamiento pueda ejecutar su parte del algoritmo de manera simultánea con los otros. Los elementos de procesamiento son di-

procesadores de forma continua. Por lo tanto, el procesamiento de los datos se debe tener en cuenta en el diseño del sistema. La memoria del sistema, junto con las interconexiones que unen los nodos de procesamiento entre sí, impactan en la rapidez con la que los datos llegan a los procesadores.

versos e incluyen recursos tales como una computadora con múltiples procesadores, varios ordenadores en red, Hardware especializado, o cualquier combinación de los anteriores.

Los programas informáticos paralelos son más difíciles de escribir que los secuenciales, porque la concurrencia introduce nuevos tipos de errores de Software, siendo las condiciones de sincronización las más comunes. La comunicación y sincronización entre diferentes subtareas son algunos de los mayores obstáculos para obtener un buen rendimiento del programa paralelo.

¿Cómo Paralelizo mi programa? El problema de todos los usuarios de Ciencias e Ingeniería que hacen uso de métodos numéricos y su implementación computacional es: ¿cómo paralelizo mi programa?, esta pregunta no tiene una respuesta simple, depende de muchos factores, por ejemplo: en que lenguaje o paquete está desarrollado el programa, el algoritmo usado para solucionar el problema, a que equipos paralelo tengo acceso y un largo etc. Algunas respuestas ingenuas podrían ser:

- Si uso lenguajes de programación compilables, puedo conseguir un compilador que permita usar directivas de compilación en equipos de memoria compartida sobre programas escritos de forma secuencial, con la esperanza que sean puestos por el compilador como programas paralelos haciendo uso de hilos, OpenMP y optimización del código generado.

- Si uso paquetes como MatLab, Julia o Python, es posible conseguir una versión del paquete que implemente bibliotecas que usen CUDAs, OpenMP o MPI para muchos de los algoritmos más usados en la programación.

- Si mi problema cabe en un sólo equipo, entonces puedo usar dos o más cores para resolver mi problema usando memoria compartida (puedo usar OpenMP, trabajar con hilos o usando paquetes como MatLab o lenguajes como Python, Julia, Fortran, C y C++, etc.), pero sólo puedo escalar para usar el máximo número de cores de un equipo (que en la actualidad puede ser del orden de 128 cores, hasta Terabytes de RAM y con almacenamiento de algunas decenas de Terabytes).

- Si mi problema cabe en la GRAM de una GPU, entonces el posible usar una tarjeta gráfica (usando paquetes como MatLab o lenguajes como

Python, Julia, Fortran, C y C++, etc.), pero sólo puedo escalar a la capacidad de dichas tarjetas gráficas.

- Puedo usar un cluster que usa memoria distribuida-compartida en conjunto con tarjetas gráficas, este tipo de programación requiere una nueva expertes y generalmente implica el uso de paso de mensajes como MPI y rediseño de los algoritmos usados en nuestro programa.

Notemos primero, que no todos los algoritmos son paralelizables. En cualquier caso se tienen que ver los pros y contras de la paralelización para cada caso particular. Pero es importante destacar que existen una gran cantidad de bibliotecas y paquetes que ya paralelizan ciertos algoritmos que son ampliamente usados como la solución de sistemas lineales[150] y no lineales. Además, el tiempo de programación necesario para desarrollar una aplicación paralela eficiente y eficaz para la gran mayoría de los programadores puede ser de semanas o meses en el mejor de los casos. Por ello, es necesario hacer un balance entre las diferentes opciones de paralelización para no invertir un tiempo precioso que puede no justificar dicha inversión económica, de recursos computacionales y sobre todo de tiempo.

Antes de iniciar con los tópicos del cómputo en paralelo, es necesario conocer cómo está constituida nuestra herramienta de trabajo: la computadora. Los conocimientos básicos de arquitectura de las computadoras nos permite identificar los componentes de Hardware que limitarán nuestros esfuerzos por realizar una paralelización óptima de nuestro programa y en muchos casos descubriremos de la peor manera que los algoritmos usados en nuestro programa no son los más adecuados para paralelizar; y descubriremos con horror que en algunos casos los tiempos de ejecución no mejoran sin importar cuantos equipos adicionales usemos, en el peor de los casos, el tiempo de ejecución se incrementa al aumentar equipos en vez de disminuir.

[150] Algunas de las bibliotecas más usadas para resolver sistemas lineales usando matrices bandadas y dispersar son PETCs, HYPRE, ATLAS, LAPACK++, LAPACK, EISPACK, LINPACK, BLAS, entre muchas otras alternativas, tanto para implementaciones secuenciales como paralelas y más recientemente para hacer uso de los procesadores CUDA en las GPU de nVidia.

9.1 Computadoras Actuales

La computadora (también conocida como ordenador) actual es una máquina digital programable que ejecuta una serie de comandos para procesar los datos de entrada, obteniendo convenientemente información que posteriormente se envía a las unidades de salida. Una computadora está formada físicamente por numerosos circuitos integrados y varios componentes de apoyo, extensión y accesorios, que en conjunto pueden ejecutar tareas diversas con suma rapidez y bajo el control de un programa (Software).

La constituyen dos partes esenciales, el Hardware, que es su estructura física (circuitos electrónicos, cables, gabinete, teclado, etc.), y el Software, que es su parte intangible (programas, datos, información, documentación, etc.).

Con respecto al Hardware[151], se encuentra compuesto por una serie de dispositivos, clasificados según la función que estos desempeñen. Dicha clasificación se compone de:

- Los dispositivos de entrada son todos aquellos que permiten la entrada de datos a una computadora. Estos dispositivos (periféricos), son los que permiten al usuario interactuar con la computadora. Ejemplos: teclado, Mouse (ratón), micrófono, Webcam, Scanner, etc.

- Los dispositivos de salida, son todos aquellos que permiten mostrar la información procesada por la computadora. Ejemplos: monitor, impresora, auriculares, altavoces, etc.

- Los dispositivos de comunicación son aquellos que permiten la comunicación entre dos o más computadoras. Ejemplos: Modem, Router, placa de red, Bluetooth, etc.

[151] En Debian GNU/Linux podemos instalar la aplicación *lshw* para conocer los distintos componentes de la computadora, mediante:

```
# apt install lshw
```

Así, para ver de forma resumida los dispositivos que componen la computadora usamos:

```
# lshw -short
```

Si necesitamos más detalle usamos:

```
# lshw
```

- Los dispositivos de almacenamiento, son todos aquellos que permiten almacenar datos en el ordenador. Ejemplos: disco duro, Pendrive, Diskette, CD y DVD, etc.

- Los dispositivos de cómputo, son aquellos encargados de realizar las operaciones de control necesarias, sobre el resto de los dispositivos la computadora.

La unidad central de procesamiento (Central Processing Unit CPU) se comunica a través de un conjunto de circuitos o conexiones llamada Bus de datos o canal de datos. El bus conecta la CPU a los dispositivos de almacenamiento, los dispositivos de entrada y los de salida.

En la actualidad, una computadora puede tener una CPU de 32 o de 64[152] bits para describir anchura de registros, Bus de direcciones, Bus de datos o instrucciones. Es decir que la cantidad de bits nos permite diferenciar el tipo de CPU que tiene nuestro equipo.

La CPU, el sistema operativo, el Software en general y los Drivers se basan en una misma arquitectura, pero que tenga más o menos bits puede ser significativo en la experiencia del usuario.

[152] ¿Por qué se llama x86 a la de 32 bits y x64 a la de 64 bits?

Por pereza. El término adecuado para la arquitectura de 64 bits basada en las tecnologías desarrolladas por Intel y AMD desde 1976 es x86-64, que es como la llamó AMD cuando la desarrolló.

El término "x86" se acuñó porque Intel denominó 8086 a la primera CPU que utilizaba esta microarquitectura. Era una CPU de 16 bits con un conjunto de instrucciones similar al anterior Intel 8085 de 8 bits, por lo que Intel le dio un número más alto. A principios de los años 90, a los desarrollos posteriores de la 8086 se les asignaron números que terminaban en "86": 80186, 80286, 80386 y 80486. En la mayoría de los casos, se omitió la parte "80" y toda la familia de CPU se denominó "x86", por razones obvias. Con lo que habría sido el 80586, Intel eliminó los números porque no podían registrarlos como marca, así que nació la marca "Pentium", que primero significaba quinta generación, pero luego como una simple marca.

Cuando Intel pasó a los 32 bits con el conjunto de instrucciones x86 con el 386 en 1985, no cambiaron la marca de nada, pero cuando AMD extendió x86 a 64 bits, lo comercializaron como x86-64. Muchos desarrolladores de Software lo llamaron "AMD64" en su lugar, ya que AMD lo desarrolló. Intel no estaba dispuesto a hacer eso, así que lo llamaron "EMT64" cuando adoptaron las extensiones de 64 bits de AMD después del fracaso de Itanium. Al final, la mayoría de la gente simplemente lo llamó "x64", aunque esto no tiene ningún sentido.

En la actualidad, la mayoría de ordenadores cuentan con CPU de 64[153] bits. Ahora bien, todavía hay equipos algo antiguos que cuentan con la arquitectura de 32 bits, por lo que sigue coexistiendo en el mercado y el sector hace desarrollos también pensando en sus capacidades.

Desde el punto de vista funcional es una máquina que posee, al menos, una unidad central de procesamiento, unidad de memoria (Random Access Memory RAM , Read Only Memory ROM y Caché) y dispositivos de entrada/salida (periféricos). Los periféricos de entrada permiten el ingreso de datos, la CPU se encarga de su procesamiento (operaciones aritmético-lógicas) y los dispositivos de salida los comunican a los medios externos. Es así, que la computadora recibe datos, los procesa y emite la información resultante, la que luego puede ser interpretada, almacenada, transmitida a otra máquina o dispositivo o sencillamente impresa; todo ello a criterio de un operador o usuario y bajo el control de un programa de computación.

[153] ¿Alguna empresa ha fabricado una CPU de 128 bits?

Si alguien quiere fabricar una CPU de 128 bits, puede utilizar el diseño RISC-V, que admite núcleos de 32, 64 y 128 bits. No hay problema, solo se necesita un diseño adecuado y listo. Muchos dicen que el beneficio de una CPU (núcleo) de 64 bits es la enorme memoria direccionable, 64 bits en comparación con los 32 bits de los núcleos de 32 bits. ¡Pero eso no es cierto!. Todos los núcleos de 64 bits (excepto los muy especiales) tienen un bus de direcciones de 52 o 56 bits. Windows y Linux utilizan internamente un espacio de direccionamiento de a lo más 48 bits. No hay necesidad de memoria física de 64 bits porque hoy en día no es posible tenerla (teóricamente hasta 16 Exabytes).

Algo similar ocurre con los núcleos de 128 bits. Como la memoria no es importante, lo único que queda es la matemática de 128 bits. La pregunta es: ¿quién necesita la matemática de 128 bits? En 1978, el 8086 tenía un coprocesador matemático, el 8087, que admitía tipos de punto flotante de 80 bits. Hoy en día, es similar, pero el tipo máximo admitido sigue siendo de 64 bits que tiene aproximadamente 16 dígitos decimales de precisión con un rango del orden de 1.7×10^{-308} a 1.7×10^{308}, el número de 32 bits que tiene 14 dígitos decimales de precisión con un rango del orden de 3.4×10^{-38} a 3.4×10^{38}. Además, existe el número de 80 bits que tiene 18 dígitos decimales de precisión con un rango del orden de 3.4×10^{-4096} a 1.1×10^{4096}.

En algunos compiladores existen 128 bits de datos, pero no existe matemática de Hardware, sino que se implementa en Software. Se necesitan tipos tan grandes en muy pocas circunstancias.

Lo que importa hoy en día es la paralelización, el procesamiento de múltiples tipos en paralelo. Durante mucho tiempo, x86 ha tenido AVX512, que es un motor matemático de 512 bits (reemplazó al 8087), pero es capaz de trabajar con tipos de 8 a 64 bits en paralelo, por ejemplo, 8 operaciones matemáticas en paralelo utilizando FP64 (punto flotante de 64 bits).

CPU la Unidad Central de Proceso (Central Processing Unit CPU) es aquella parte del procesador que se encarga de ejecutar las diversas acciones que ordenemos al dispositivo que debe llevar a cabo. La CPU es el componente básico dentro de todo dispositivo inteligente, ya que prácticamente cualquier proceso que se ordene al sistema pasa por él. Con el paso del tiempo, además, su eficiencia y calidad ha alcanzado grandes cotas, aunque tecnologías al alza como las NPU han supuesto un mayor salto cualitativo que el de aumentar la potencia bruta de la CPU.

GPU, para el procesamiento gráfico la Unidad de Procesamiento Gráfico (Graphics Processing Unit GPU) es el apartado que se dedica a las acciones de mayor peso: las de componente gráfico. De este modo, acciones como la ejecución de videojuegos, o la edición y renderizado de vídeos, se llevan a cabo a través de la GPU del sistema. La calidad del teléfono, ordenador u otro dispositivo inteligente, suele ir supeditada a menudo a la calidad de su GPU, que dependerá de la banda de precio del aparato en cuestión. Eso sí, si los otros componentes no tienen la misma calidad, se puede producir el temido cuello de botella.

En los smartphones, la GPU ya va integrada en los procesadores, pero en los PC's, como AMD y NVIDIA se muestran como marcas especializadas en las GPUs. Cuentan con todo tipo de gamas y también poseen un amplio abanico de precios, un valor siempre directamente proporcional a la calidad y capacidad de sus gráficas, y a su vigencia en el mercado.

NPU, redes neuronales en tu dispositivo la Unidad de Procesamiento Neuronal (Neural Processing Unit NPU), a diferencia de la GPU, que cuenta con un funcionamiento paralelo al de la CPU, puede encargarse de funciones similares a las de la CPU, pero lo hace de un modo mucho más eficiente. Impulsada por Inteligencia Artificial, una NPU[154] es capaz de priorizar procesos para ejecutarlos de un modo exponencialmente más veloz y con un consumo mucho menor. En móviles, se usa especialmente para mejorar el procesado de fotografías, aunque participa en muchos otros procesos.

[154] Para muestra, la compañía Cerebras con su procesador WSE-3 aglutina 4 billones de transistores, tiene una superficie de 46,225 mm^2, integra nada menos que 900,000 núcleos optimizados para IA y es capaz de entrenar hasta 24,000 millones de parámetros, lo que también equivaldría a un rendimiento máximo de IA de 125 petaflops.

Además, esta arquitectura todavía tiene años de progreso por delante, a diferencia de la CPU y la GPU, cuyas mejoras ya son de carácter más leve y basadas en aumentar la potencia bruta. La tecnología NPU, en cambio, lleva menos tiempo entre nosotros y todavía tiene mucho margen de mejora para ofrecer un rendimiento cada vez más poderoso.

Desde la llegada de los procesadores de más de un núcleo a PC, como consecuencia de la imposibilidad de hacerlos escalar en potencia solo por velocidad de reloj, la forma de entender los diferentes chips cambió. Han pasado ya dos décadas de dicho cambio y ante la inminente salida de los Chips disgregados o por Chiplets al mercado de masas, no está de más recordar la organización más común en el mundo del Hardware en todo este tiempo.

¿Qué es un SoC? las siglas SoC significan System on a Chip y hace referencia a todo chip que tiene la mayoría de componentes integrados en una misma pieza de silicio sin llegar a ser un microcontrolador. Se trata de la pieza de Hardware más usada por el hecho de que a día de hoy todo procesador para PC, teléfono móvil, consola de videojuegos, televisor o incluso servidores, es un SoC y pese a las diferencias entre ellos, todos tienen una organización común.

En realidad, todas las CPU actuales son SoC, ya que se trata de "varias CPU" diseñadas para funcionar alrededor de un elemento de intercomunicación central. Este se encarga de interconectar los diferentes elementos entre sí y de darles acceso a interfaces externas.

Por ejemplo, con la memoria RAM (u otros), a la que está asociado el controlador de memoria que comparten todos sus elementos o a los periféricos, y a los cuales se puede acceder directamente con una serie de interfaces específicas, o a través de un Chipset externo encargado de gestionar los diferentes periféricos y componentes.

En PC, debido a que la comunicación con los periféricos se hace a través del uso de direcciones de memoria de la RAM principal, los componentes relacionados con esta se encuentran subordinados al controlador de memoria. Por lo que son una pieza más conectada a la parte central.

¿Qué es una APU y en qué se diferencia de un SoC? las siglas APU significan Accelerated Processor Unit y fue usada por AMD cuando sus CPU e iGPU (o GPU integrada, la veremos a continuación) no traían consigo ningún sistema de gestión de E/S (entrada y salida). Sin embargo,

la cosa empezó a cambiar ya con la arquitectura "Carrizo" que fue el nombre clave de los últimos SoC antes del lanzamiento de los AMD Ryzen, donde se incorporaron varias interfaces de periféricos directamente en la CPU.

A día de hoy todo es un SoC, lo que ocurre es que, en sobremesa, las torres tienen tanta conectividad y capacidad de expansión que se suele emplear un Chipset, y lo mismo ocurre en estaciones de trabajo y servidores, pero no en el resto. Y es que si hablamos de un Chip para un PC portátil, una consola o un móvil, entonces al no existir tantas interfaces para periféricos y otros componentes, entonces estás se pueden integrar en un mismo chip.

En realidad, el término APU es más bien comercial de AMD y al día de hoy se utiliza como sinónimo de SoC, pero se puede resumir en que una APU carece de cualquier gestión de periféricos y requiere de un Chip externo para ello. Mientras que un SoC tiene las especificaciones mínimas en ese aspecto, pese a ser también ampliables. Para simplificar la idea, un SoC es una APU más completa.

¿¡GPU qué es y cuáles son sus características concretas frente a una dGPU? las siglas iGPU corresponden a integrated GPU, y hace referencia a todo componente de este tipo que se encuentre integrado en una APU o un SoC. Por lo que se trata de procesadores gráficos de potencia limitada que se pueden ver lastrados en velocidad de reloj por el problema del ahogamiento termal (Thermal Throttling) que se produce cuando muchas partes comparten el mismo espacio físico.

Si bien, es posible llegar a ciertos niveles de rendimiento que son aceptables de cara a reproducir las escenas en 3D a tiempo real en los videojuegos, y otras tareas de carácter profesional donde se usa una tarjeta gráfica, estas se ven cuanto menos limitadas:

El hecho de tener que compartir espacio en el mismo Chip con los diferentes núcleos de la CPU produce que esta no pueda alcanzar la misma velocidad de reloj que se alcanzaría siendo un Chip aparte e independiente.

En PC, debido a que como la memoria RAM usa DDR o LPDDR, el ancho de banda es pequeño y hemos de partir del hecho de que el rendimiento de todo Chip gráfico, incluido una iGPU, depende del ancho de banda que se le puede otorgar con dicha memoria RAM.

Los SoC con iGPU actuales tienen un tamaño fijo, definido este por la cantidad de pines soportados por la interfaz con la placa base. Esto limita el tamaño, no solo del Chip, sino también de la gráfica integrada en el mismo.

iGPU en consolas de videojuegos al contrario de lo que ocurre con los PC, las consolas de videojuegos no tienen que seguir una serie de normas respecto a sus componentes. Para empezar, no ven el tamaño de sus chips limitados por un estándar de placa base, dado que son productos únicos y exclusivos. Esto les permite tener el tamaño que quieran, incluso más que una CPU convencional para PC, lo que les permite tener una iGPU en su SoC mucho más avanzada.

El otro punto es la memoria utilizada, ya que la de las tarjetas gráficas también se usan como memoria principal. Esta da el ancho de banda necesario para que la iGPU alcance cierto nivel de rendimiento, pero su latencia es mucho más alta que la RAM convencional de PC que está más optimizada en ese aspecto, por lo que el rendimiento de su CPU suele ser más bajo que su equivalente para ordenador.

¿Qué es una dGPU en un PC o portátil? las dGPU, o GPU dedicadas, no son otra cosa que las GPU de toda la vida, pero la particularidad es que también son SoC, ya que tiene varios núcleos, especializados en tareas gráficas, alrededor de una interfaz central y compartiendo todos ellos un mismo acceso a memoria.

Sin embargo, carecen de núcleos de CPU en su interior, de ahí a que no se les llame APU o SoC, por el hecho de que de existir estos elementos pasarían a ser una iGPU. Por lo tanto, su principal particularidad de las dGPU es que tienen su propia memoria RAM (SDRAM para ser concretos) la cual suele rodear estos Chips, ya sea en forma de tarjeta gráfica o soldados en la placa de los ordenadores portátiles. A esta la llamamos VRAM y es de uso exclusivo de la dGPU o GPU.

Los equipos de cómputo los podemos clasificar[155] por:

- Equipos móviles: estos equipos buscan un equilibrio entre su capacidad de cómputo versus el rendimiento energético de sus baterías -para operar el mayor tiempo posible sin recargarse- y su peso, entre estos equipos destacan las Laptops, Notebook, Netbook, Ultrabook, tabletas, teléfonos inteligentes, etc.

[155] El ordenador del Apollo 11, el Block II, funcionaba a una velocidad de 2 MHz y tenía 2 KB de memoria RAM y 32 KB de memoria ROM. Para ponerlo en contexto, el chip de un cargador USB-C moderno es 563 veces más potente que la computadora que se usó en el Apollo 11, al menos en términos de potencia bruta.

- Equipos de escritorio: estos equipos al estar permanentemente conectados a la corriente eléctrica pueden tener un mayor número de componentes y disponen de una mejor capacidad disipación de calor por lo que pueden contener una mayor cantidad componentes, como discos, RAM o tarjetas de video y el tamaño, peso o consumo energético no es un inconveniente.

- Servidores: son equipos que suelen atender a múltiples usuarios simultáneamente y disponen de gran cantidad de Cores, RAM, disco y son interconectados por red de alta velocidad con otros servidores para atender las crecientes necesidades de los centros de datos los cuales deben estar permanentemente en operación. Generalmente los equipos son montados en Racks con otras decenas de equipos, por lo que su arquitectura se ve limitada a una moderada generación de calor por parte de sus componentes ya que su sistema de ventilación es por aire para todo el Rack.

- Estaciones de trabajo: son equipos individuales diseñados para atender cargas computacionales intensas, por lo que requieren Hardware más complejo y potente como puede ser múltiples tarjetas de video (con decenas de miles de Cores gráficos) , discos (con cientos de Terabytes), gran cantidad de RAM (pueden llegar a superar el Terabyte) y sistema de refrigeración por aire o líquido, etc.

- Cómputo intensivo: son equipos interconectados por red de alta velocidad con procesadores y tarjetas gráficas dedicadas para el cálculo numérico que soportan cargas intensas por largos periodos de tiempo, los más comunes son los que forman parte de los Cluster que llegan a tener millones de cores.

FLOPS Una medida relativamente objetiva para analizar el rendimiento de un dispositivo suele ser medir sus operaciones de punto flotante por segundo o más conocidas como FLOPS[156]. Hay que tener en cuenta que la medición de FLOPS es muy compleja porque las diferentes operaciones en punto flotante

[156] El Cray-1 fue puesto marcha en 1975 y utilizaba una CPU a 80 MHz y llevaba integrada una unidad SIMD de 64 bits de precisión de punto flotante, lo cual fue un salto de gigante que permitió un salto de los 3 MFLOPS de potencia del CDC 6600 a los 160 MFLOPS en el Cray-1.

llevan diferentes cantidades de tiempo para ejecutarse. Y no todo el mundo utiliza las mismas operaciones para establecer los cálculos.

Por ejemplo, una división simple como 1/5, toma significativamente menos tiempo que el cálculo del logaritmo de 5. Por eso, se estableció el algoritmo de Linpack como un estándar representativo con el que poder medir todos los sistemas bajo el mismo baremo de FLOPS-.

Es importante señalar que el algoritmo de Linpack utiliza el formato en punto flotante de doble precisión (64 bits[157] que tiene aproximadamente 16 dígitos decimales de precisión con un rango del orden de 1.7×10^{-308} a 1.7×10^{308}). Sin embargo, como veremos la mayoría de los valores que dan los fabricantes son con precisión simple (32 bits que tiene 14 dígitos decimales de precisión con un rango del orden de 3.4×10^{-38} a 3.4×10^{38}). Además, los valores que dan los fabricantes suelen ser teóricos y en la práctica suelen ser inferiores debido a otros factores limitantes como la frecuencia de reloj o la velocidad de las memorias ROM y RAM.

Por tanto, aunque todos hemos acabado midiendo el rendimiento en FLOPS, no es una medida absoluta de la potencia del CPU ni de una GPU. Por ejemplo para algunos dispositivos tenemos:

Móviles El SoC Snapdragon 821 que monta una GPU Adreno 530 tiene una potencia de 519.2 gigaFLOPS (0.52 TFLOPS), y los Chips Apple A9X del iPad Pro alcanzan los 345.6 gigaFLOPS (0.35 TFLOPS), todos ellos

[157] ¿Por qué no hay un procesador de 128 bits, si los hay de 64 y 32 bits?

En gran medida, esto se reduce a la cuestión de qué se entiende realmente por "procesador de 64 bits". Al inicio de las computadoras electrónicas, un "procesador de 8 bits" tenía registros de 8 bits, pero la mayoría tenía direccionamiento de 16 bits (y la Unidad de Aritmética y Lógica ALU a veces tenía 4 bits, por lo que cuando se añadían dos registros de 8 bits, la CPU a menudo lo hacía en dos pasos, añadiendo los 4 bits inferiores y luego añadiendo por separado los 4 bits superiores más el acarreo de los 4 inferiores).

Hoy en día, hemos invertido un poco esa situación: un procesador de "64 bits" (más o menos) tiene direccionamiento de 64 bits, pero tiene registros de 128 bits, 256 bits y, en algunos casos, incluso de 512 bits. Pero como los usos de números mayores a 64 bits son bastante raros, esos registros más grandes se usan normalmente para realizar operaciones en una cantidad de operandos más pequeños con una sola instrucción. Por lo tanto, en lugar de una ALU que tiene la mitad del tamaño de un registro y realiza una sola operación en múltiples ciclos de reloj, tenemos una ALU que tiene entre 2 y 4 veces el tamaño de un registro normal y realiza múltiples operaciones en un solo ciclo de reloj.

Dependiendo de cómo prefieras ver las cosas, podrías argumentar razonablemente que una CPU actual de "64 bits" es en realidad una CPU de 128 bits, 256 bits o incluso 512 bits.

medidos con precisión simple de 32-bits.

CPU

- Intel Xeon W-3245: 1.4 TFLOPS
- Intel Core i9-9900X: 1.2 TFLOPS
- AMD Ryzen 9 3950X: 1.1 TFLOPS

Los procesadores de gama media-alta rondan el medio TFLOPS:

- AMD Ryzen 7 3700X: 546.0 GFLOPS - 0.55 TFLOPS
- Intel Core i9-9900: 499.0 GFLOPS - 0.50 TFLOPS
- AMD Ryzen 5 3600X: 461.0 GFLOPS - 0.46 TFLOPS

Tarjetas Gráficas Ojo: La tabla está ordenada por los valores en precisión simple (32-bit) primer columna

GPU	FP32 TFLOPS	FP64 TFLOPS
TITAN V	13.8	6.9
Radeon RX Vega 64	12.7	0.8
GeForce GTX 1080 Ti	11.3	0.4
GeForce GTX 1080	8.9	0.3
Radeon R9 Fury X	8.6	0.5
Radeon HD 7990	7.8	1.9
GeForce GTX 1070	6.5	0.2
Radeon RX 480	5.8	0.4
GeForce GTX 690	5.6	0.2
Radeon R9 290X	5.6	0.7
GeForce GTX 780 Ti	5.3	0.2
Radeon HD 6990	5.1	1.3
GeForce GTX 980	4.9	0.15
Radeon RX 470	4.9	0.3
Radeon R9 290	4.8	0.6
GeForce GTX Titan	4.7	1.5
GeForce GTX 1060	4.4	0.14
Radeon HD 7970 GHz	4.3	1.1
GeForce GTX 780	4.1	0.17

Radeon R9 280X	4.0	1.0
Radeon R9 280	3.3	0.83
GeForce GTX 680	3.1	0.13
Radeon HD 7950	2.9	0.71

Como podemos ver, las tarjetas gráficas de Nvidia, normalmente, tienen una potencia muy alta en precisión simple, pero muy mala en precisión doble. La precisión simple es la que se usa en los juegos, pero la precisión doble es la que se utiliza en los cálculos complejos científicos y en el minado de muchas criptomonedas.

Consolas Todas ellos son en valores de precisión simple (32-bit)

- PlayStation 4: 1.3 TFLOPS
- Xbox One: 1.8 TFLOPS
- PlayStation 4 Pro: 4.2 TFLOPS
- Nintendo Switch: entre 0.4 y 0.5 TFLOPS
- PlayStation 5 promete una GPU con 10.28 TFLOPS
- La Xbox Series X promete una GPU de 12 TFLOPS

SuperCómputo Hace pocos se publicó la 63^a edición del ranking «Top 500» (mayo del 2024) que clasifica los Clusters con mayor rendimiento del mundo. Esta lista se basa en la medida del rendimiento de los sistemas utilizando la prueba de referencia LINPACK, que calcula la velocidad a la que un sistema puede resolver un conjunto de ecuaciones lineales.

Y en esta nueva edición que se presenta, en comparación con la publicación anterior (la edición 62) los Clusters que ocuparon los primeros cinco lugares en el ranking anterior mantuvieron sus posiciones.

- El primer lugar del Top 500 lo sigue manteniendo el «Frontier», quien ha estado liderando el puesto durante 2 años consecutivos. Ubicado en el Laboratorio Nacional Oak Ridge del Departamento de Energía de Estados Unidos, este Clúster cuenta con 8.7 millones de núcleos de procesador (AMD EPYC 64C de 2 GHz) y ofrece un rendimiento de 1206 exaflops. El clúster está potenciado por HPE Cray OS (el software del Clúster está basado en SUSE Linux Enterprise Server edición 15).

- El sistema Aurora en Argonne Leadership Computing Facility en Illinois, EE.UU., ha conseguido el segundo puesto en el TOP500. Este Cluster ha aumentado su número de núcleos de procesador (Xeon CPU Max 9470 52C 2.4 GHz) de 4.8 a 9.2 millones, lo que resulta en un rendimiento de 1012 exaflops (el software del Clúster está basado en SUSE Linux Enterprise Server 15 SP4).
- En el tercer lugar se posiciona el «Eagle» instalado en Microsoft Azure Cloud en EE. UU. recuperando su posición que había alcanzado después de su primera aparición en la lista anterior. El «Eagle» es un Clúster desplegado por Microsoft para la nube Azure y contiene 2 millones de núcleos de procesador (CPU Xeon Platinum 8480C 48C 2GHz), ofrece un rendimiento máximo de 561 petaflops (el software del Clúster está basado en Ubuntu 22.04).

Para poner en contexto los avances en este campo, en el año 2004 IBM era dueña y señora del mundo de la supercomputación, su espectacular BlueGene/L dominaba la lista TOP.500. Aquel monstruo contaba con 32,768 procesadores PowerPC 440 a 700 MHz y 16 TB de memoria. 20 años después una sola NVIDIA GeForce RTX 4090 con 24 GB de memoria GDDR6X es más potente que esa supercomputadora -lo es al menos en rendimiento bruto-, BlueGene/L contaba en ese momento con un rendimiento de 70.72 TFLOPS, pero la propia NVIDIA dejaba claro en el lanzamiento de sus RTX 4090 que estas tarjetas gráficas contaban con una potencia de 83 TFLOPS.

Es más, cuatro RTX 4090 con soporte FP8 logran también rivalizar con la supercomputadora más potente de 2009. Y eso sin apretarle las tuercas a las RTX 4090: en noviembre de 2022 es precisamente lo que hicieron en Wccftech y lograron que la RTX 4090 se convirtiera en la primera tarjeta gráfica del mundo en alcanzar los 100 TFLOPS.

Esa comparación es como decimos real en esa potencia de cálculo en bruto, pero también es cierto que en esa y otras supercomputadoras se tenian mecanismos especiales de comunicación entre procesadores o de transferencia de datos, algo para lo que las GPUs actuales, aún siendo sobresalientes, no están tan optimizadas.

¿Cómo Trabaja una Computadora? Todas las computadoras sean de uno o más procesadores ejecutan los programas realizando los siguientes pasos:

1. Se lee una instrucción
2. Se decodifica la instrucción
3. Se encuentra cualquier dato asociado que sea necesario para procesar la instrucción
4. Se procesa la instrucción
5. Se escriben los resultados

Esta serie de pasos, simple en apariencia, se complican debido a la jerarquía de memoria RAM, en la que se incluye la memoria Caché, la memoria principal y el almacenamiento no volátil como pueden ser los discos duros o de estado sólido (donde se almacenan las instrucciones y los datos del programa), que son más lentos que el procesador en sí mismo. Con mucha frecuencia, el paso (3) origina un retardo muy largo (en términos de ciclos del procesador) mientras los datos llegan en el bus de la computadora.

Durante muchos años, una de las metas principales del diseño microinformático ha sido la de ejecutar el mayor número posible de instrucciones en paralelo, aumentando así la velocidad efectiva de ejecución de un programa. No obstante, estas técnicas han podido implementarse en Chips semiconductores cada vez más pequeños a medida que la fabricación de estos fue progresando y avanzando, lo que ha abaratado notablemente su costo.

El procesador es el cerebro de un ordenador. No hay que olvidar otros componentes como la memoria, el almacenamiento o la tarjeta gráfica dedicada, desde luego, pero el procesador está un escalafón por encima en la jerarquía.

Piensa que, si cambiamos el procesador en dos equipos con la misma memoria, almacenamiento o tarjeta gráfica, el comportamiento puede variar notablemente. Sin embargo, para un mismo procesador, los cambios en el resto de componentes no impactan de forma tan directa en la experiencia de uso de un equipo.

¿Qué es una CPU? Antes de nada, vamos a definir exactamente lo que es una CPU o un procesador. Como bien indican sus siglas en inglés (Central Processing Unit) es la unidad de procesamiento -puede ser Intel, AMD, ARM, etc- encargada de interpretar las instrucciones de un Hardware haciendo uso de distintas operaciones aritméticas y matemáticas. Características principales de un procesador:

- Frecuencia de reloj. Este primer término hace referencia a la velocidad de reloj que hay dentro del propio procesador. Es un valor que se mide en Mhz o Ghz y es básicamente la cantidad de potencia que alberga la CPU. La mayoría de ellas cuentan con una frecuencia base -para tareas básicas- y otra turbo que se utiliza para procesos más exigentes -con un aumento en el consumo de energía y por ende un aumento en la temperatura del procesador, requiriendo sistemas de disipación de calor eficientes-.

- Consumo energético. Es normal que nos encontremos con CPU 's donde su consumo energético varía notablemente. Es un valor que se muestra en vatios (W) y como es obvio, aquellos procesadores de gama superior, serán más propensos a consumir más energía. Ante esto, es importante contar con un eficiente sistema de enfriamiento además de contar con una fuente de alimentación acorde a la potencia requerida por el procesador, la tarjeta gráfica y sus respectivos sistemas de enfriamiento.

- Número de núcleos. Con el avance de la tecnología, ya es posible encontrar tanto procesadores de Intel como de AMD que cuentan ya con decenas de núcleos. Estos cores son los encargados de llevar a cabo multitud de tareas de manera simultánea.

- Número de hilos. Si un procesador tiene Hyperthreading en el caso de Intel o SMT (Simultaneous Multi-Threading) en el caso de AMD, significa que cada uno de los núcleos es capaz de realizar dos tareas de manera simultánea, lo que se conoce como hilos de proceso. Por lo tanto, un procesador de cuatro núcleos físicos con Hyperthreading tendría ocho hilos de proceso, y sería capaz de ejecutar ocho órdenes al mismo tiempo -los hilos no tienen las mismas capacidades de un core real y en muchos casos su uso merma el rendimiento del CPU, pero los sistemas operativos los reconocen como si fueran cores reales-[158].

- Memoria Caché. A la hora de "recordar" cualquier tarea, el propio ordenador hace uso de la memoria RAM. Sin embargo no es eficiente este proceso y por tanto es necesario que utilice la memoria Caché de la CPU para paliar esta deficiencia. El Caché se caracteriza porque se llega a ella de forma más rápida y puede ser tipo L1, L2 y L3.

[158] El AMD EPYC 9845 de 160 núcleos y 320 hilos a una frecuencia de 2,00 GHz basada en Zen 5c, este se acompaña de 640 MB de caché L3.

- Zócalo. Es el tipo de conector con pines o Socket al que se conecta la placa base. Por ejemplo, las últimas de Intel suelen tener el Socket LGA 1200, mientras que las de AMD con Ryzen son AM4.
- Red. Si bien la red es un recurso indispensable en un equipo de cómputo, en el caso de equipos paralelos la velocidad de la red es el mayor cuello de botella en cuanto a rendimiento, por ello es necesario usar redes de alto desempeño como las de InfiniBand con un alto costo económico pero de alto desempeño que pueden llegar al orden de cientos de Gigabytes por segundo.

Nuevos Procesadores la creciente demanda de dispositivos de cómputo ha generado una gran variedad de procesadores, los podemos clasificar como:

- Procesador compuesto por múltiples núcleos de alta eficiencia -con un consumo energético reducido- que sacrifican potencia de procesamiento en aras de extender la carga útil de las baterías de los dispositivos móviles.
- Procesador compuesto por múltiples núcleos de alto rendimiento que pueden estar al tope de su capacidad sin generar excesivo calor y son especialmente usados en servidores y en cómputo intensivo.
- Procesadores compuestos por múltiples núcleos de alto rendimiento que pueden ajustar su velocidad de reloj de manera dinámica para tratar cargas de trabajo pesadas por un cierto tiempo -pues generan gran cantidad de calor-, por lo que requieren un sistema eficiente de enfriamiento, son ideales para estaciones de trabajo.
- Procesadores compuestos por múltiples núcleos híbridos que en lugar de tener un único tipo de núcleo multipropósito, estos Chips cuentan con dos grupos de núcleos. El primero de ellos, compuesto por múltiples núcleos de alta eficiencia, se encarga de procesar las tareas más livianas o en segundo plano que deba realizar un procesador, todo ello, con un consumo energético menor. El otro grupo, compuesto por múltiples núcleos de alto rendimiento, sigue una dinámica opuesta, su consumo ener-gético es superior, pero únicamente entran en funcionamiento cuando la tarea en cuestión requiere un extra de procesamiento.

Para gestionar esta división de núcleos híbridos, se ha integrado un "Thread Director", un elemento que se encarga de determinar qué núcleo procesa cada tarea. Las compañías, además, ha modificado cómo funciona la caché de sus procesadores:

- Cada núcleo de rendimiento tiene su propia caché L2.
- Cada cluster de núcleos de eficiencia tiene una "piscina" de memoria L2 común, de la que beben todos los núcleos que sean partícipes.
- Tanto los núcleos de rendimiento como los de eficiencia tienen acceso a una "piscina" de memoria L3 común para todos ellos.

Otros de los cambios que impactarán en el desempeño de las CPUs es el aumento de velocidad y una mayor cantidad de memoria Caché, compatibilidad con memorias DDR6 y con la interfaz PCIe 6.0.

PCIe PCI Express (Peripheral Component Interconnect Express), abreviado como PCIe, es una tecnología de conexión de Hardware utilizada para la comunicación de alta velocidad entre diferentes componentes de un equipo informático. Este estándar se ha convertido en la interfaz más habitual para la conexión de tarjetas de expansión, como tarjetas gráficas que sirven para correr juegos, tarjetas de sonido, tarjetas de red y dispositivos de almacenamiento de alta velocidad.

Una de las ventajas destacadas de PCI Express es su arquitectura de canales independientes, que permiten la transferencia simultánea de datos en ambos sentidos. Cada carril tiene una tasa de transferencia específica, medida en gigabits por segundo (Gbps), y la capacidad de un slot PCIe se expresa como el número de carriles que tiene. Esto se traduce en un ancho de banda total mayor, lo que facilita la conexión de dispositivos que requieren altas tasas de transferencia, como las tarjetas gráficas modernas o los dispositivos de almacenamiento de última generación.

Los diferentes tipos de ranuras de PCI Express según su tamaño PCIe X1 carriles 1, pines 18, PCIe x4 carriles 4, pines 32, PCIe x8 carriles 8, pines 49, PCIe x16 Carriles 16, pines 82.

Adicionalmente, también es interesante fijarse en las diferentes versiones que se han ido lanzando desde que PCIe se lanzó al mercado:

- PCIe 1.0 ancho banda 8 GB/s, velocidad de transferencia 2.5 GT/s

- PCIe 2.0 ancho banda 16 GB/s, velocidad de transferencia 5 GT/s
- PCIe 3.0 ancho banda 32 GB/s, velocidad de transferencia 8 GT/s
- PCIe 4.0 ancho banda 64 GB/s, velocidad de transferencia 16 GT/s
- PCIe 5.0 ancho banda 128 GB/s, velocidad de transferencia 32 GT/s
- PCIe 6.0 ancho banda 256 GB/s, velocidad de transferencia 64 GT/s

Características Arquitectónicas los procesadores Intel x86 admiten un formato de precisión extendido de 80 bits con un significado de 64 bits, que es compatible con el especificado en el estándar IEEE. Cuando un compilador usa este formato con registros de 80 bits para acumular sumas y productos internos, está trabajando efectivamente con un redondeo unitario de 2^{-64} en vez de 2^{-53} para precisión doble, dando límites de error más pequeños en un factor de hasta $2^{11} = 2048$.

Algunos procesadores Intel y AMD tienen una operación fusionada de multiplicación y suma (FMA), que calcula una multiplicación y una suma combinadas $x + yz$ con un error de redondeo en lugar de dos. Esto da como resultado una reducción en los límites de error por un factor 2.

Las operaciones FMA de bloques de precisión mixta $D = C + AB$, con matrices A, B, C y D de tamaño fijo, están disponibles en las unidades de procesamiento tensorial de Google, las GPU NVIDIA y en la arquitectura ARMv8-A. Para entradas de precisión media, estos dispositivos pueden producir resultados de calidad de precisión simple, lo que puede proporcionar un aumento significativo en la precisión cuando los bloques FMA se encadenan para formar un producto matricial de dimensión arbitraria.

Meltdown y Spectre El tres de enero del 2018 se dio a conocer al público, que 6 meses antes se habían detectado dos distintos fallos en los procesadores de los equipos de cómputo, comunicaciones y redes de internet que usamos. Esto para dar tiempo a los desarrolladores de procesadores y de sistemas operativos de implementar estrategias para mitigar el problema. Estos son problemas de diseño de los procesadores de Intel, AMD, IBM POWER y ARM, esto significa que procesos con privilegios bajos -aquellos

que lanzan las aplicaciones de usuarios convencionales- podían acceder a la memoria del Kernel del sistema operativo[159].

Un ataque que explota dicho problema permitiría a un Software malicioso espiar lo que están haciendo otros procesos y también espiar los datos que están en esa memoria en el equipo de cómputo (o dispositivo móvil) atacado. En máquinas y servidores multiusuario, un proceso en una máquina virtual podría indagar en los datos de los procesos de otros procesos en ese servidor compartido.

Ese primer problema, es en realidad solo parte del desastre. Los datos actuales provienen especialmente de un grupo de investigadores de seguridad formados por expertos del llamado Project Zero[160] de Google. Ellos han publicado los detalles de dos ataques (no son los únicos[161]) basados en estos fallos de diseño. Los nombres de esos ataques son Meltdown y Spectre. Y en un sitio Web dedicado a describir estas vulnerabilidades destacan que "aunque los programas normalmente no tienen permiso para leer datos de otros programas, un programa malicioso podría explotar Meltdown, Spectre y apropiarse de secretos almacenados en la memoria de otros programas". Como revelan en su estudio, la diferencia fundamental entre ambos es que Meltdown permite acceder a la memoria del sistema, mientras que Spectre permite acceder a la memoria de otras aplicaciones para robar esos datos.

Ya que Meltdown y Spectre son problemas de diseño en los procesadores, no es posible encontrar solución por Hardware para los procesadores existentes y dado que constantemente aparecen nuevas formas de explotar dichos

[159] En GNU/Linux, el Kernel (si usamos una versión actualizada) nos indica las fallas del procesador a las que es vulnerable, usando:

```
$ cat /proc/cpuinfo
$ lscpu
```

[160] https://googleprojectzero.blogspot.com/

[161] Entre las distintas vulnerabilidades detectadas y sus variantes resaltan: Meltdown (AC, DE, P, SM, SS, UD, GP, NM, RW, XD, BR, PK, BND), Spectre (PHT, BTB, RSB, STL, SSB, RSRE), PortSmash, Foreshadow, Spoiler, ZombieLoad (1 y 2), Kaiser, RIDL, Plundervolt, LVI, Take a Way, Collide+Probe, Load+Reload, LVI-LFB, MSD, CSME, RYZENFALL (1, 2, 3, 4), FALLOUT (1, 2, 3) , CHIMERA (FW, HW), MASTERKEY (1, 2, 3), SWAPGS, ITLB_Multihit, SRBDS, L1TF, etc. Más información en:

https://cve.mitre.org
https://meltdownattack.com/

fallos, la única manera de mantener el equipo de cómputo, comunicaciones y redes de internet a salvo es mediante Software que debe implementar las soluciones en los sistemas operativos. En particular en el Kernel de Linux se trabaja en parchar en cada versión del Kernel todos los fallos reportados, por esto y por otra gama de fallos e inseguridades es necesario mantener siempre el sistema operativo y sus aplicaciones actualizadas.

Como se había comentado anteriormente, estos problemas de diseño afectan a todos los procesadores Intel, AMD, IBM POWER y ARM. Eso incluye básicamente a todos los procesadores que están funcionando al día de hoy[162] en nuestros equipos, ya que estos procesadores llevan produciéndose desde 1995. Afecta a una amplia gama de sistemas.

En el momento de hacerse pública su existencia se incluían todos los dispositivos que no utilizasen una versión convenientemente parcheada de IOS, GNU/Linux, MacOS, Android, Windows y Android. Por lo tanto, muchos servidores y servicios en la nube se han visto impactados, así como potencialmente la mayoría de dispositivos inteligentes y sistemas embebidos que utilizan procesadores con arquitectura ARM (dispositivos móviles, televisores inteligentes y otros), incluyendo una amplia gama de equipo usado en redes. Se ha considerado que una solución basada únicamente en Software para estas fallas alenta los equipos de cómputo entre un 20 y un 40 por ciento dependiendo de la tarea que realizan y el procesador del equipo.

Memoria RAM La memoria RAM (Random Access Memory) o memoria de acceso aleatorio es un componente físico de nuestro ordenador, generalmente instalado sobre la misma placa base. La memoria RAM es extraíble y se puede ampliar mediante módulos de distintas capacidades.

La función de la memoria RAM es la de cargar los datos e instrucciones que se ejecutan en el procesador. Estas instrucciones y datos provienen del sistema operativo, dispositivos de entrada y salida, de discos duros y todo lo que está instalado en el equipo.

En la memoria RAM se almacenan todos los datos e instrucciones de los programas que se están ejecutando, estas son enviadas desde las unidades de almacenamiento antes de su ejecución. De esta forma podremos tener disponibles todos los programas que ejecutamos. Se llama memoria de acceso aleatorio porque se puede leer y escribir en cualquiera de sus posiciones de

[162] Solo en el año 2021 se detectaron 16 vulnerabilidades en procesadores INTEL y 31 en los procesadores AMD.

memoria sin necesidad de respetar un orden secuencial para su acceso.

De forma general existen o han existido dos tipos de memorias RAM. Las de tipo asíncrono, que no cuentan con un reloj para poder sincronizarse con el procesador. Y las de tipo síncrono que son capaces de mantener la sincronización con el procesador para ganar en eficacia y eficiencia en el acceso y almacenamiento de información en ellas. Veamos cuales existen de cada tipo.

Memorias de Tipo Asíncrono o DRAM las primeras memorias DRAM (Dinamic RAM) o RAM dinámica eran de tipo asíncrono. Se denomina DRAM por su característica de almacenamiento de información de forma aleatoria y dinámica. Su estructura de transistor y condensador hace que para que un dato quede almacenado dentro una celda de memoria, será necesario alimentar el condensador de forma periódica.

Estas memorias dinámicas eran de tipo asíncrono, por lo que no existía un elemento capaz de sincronizar la frecuencia del procesador con la frecuencia de la propia memoria. Esto provocaba que existiera menor eficiencia en la comunicación entre estos dos elementos.

Memorias de Tipo Síncrono o SDRAM a diferencia de las anteriores esta memoria RAM dinámica cuenta con un reloj interno capaz de sincronizar esta con el procesador. De esta forma se mejoran notablemente los tiempos de acceso y la eficiencia de comunicación entre ambos elementos. Actualmente todos nuestras computadoras cuentan con este tipo de memorias operando en ellos. Las principales tipos de memoria son: DDR, DDR2, DDR3, DDR4 y la nueva DDR5. Donde las tasas de transferencia (GB/s) son: DDR (2.1 - 3.2), DDR2 (4.2 - 6.4), DDR3 (8.5 - 14.9), DDR4 (17 - 25.6) y DDR5 (38.4 - 51.2).

Aparte las características propias de cada una de las diferentes memorias DDR, la característica más importante es que, por ejemplo, en la memoria DDR4 cuatro cores pueden acceder simultáneamente a ella y en la DDR5 serán cinco cores.

Caché L1, L2 y L3 La memoria Caché es otra de las especificaciones importantes de los procesadores, y sirve de manera esencial de la misma manera que la memoria RAM: como almacenamiento temporal de datos. No obstante, dado que la memoria Caché está en el procesador en sí, es mucho

más rápida y el procesador puede acceder a ella de manera más eficiente, así que el tamaño de esta memoria puede tener un impacto bastante notable en el rendimiento, especialmente cuando se realizan tareas que demandan un uso intensivo del CPU como en el cómputo de alto desempeño o cómputo científico.

La Caché se divide en diferentes jerarquías de acceso:

- La Caché L1 es el primer sitio donde la CPU buscará información, pero también es la más pequeña y la más rápida, a veces para mayor eficiencia, la Caché L1 se subdivide en L1d (datos) y L1i (instrucciones), actualmente los procesadores modernos en cada core tiene su propio cache de datos e instrucciones.

- La Caché L2 suele ser más grande que la L1 pero es algo más lenta. Sin embargo, por norma general es la que mayor impacto tiene en el rendimiento, este también está incluido en cada core.

- La Caché L3 es mucho más grande que las anteriores, y generalmente se comparte entre todos los núcleos del procesador (a diferencia de las anteriores, que normalmente van ligadas a cada core). Este tercer nivel es en el que buscará el procesador la información tras no encontrarla en la L1 y L2, por lo que su tiempo de acceso es todavía mayor.

Para poner en contexto la relevancia de la memoria Caché, supongamos que el acceso a los datos de la memoria Caché L1 por el procesador es de dos ciclos de reloj, el acceso a la memoria Caché L2 es de 6 ciclos de reloj, el acceso a la memoria Caché L3 es de 12 ciclos y el acceso a la RAM es de 32 ciclos de reloj.

Además supongamos que la operación suma y resta necesitan de 2 ciclos de reloj para completar la operación una vez que cuente con los datos involucrados en dicha operación, que la multiplicación requiere 4 ciclos de reloj para completar la operación, la división necesita 6 ciclos de reloj para completar la operación y estamos despreciando el tiempo necesario para poner los datos del Caché L1 a los registros del procesador para poder iniciar el cálculo, así también despreciamos el tiempo requerido para sacar el resultado de los registros del procesador al Caché L1.

Esto nos da una idea del número máximo teórico de operaciones básicas que un procesador puede realizar por segundo dependiendo de la velocidad

de reloj de la CPU[163].

Si nosotros necesitamos hacer la multiplicación de una matriz $\underline{\underline{A}}$ es de tamaño $n \times n$ por un vector $\underline{u}$ de tamaño n y guardar el resultado en el vector $\underline{f}$ de tamaño n. Entonces algunos escenarios son posibles:

1. Si el código del programa cabe en el Caché L1 de instrucciones y la matriz $\underline{\underline{A}}$, los vectores $\underline{u}$ y $\underline{f}$ caben íntegramente en el Caché L1 de datos, entonces el procesador estará siendo utilizado de forma óptima al hacer los cálculos pues no tendrá tiempos muertos por espera de datos.

2. Si el código del programa cabe en el Caché L1 de instrucciones y los vectores $\underline{u}$ y $\underline{f}$ caben íntegramente en el Caché L1 de datos pero la matriz $\underline{\underline{A}}$ está dispersa entre los Cachés L1 y L2, entonces el procesador estará teniendo algunos tiempos muertos mientras carga la parte que necesita de la matriz del Caché L2 a L1 para hacer los cálculos y utilizado de forma óptima el procesador mientras no salga del Caché L1.

3. Si el código del programa cabe en el Caché L1 de instrucciones y los vectores $\underline{u}$ y $\underline{f}$ caben íntegramente en el Caché L1 de datos pero la matriz $\underline{\underline{A}}$ está dispersa entre los Cachés L1, L2 y L3, entonces el procesador estará teniendo muchos tiempos muertos mientras carga la parte que necesita de la matriz del Caché L3 y L2 a L1 para hacer los cálculos resultando en mediana eficiencia en el uso del procesador.

4. Si el código del programa cabe en el Caché L1 de instrucciones y los vectores $\underline{u}$ y $\underline{f}$ caben íntegramente en los Cachés L3, L2 y L1 pero los datos de la matriz $\underline{\underline{A}}$ está dispersa entre la RAM y los Cachés L3, L2 y L1, entonces el procesador estará teniendo un exceso de tiempos muertos mientras carga la parte que necesita de la matriz de la RAM a los Cachés L3, L2 y L1 para hacer los cálculos resultando en una gran pérdida de eficiencia en el uso del procesador.

[163] Por ejemplo en un procesador AMD Ryzen 9 3900X con 12 Cores (2 Threads por Core) por procesador emulando un total de 24 Cores, corre a una frecuencia base de 3,340 MHz, con una frecuencia mínima de 2,200 MHz y máxima de 4,917 Mhz, con Caché L1d de 384 KiB, L1i de 384 KiB, Caché L2 de 6 MiB y Caché L3 de 64 MiB.

Además, debemos recordar que la computadora moderna nunca dedica el cien por ciento del CPU a un solo programa, ya que los equipos son multitarea[164] y multiusuario[165] por lo que la conmutación de procesos (que se realiza cada cierta cantidad de milisegundos) degrada aún más la eficiencia computacional de los procesos que demandan un uso intensivo de CPU[166].

Last Level Cache se le llama Last Level Cache siempre al último nivel de Caché de una CPU, existen dos tipos:

- Last Level Cache Estándar.
- Victim Cache.

Una Victim Cache no actúia como la Caché de último nivel de una CPU, sino que en ese caso lo hace el penúltimo nivel y en la Victim Cache acaban los últimos datos descartados de la Caché y que han sido volcados en la RAM, los cuales son copiados en la Victim Cache para poder acceder a ellos más rápido.

[164] Cuentan con la capacidad para ejecutar varios procesos simultáneamente en uno o más procesadores, para ello necesitan hacer uso de la conmutación de tareas, es decir, cada cierto tiempo detiene el programa que está corriendo y guardan sus datos, para poder cargar en memoria otro programa y sus respectivos datos y así reiniciar su ejecución por un período determinado de tiempo, una vez concluido su tiempo de ejecución se reinicia la conmutación de tareas con otro proceso.

[165] Se refiere a todos aquellos sistemas operativos que permiten el empleo de sus procesamientos y servicios al mismo tiempo. Así, el sistema operativo cuenta con la capacidad de satisfacer las necesidades de varios usuarios al mismo tiempo, siendo capaz de gestionar y compartir sus recursos en función del número de usuarios que estén conectados a la vez.

[166] Actualmente existen una gran cantidad de distribuciones de GNU/Linux que vienen muy optimizadas intentando conseguir la mejor desenvoltura de su arquitectura y configuraciones de serie. En el caso de la configuración por omisión de Debian GNU/Linux y Ubuntu, están pensadas para que sean lo más robusta posible y que se use en todas las circunstancias imaginables, por ello están optimizadas de forma muy conservadora para tener un equilibrio entre eficiencia y consumo de energía. Pero es posible agregar uno o más Kernels GNU/Linux generados por terceros que contenga las optimizaciones necesarias para hacer más eficiente y competitivo en cuestiones de gestión y ahorro de recursos del sistema.

Hay varias opciones del Kernel GNU/Linux optimizado (Liquorix viene optimizado para multimedia y Juegos, por otro lado XanMod tiene uno para propósito general, otro aplicaciones críticas en tiempo real y otro más para cálculos intensivos) de las últimas versiones estable del Kernel.

Smart Cache la Smart Cache (o Caché) es esencialmente L3 pero optimizada por Intel para ser más eficiente a la hora de compartir la información en los núcleos de la CPU. A efectos prácticos, se comporta de igual manera que la Caché L3.

Disco Son dispositivos no volátiles (los hay del orden de 24 TB y continuamente incrementan su capacidad[167]), lo que significa que retienen datos incluso cuando no tienen energía. La información almacenada permanece segura e intacta a menos que el disco duro sea destruido o interferido. La información se almacena o se recupera de manera aleatoria en lugar de acceso secuencial. Esto implica que se puede acceder a los bloques de datos en cualquier momento sin necesidad de pasar por otros bloques de datos.

Actualmente, podemos agrupar los discos duros disponibles en cuatro tipos:

- Parallel Advanced Technology Attachment (PATA)
- Serial ATA (SATA)
- Interfaz de sistema de computadora pequeña (SCSI)

[167] Durante años la tecnología más popular entre los fabricantes ha sido la PMR (Perpendicular Magnetic Recording), también conocida como CMR (Conventional Magnetic Recording). A esta tecnología luego se le sumó la variante SMR (Shingled Magnetic Recording), que lograba aumentar la densidad de grabación, pero lo hacía sacrificando velocidad de transferencia y fiabilidad de las operaciones.

Western Digital ha creado la tecnología ePMR (energy-assisted Perpendicular Magnetic Recording) que permite ofrecer mayores densidades de grabación y, según este fabricante, mejorar la fiabilidad de las escrituras y evitar así los sacrificios que había que hacer con SMR (en los últimos tiempos han aparecido unidades de 20 y 24 TB basadas en dicha tecnología).

Más interesante aún es el sistema de grabación MAMR (Microwave-Assisted Magnetic Recording) que hace uso de microondas para calentar el medio de almacenamiento y así lograr mejorar densidad de grabación y fiabilidad de lecturas y escrituras. Hace años ya prometían que gracias a esta tecnología contaríamos con unidades de 40 TB en 2025, pero parece que dicho logro aún tardará en llegar.

Esta otra opción es una alternativa a las microondas, pero en HAMR (Heat-Assisted Magnetic Recording) el proceso de calentar el medio de almacenamiento lo realiza un láser. Toshiba prometió lanzar unidades de más de 30 TB en 2024, mientras que Seagate también quería ofrecer esa capacidad de forma inminente para luego dar el salto a unidades de 40 TB e incluso a los 100 TB que plantean para 2030. Ahí es donde probablemente entre en acción la evolución de HAMR+, que tratará de exprimir aún más la densidad de grabación.

- Adjunto de tecnología avanzada paralela
- Unidades de estado sólido (SSD)

En promedio, las velocidades máximas de los discos actuales son:

- Disco SATA3 de 5,400 RPM, Lectura: 102 MB/s, Escritura: 96 MB/s
- Disco SATA3 de 7,200 RPM, Lectura: 272 MB/S, Escritura: 200 MB/s
- Disco SSD SATA, Lectura 550 MB/s, Escritura 520 MB/s
- Disco SSD NVMe, Lectura 6,600 MB/s, Escritura 5,500 MB/s
- Disco SSD PCI 5.0, Lectura 13,000 MB/s, Escritura 12,000 MB/s
- Unidad Flash USB[168] 2.0, 35 MB/s
- Unidad Flash USB 3.0 o 3.1 gen 1, 5 Gbit/s
- Unidad Flash USB 3.0 o 3.1 gen 2, 10 Gbit/s
- Unidad Flash USB 3.2 gen 2x2, 20 Gbit/s

Disco de Estado Sólido SSD Estos son los últimos avances en tecnología de almacenamiento que tenemos en la industria de las computadoras. Son totalmente diferentes de las otras unidades en que no consisten en partes móviles. Tampoco almacenan datos utilizando magnetismo. En su lugar, hacen uso de la tecnología de memoria flash, circuitos integrados o dispositivos semiconductores para almacenar datos de forma permanente, al menos hasta que se borren. Estas son algunas de sus ventajas.

- Acceso a datos más rápido
- Menos susceptible a los golpes
- Menores tiempos de acceso y latencia

[168] Los colores en los puertos USB son: USB 1.X blanco (12 Mbps), USB 2.X Negro (480 Mbps), USB 3.0 Azul oscuro (5 Gbps), USB 3.1 Azul claro (10 Gbps), USB 3.2 Rojo (20 Gbps). En el caso del USB de color Amarillo, esté es un puerto de carga aún con el dispositivo apagado.

- Menos consumo de energía

Los SSD actuales están disponibles tanto en versiones SATA como en versiones M.2, U.2 y en formato de tarjeta PCI Express 4.0. Los tres últimos hacen uso del protocolo NVMe y la interfaz PCI Express 4.0 x4, lo que les permite superar los 6,600 MB/s de velocidades de lectura y escritura, frente a los 550 MB/s que suelen alcanzar como máximo las unidades SATA. La nueva versión PCI 5.0 ofrece un ancho de banda de 32 GT/s el doble de PCI 4.0, permitiendo discos SSD con 13,000 MS/s de velocidad de lectura secuencial y realizar hasta 2,500K operaciones por segundo de lectura aleatoria y tamaño máximo 15.36 TB a un precio exorbitante.

Tarjetas microSD Cuando compramos una tarjeta microSD para ampliar el almacenamiento de nuestro Smartphone, cámara, tableta o cualquier otro dispositivo electrónico la mayoría de la gente normalmente solo se fija en la capacidad de almacenamiento. Ahora bien, ¿qué significan todas esas etiquetas y nombres que llevan adscritas las microSD? ¿Cuál es la diferencia entre una microSDXC y una microSDHC? ¿Es mejor una UHS-I o una UHS-II? ¿Qué quiere decir que una tarjeta es A1 y V30? A continuación, intentamos aclarar toda esta nomenclatura.

Lo primero que tenemos que tener claro es que cuando analizamos una tarjeta micro SD existen multitud de factores que limitan su velocidad y capacidad de almacenamiento. Su rendimiento depende de factores como el tipo de tarjeta que estamos usando, su clase, y otros detalles como el tipo de bus o el número de operaciones que puede realizar por segundo.

Tipos de tarjetas microSD actualmente existen 4 generaciones distintas de tarjetas de memoria microSD. Cuanto más modernas sean, mayores velocidades y almacenamiento podrán ofrecer:

- Tarjetas micro SD (Secure Digital): Estas son las memorias de primera generación. Las desarrolló el fabricante SanDisk y fueron las primeras en utilizar el formato de 15 x 11 x 1 milímetros. Su capacidad máxima es de 32 GB.

- Tarjetas micro SDHC (Secure Digital High Capacity): Tarjetas de segunda generación. Cuentan con un bus de datos mejorado que permite alcanzar velocidades superiores, aunque su capacidad máxima sigue siendo de 32 GB.

- Tarjetas micro SDXC (Secure Digital Extended Capacity): Estas micro SD utilizan un sistema de archivos exFAT y su velocidad de transferencia puede llegar hasta los 312 MB/s. Su capacidad de almacenamiento puede llegar hasta los 2 TB y es el tipo de tarjeta más común utilizado a día de hoy.

- Tarjetas micro SDUC: Estas son las tarjetas de memoria más modernas y punteras. Utilizan el sistema de archivos exFAT y permiten almacenar entre 2 TB y 128 TB de datos.

Evidentemente con esto no es suficiente. Si queremos tener una idea aproximada de la velocidad de la micro SD tendremos que fijarnos en aspectos como la clase y tipo de bus que emplea.

La clase es una característica que nos indica la velocidad de transferencia de datos mínima de la tarjeta de memoria. Actualmente hay 4 tipos de clase diferentes:

- Clase 2: Velocidad mínima de 2 MB/s

- Clase 4: Velocidad mínima de 4 MB/s

- Clase 6: Velocidad mínima de 6 MB/s

- Clase 10: Velocidad mínima de 10MB/s

Hoy en día la mayoría de tarjetas micro SD son de clase 10, ya que son capaces de transferir más de 10 MB/s y superan esa cifra fácilmente.

El bus determina la velocidad de la interfaz de la tarjeta de memoria, y nos puede servir como indicativo para conocer la rapidez con la que se pueden leer y escribir los datos:

- Bus estándar: Su velocidad de transferencia alcanza hasta los 12.5 MB/s. Es el tipo de bus utilizado en tarjetas de clase 2, 4 y 6.

- Bus de alta velocidad (High Speed): Se utiliza en tarjetas de clase 10 y alcanza una velocidad de hasta 25 MB/s.

- Bus Ultra High Speed (UHS): Estos son los buses con la interfaz más rápida, y existen varios tipos:

  - UHS-I: Hay dos tipos de buses UHS-I. Por un lado, tenemos el UHS-I clase 1 (U1) que alcanza velocidades de 50 MB/s. Y luego tenemos el UHS-I clase 3 (U3) que llega hasta los 104 MB/s.
  - UHS-II: Alcanza velocidades de transferencia hasta 312 MB/s.
  - UHS-III: Velocidades de transferencia de datos que alcanzan hasta los 624 MB/s.

- SD-Express: Este es el tipo de bus más potente de todos, llegando hasta los 985 MB/s.

Como referencia, te interesará saber que actualmente la mayoría de tarjetas micro SD de gama media utilizan un bus UHS-I de clase 3 (U3) con velocidades de lectura de hasta 104 MB/s.

Otro factor importante es la velocidad de lectura y escritura aleatoria (IOPS) u operaciones por segundo que puede realizar una tarjeta. Este dato determina el rendimiento mínimo en la lectura y escritura aleatoria de la SD:

- Clase de rendimiento de aplicación A1: Las tarjetas A1 tienen una velocidad mínima de lectura aleatoria de 1,500 IOPS, y una velocidad mínima de escritura aleatoria de 500 IOPS.

- Clase de rendimiento de aplicación A2: Las tarjetas A2 ofrecen velocidades superiores, con 4,000 IOPS de lectura y 2,000 IOPS de escritura.

Normalmente con una tarjeta A1 es más que suficiente para tareas del día a día, aunque si necesitamos un rendimiento superior, por ejemplo, para ejecutar aplicaciones desde la SD o jugar a videojuegos, las tarjetas A2 ofrecen un mejor rendimiento.

La velocidad de escritura aleatoria (A1 y A2) es un dato relevante para los Smartphones y tabletas, pero si tenemos una cámara de grabación, una Action camera o un Dron, la característica en la que nos tenemos que fijar es en el Velocidad de escritura secuencial (sistema V) que utiliza (en inglés, Video Speed Class). O dicho de otra forma, en su velocidad de escritura secuencial.

Esta característica nos indica la cantidad de datos que se pueden grabar en la micro SD de forma constante sin bajar de una velocidad mínima. Esto resulta esencial cuando queremos grabar vídeos en alta y ultra-alta definición:

- V30 (Video Speed Class 30): Velocidad de escritura mínima de 30 MB/s
- V60 (Video Speed Class 60): Velocidad de escritura mínima de 60 MB/s
- V90 (Video Speed Class 90): Velocidad de escritura mínima de 90 MB/s

Por ejemplo, si vamos a grabar vídeo en resolución 4K directamente en la tarjeta micro SD, es necesario que la velocidad V sea lo máximo posible, especialmente si vamos a utilizar un amplio BitRate con bajos niveles de compresión, o calidades superiores como el 8K.

Cintas Magnéticas En el año 2010, se comunicó que todos los datos utilizados para el proyecto del satélite Nimbus se recuperaron de cintas que en ese momento tenían 46 años. A partir de dicho comunicado se extendió el uso de la cinta magnética para almacenamiento de datos en todo el mundo.

En un mundo altamente digital, la cinta magnética es una de las pocas tecnologías que utiliza señales analógicas para mover parte de los datos, en su esencia, la cinta se parece mucho a un HDD, utiliza materiales de base magnética, pero en este caso, la cinta es literalmente una base de material generalmente nailon que tiene un revestimiento magnético. Y en lugar de un disco giratorio, la cinta entra, está enhebrada. Puede parecer una cinta VHS, pero es mucho más robusta.

La cinta se mueve linealmente hacia la unidad de cinta y hacia el cartucho de cinta. Para escribir, el cabezal de la cinta toma señales electrónicas y crea un mini campo magnético que puede cambiar la polaridad del material de la película para formar un patrón de ceros y unos. Una vez que los datos se escriben en la cinta, no se pueden cambiar (pero se pueden borrar y reescribir).

La inmutabilidad y las capacidades de encriptación de la cinta, así como la simplicidad de crear un espacio para almacenarla en una bóveda hacen de la cinta un arma clave para asegurar que los datos sobrevivan frente al Ransomware. Uno de los grandes productores de cintas en la actualidad es IBM, los cuales argumentan que esta función hace que la cinta sea el medio ideal para almacenar datos de archivo a los que no es necesario acceder con frecuencia.

La cinta también puede servir como una copia de seguridad y de versiones fuera de línea de archivos importantes o confidenciales que son resistentes a los ataques cibernéticos. Los tipos de datos que permanecen en la cinta abarcan registros financieros, registros médicos, información de identificación personal y documentos que forman parte de una retención legal de múltiples gobiernos.

Una sola cinta mide aproximadamente 3 pulgadas por 3 pulgadas y 3/4 de pulgada de grosor. Es más pequeño que una unidad de disco duro (HDD), pero pesa alrededor de 0.6 kilogramos. Un cartucho puede almacenar 18 Terabytes de datos sin comprimir y 45 Terabytes comprimidos. IBM está trabajando para duplicar esta capacidad en la próxima generación de la tecnología. En cuanto a la velocidad de recuperación, se obtiene un flujo de datos de una unidad de cinta de 1,000 Megabits por segundo, comprimidos.

Una biblioteca de cintas puede variar en tamaño desde algo que puede poner en su escritorio hasta algo que es del tamaño de un refrigerador pequeño (alrededor de 8 pies cuadrados). La pequeña biblioteca del tamaño de un refrigerador tiene capacidad para 1584 cartuchos. IBM promociona que su biblioteca Diamondback será la biblioteca de cintas más densa del mercado. Podrá contener 69 Petabytes de información mientras ocupa menos de 8 pies cuadrados de espacio.

La cinta magnética supera al disco duro y al flash en cuanto a longevidad, costo financiero y costo de huella de carbono, pero pierde en velocidad de acceso. Las cintas no son recomendables para poner datos de producción en vivo o incluso copias de seguridad, pero son perfectas para cualquier información a la que se acceda con poca frecuencia y que deba conservarse durante mucho tiempo, como registros médicos o datos de archivo.

Es conocido que muchas empresas han usado y seguirán usando cinta magnética en sus operaciones, entre las que destacan las hiperescalas (empresas que han crecido tanto que ofrecen sus propias infraestructuras o tienen datos masivos como resultado de su infraestructura) siempre necesitan muchas formas diferentes de tecnología para manejar la variedad de datos que ingresan a sus sistemas para alimentar una gama de servicios. Entre otras destacan: Bancos, Gobiernos, Milicia y organizaciones como CERN, así como corporaciones como Amazon, Google, Meta, Baidu, Alibaba y Tencent.

Tarjeta Gráfica La tarjeta gráfica o tarjeta de vídeo es un componente que viene integrado en la placa base de la computadora o se instala aparte

para ampliar sus capacidades. Concretamente, esta tarjeta está dedicada al procesamiento de datos relacionados con el vídeo y las imágenes que se están reproduciendo en la computadora.

Procesador Gráfico GPU el corazón de la tarjeta gráfica es la GPU o Unidad de procesamiento gráfico, un circuito muy complejo que integra varios miles de millones de transistores diminutos y puede tener desde uno a miles de núcleos (ya es común encontrar computadoras personales con tarjeta de 10496 cores y 24 GB de GRAM) que tienen capacidad de procesamiento independiente. De la cantidad y capacidad de estos núcleos dependerá la potencia.

Así como los procesadores centrales de las CPU, están diseñados con pocos núcleos pero altas frecuencias de reloj, las GPU tienden al concepto opuesto, contando con grandes cantidades de núcleos con frecuencias de reloj relativamente bajas. Luego tienes la memoria gráfica de acceso aleatorio o GRAM, que son Chips de memoria que almacenan y transportan información entre sí. Esta memoria no es algo que vaya a determinar de forma importante el rendimiento máximo de una tarjeta gráfica, aunque si no es suficiente puede acabar lastrando y limitando la potencia de la CPU.

La idea de usar esa potencia para otros menesteres se denomina GPGPU (General Purpose Computation on Graphics Processing Units) o GPU Computing. En el momento en el que las tarjetas gráficas permiten que se programen funciones sobre su Hardware se empieza a hacer uso de GPGPU. Al principio era necesario utilizar los lenguajes enfocados a la visualización en pantalla (como OpenGL) para realizar otros cálculos no relacionados con los gráficos. Esto implicaba el uso de funciones muy poco flexibles, originalmente diseñadas para otros fines, lo que hacía que la labor de programar para tal fin fuese realmente tediosa y complicada.

Para facilitar el empleo de las tarjetas gráficas para cualquier uso no vinculado con los gráficos, NVidia desarrolló toda una tecnología alrededor de la tarjeta, que permitía usar la misma para cualquier tarea: CUDA. ATI, la principal (y actualmente casi única) competidora, un poco más tarde haría lo propio lanzando su propia tecnología: Stream. En un principio las tarjetas gráficas solo trabajaban con aritmética de 32 bits, pero en la actualidad ya se cuenta con aritmética de 64 bit (para lograr esto, muchas tarjetas usan dos de sus cores de 32 bits para emular uno de 64 bits, reduciendo su número de cores útiles a la mitad).

Procesador Gráfico Integrado muchos procesadores CPU incorporan una o más GPU en su interior, llamada gráfica integrada (iGPU Integrated Graphics Processing Units o APU Accelerated Processing Unit). Generalmente es muy poco potente, pero lo suficiente para realizar tareas básicas como navegar por Internet, ver vídeos, e incluso para algunos juegos básicos, especialmente en las últimas generaciones puesto que cada vez son más potentes. No obstante, en las últimas generaciones de procesadores cada vez se están introduciendo gráficos integrados más potentes, y ya son capaces de manejar varios monitores, resoluciones 4K e incluso son capaces de mover algunos juegos a una tasa digna de FPS.

Tarjeta Gráfica por ejemplo, la tarjeta gráfica de AMD Instinct MI200 cuenta con más de 200,000 cores y 128 GB de HBM2, NVidia GEFORCE RTX 3090 proporciona 10,496 cores y 24 GB de GDDR6x, NVidia A100 cuenta con 80 GB de memoria HBM2 6192 cores y 432 núcleos tensor[169], mientras que la tarjeta NVidia Titan RTX proporciona 130 Tensor TFLOP de rendimiento, 576 núcleos tensores y 24 GB de memoria GDDR6.

Por otra parte, Intel ha desarrollado una aceleradora gráfica Artic Sound-M pensada para centro de datos (especialmente diseñada para juegos en la nube) que utiliza una GPU DG2 Xe-HPG que viene con una configuración de 512 unidades de ejecución lo que equivale a 4,096 Shaders, por ejemplo, esta aceleradora puede manejar hasta 8 Streamings simultáneos de video 4K o más de 30 si el vídeo es en 1080p y cuenta con más de 60 funciones virtualizadas.

En agosto del 2021, se anunció la construcción de la supercomputadora Polaris, acelerado por 2240 GPU NVIDIA A100 Tensor Core, el sistema puede alcanzar casi 1.4 exaflops de rendimiento teórico de IA y aproximadamente 44 petaflops de rendimiento máximo de doble precisión. Polaris, que será construido por Hewlett Packard Enterprise, combinará simulación y aprendizaje automático al abordar cargas de trabajo informáticas de alto rendimiento de inteligencia artificial y con uso intensivo de datos, impulsadas por 560 nodos en total, cada uno con cuatro GPU NVIDIA A100.

Tipos de Redes Según el Medio Físico Si bien, nuestros dispositivos de cómputo pueden funcionar sin conexión de red, estos se ven inmediatamente

[169] Un Tensor core (o núcleos Tensor) calculan la operación de una matriz 4x4 completa, la cual se calcula por reloj. Estos núcleos pueden multiplicar dos matrices FP16 4x4 y sumar la matriz FP32 al acumulador.

limitados. La red nos permite conectarnos a Internet que es el camino por el cual nos conectamos con el mundo. Las formas de conectar nuestros equipos a Internet en un principio fue exclusivamente por red alámbrica, desde ya hace unos años a la fecha se dispone de conexión a red alámbrica e inalámbrica, pero actualmente nuestros dispositivos cuentan casi exclusivamente con conexión inalámbrica.

¿Qué es el ancho de banda? Se trata de la capacidad máxima y la cantidad de datos que se pueden transmitir a través de una conexión (de internet, por ejemplo), en un momento determinado. Algo que debemos tener claro es que el ancho de banda de red es fundamental para la calidad y velocidad de la conexión.

El ancho de banda se mide en *bit/s* o en sus múltiplos *k/bits* o *m/bits* por segundo. Y para la mayoría de los casos, debemos asegurarnos siempre de tener el mayor ancho de banda que nos sea posible, porque de esta manera podremos tener una mejor y más rápida transferencia de datos.

¿Qué es entonces la velocidad de transmisión? Este término se puede definir como la velocidad a la que se transmite la información. Cuando un usuario adquiere un paquete con una empresa prestadora de servicios de internet, recibe, por ejemplo, 10 mbps, 30 mbps, 100 mbps, etc. Y esto se refiere a la cantidad de datos que podemos descargar o subir a la red. Como recomendación, lo indicado es que para que la velocidad pueda existir será necesario tener un ancho de banda igual o superior a la velocidad contratada en el paquete de servicio.

Normalmente vemos en los anuncios de todos los operadores, que estos ofrecen una cantidad cualquiera de megas de navegación; este valor numérico corresponde a la velocidad de descarga únicamente. Para encontrar la velocidad de subida, es necesario acceder a un test que nos revele cuál es el resultado y si lo que nos prometen, es verdad o no.

¿Qué es la latencia? Es el tiempo total que transcurre desde que enviamos una información, hasta que la misma llega a un receptor. Su valor de medición se hace en milisegundos, también se conoce como *Ping* y está presente en actividades que realiza cotidianamente como jugar en línea o hacer videollamadas.

¿Altera la latencia la velocidad de la conexión? La velocidad de conexión influye en la latencia una vez que pasamos de un rango predeterminado. Poniéndolo en un ejemplo, si tenemos una conexión a Internet de 1 Mbps y la comparamos con una conexión de 100 Mbps, dependiendo del tamaño del paquete se notará una mejora grande en la velocidad. Otros factores que importan a la hora de hablar de latencia son el estar conectado a internet por Wi-fi o un cable, si tiene servicio de fibra óptica o qué tanta distancia hay entre su ordenador y un Router, etc.

La latencia en la conexión también es la suma de otros retardos:

- De procesamiento: se define en el tiempo que tardan los Routers en examinar la cabecera y a su vez, la respuesta en determinar a dónde hay que enviar cualquier paquete haciendo una previa comprobación de sus tablas de enrutamiento.

- De cola: tiempo de espera del paquete para poder ser transmitido a través de un enlace físico. Cabe anotar que no podemos saber previamente si va a haber un retardo de cola o no, ya que este cambia en tiempo real.

- De transmisión: es el tiempo que tarda el paquete en arribar hasta el siguiente nodo o destino final.

- De propagación: es el tiempo que tarda un bit en propagarse desde un punto cualquiera de origen hasta llegar a uno de destino. Su velocidad depende del medio físico por el que se transporte.

El retardo total es la suma de todos los retardos anteriormente enunciados.

Comúnmente se asocia la latencia con la banda de ancha, pero existe una diferencia sustancial entre ambas. Si bien ambas afectan la velocidad de la conexión, la banda ancha permite que se pueda transmitir una gran cantidad de datos, mientras la latencia determina a qué velocidad se transmite esa cantidad de datos.

Si bien, tener red nos permite estar conectados, tenemos una gran limitación por las velocidades de conexión a las que tendremos acceso según el tipo de medio físico que usemos para conectarnos, así como el número de dispositivos con los que compartamos la conexión. Las redes inalámbricas parecen ser omnipresentes, pero las velocidades de interconexión dejan mucho que desear como veremos a continuación.

Redes alámbricas se comunica a través de cables de datos (generalmente basada en Ethernet). Los cables de datos, conocidos como cables de red de Ethernet o cables con hilos conductores (CAT5), conectan computadoras y otros dispositivos que forman las redes. Las redes alámbricas son mejores cuando se necesita mover grandes cantidades de datos a altas velocidades, como medios multimedia de calidad profesional.

Ventajas

- Costos relativamente bajos
- Ofrece el máximo rendimiento posible
- Mayor velocidad - cable de Ethernet estándar hasta 1 Gbps[170]
- Mayor rendimiento de Voz sobre IP.
- Mejores estándares Ethernet en la industria.
- Mayor capacidad de ancho de banda por cables.
- Aplicaciones que utilizan un ancho de banda continúo.

Desventajas

- El costo de instalación siempre ha sido un problema común en este tipo de tecnología, ya que el estudio de instalación, canaletas, conectores, cables y otros suman costos muy elevados en algunas ocasiones.
- El acceso físico es uno de los problemas más comunes dentro de las redes alámbricas. Ya que para llegar a ciertos lugares, es muy complicado el paso de los cables a través de las paredes de concreto u otros obstáculos.
- Dificultad y expectativas de expansión es otro de los problemas más comunes, ya que cuando pensamos tener un número definido de nodos en una oficina, la mayoría del tiempo hay necesidades de construir uno nuevo y ya no tenemos espacio en los Switches instalados.

[170] El término bps se refiere a transmitir Bits por segundo (se requieren 8 Bits para formar un Byte). Por eso el término de 1,000 Mbps es equivalente a 125 MB/s.

Hoy en día se puede hacer la siguiente clasificación de las redes de protocolo Ethernet para cable y fibra óptica[171]:

- Ethernet, que alcanza no más de 10 Mbps de velocidad
- Fast Ethernet, que puede trabajar con hasta 100 Mbps
- Gigabit Ethernet, alcanza hasta 1 Gbps (1000 Mbps aprox.)
- 2.5 y 5 Gigabit Ethernet hasta 2.5 Gbps si usamos cableado Cat 5a y nada menos que 5 Gbps con cableado Cat 6
- 10 Gigabit Ethernet, que puede alcanzar hasta los 10 GMbps
- 1000GbE para alcanzar la Ethernet Terabit (125 Gbytes)

La categoría del cable o el tipo de fibra óptica determina la velocidad máxima soportada por cada tipo de cable, pero aunque haya cables con la misma velocidad hay otros factores que determinan su usabilidad, como el ancho de banda o la frecuencia. La frecuencia o ancho de banda determina la potencia de la red, a mayor frecuencia mayor ancho de banda y menor perdida de datos. Este factor es importante si vamos a conectar varios equipos al mismo cable de red o vamos a hacer una gran tirada de cable, ya que cuanto más largo sea el cable de red más potencia perderá. Siempre tendrá más velocidad un cable corto que un cable largo, pero si el ancho de banda es amplio tardará más metros en perder potencia y velocidad.

[171] El récord mundial en mayo del 2022 de transmisión en fibra óptica es de 1.02 petabits por segundo enviados a través de 51.7 kilómetros (que podría transmitir hasta 10 millones de canales por segundo de vídeo a resolución 8K), que rompe al récord anterior de 319 terabits por segundo sobre una distancia de 1,800 millas (con el estimado de descarga de 80,000 películas simultáneamente en un segundo).

En 2023 los investigadores de Electronics and Computer Engineering de Aston University en U.K. alcanzaron una velocidad de 301 terabits por segundo (Tbps), equivalente a transferir 1,800 películas 4K a través de Internet en un segundo, utilizando cables de fibra óptica existentes. En comparación, la velocidad media de banda ancha fija en los EE. UU. es de 242,38 megabits por segundo (Mbps), según Speed Test.

Los resultados de la prueba, que se realizaron utilizando el tipo de cables de fibra ya tendidos en el suelo, lograron esta velocidad vertiginosa enviando luz infrarroja a través de hilos tubulares de vidrio, que es como funciona generalmente la banda ancha de fibra óptica. Pero aprovecharon una banda espectral que nunca se ha utilizado en sistemas comerciales, llamada "banda E", utilizando dispositivos nuevos hechos a medida.

Ethernet es el estándar que domina la gran mayoría de mercado, presente de manera casi exclusiva tanto en mercado doméstico, como en pequeña y mediana empresa representa también un porcentaje muy significativo en grandes centros de datos. Pero existen otros protocolos que se pueden situar en el mismo nivel de calidad que Gigabit Ethernet como InfiniBand.

InfiniBand es un bus serie bidireccional de comunicaciones de alta velocidad en las que las rápidas comunicaciones entre servidores son críticas para el rendimiento, llegando a ofrecer velocidades de hasta 2.0 Gbps netos en cada dirección del enlace en un nodo simple, 4 Gbps netos en un nodo doble y hasta 8 Gbps netos en un nodo quadruple. Estos nodos a su vez se pueden agrupar en grupos de 4 ó 12 enlaces llegando a velocidades de hasta 96 Gbps netos en un grupo de 12 nodos cuádruples. El factor de velocidad neta viene relacionado con que Infiniband de cada 10 bits que transmite 8 de ellos son datos, basándose en la codificación 8B/10B.

Recientemente se han implementado sistemas en los que ya no se utiliza esta codificación 8B/10B sino la 64B/66B que permite mejorar el porcentaje de datos útiles por trama enviada y que ha permitido los nodos FDR-10 (Fourteen Data Rate-10 a 10 Gbps), FDR (Fourteen Data Rate a 13.64 Gbps) y EDR (Enhanced Data Rate a 25 Gbps). Este último en un grupo de 12 nodos proporciona hasta 300 Gbps. Los últimos desarrollos de Gigabit Ethernet, proporcionan hasta 100 Gbps por puerto.

Estas enormes velocidades de conexión hacen que Infiniband sea una conexión con una muy importante presencia en superordenadores y clústers, por ejemplo del top 500 de superordenadores en 2020, 226 están conectados internamente con Infiniband, 188 lo están con Gigabig Ethernet y el resto con Myrinet, Cray, Fat Tree u otras interconexiones a medida.

Una de las principales ventajas de Infiniband sobre Ethernet es su bajísima latencia, por ejemplo y basándonos en los datos del estudio de Qlogic "Introduction to Ethernet Latency, an explanation to Latency and Latency measurement", la latencia en 10 Gpbs Ethernet se sitúa en 5 microsegundos mientras que la de Infiniband se sitúa por debajo de los 3 microsegundos.

Los sistemas de conmutadores inteligentes InfiniBand de NVIDIA Mellanox ofrecen el mayor rendimiento y densidad de puertos para computación de alto rendimiento (HPC), IA, Web 2.0, Big Data, nubes y centros de datos empresariales. La compatibilidad con configuraciones de 36 a 800 puertos a hasta 200 Gpbs por puerto permite que los clústeres de cómputo y los centros de datos convergentes funcionen a cualquier escala, lo que reduce los costos operativos y la complejidad de la infraestructura.

Redes inalámbricas: es una red en la que dos o más terminales (ordenadores, tabletas, teléfonos inteligentes, etc.) se pueden comunicar sin la necesidad de una conexión por cable. Se basan en un enlace que utiliza ondas electromagnéticas (radio e infrarrojo) en lugar de cableado estándar. Permiten que los dispositivos remotos se conecten sin dificultad, ya se encuentren a unos metros de distancia como a varios kilómetros.

Asimismo, la instalación de estas redes no requiere de ningún cambio significativo en la infraestructura existente como pasa con las redes cableadas. Tampoco hay necesidad de agujerear las paredes para pasar cables ni de instalar porta cables o conectores. Esto ha hecho que el uso de esta tecnología se extienda con rapidez.

Tipos de redes inalámbricas

- LAN Inalámbrica: Red de área local inalámbrica. También puede ser una red de área metropolitana inalámbrica.
- GSM (Global System for Mobile Communications): la red GSM es utilizada mayormente por teléfonos celulares.
- D-AMPS (Digital Advanced Mobile Phone Service): está siendo reemplazada por el sistema GSM.
- Fixed Wireless Data: Es un tipo de red inalámbrica de datos que puede ser usada para conectar dos o más edificios juntos para extender o compartir el ancho de banda de una red sin que exista cableado físico entre los edificios.
- Wi-Fi[172]: es uno de los sistemas más utilizados para la creación de redes

[172] Notemos que una conexión de WiFi tradicional sin obstáculos con una intensidad de banda de 2.4 GHZ puede tener un alcance máximo de 46 metros y para la banda de 5.0 GHz un alcance de 15 metros. Paro hay muchos objetos que interfieren con la señal del Router del WiFi y el sitio en el que colocamos nuestro dispositivo inalámbrico como computadora o teléfono inteligente:

- Superficies y/o objetos de metal o vidrio blindado
- Refrigeradores, lavadoras y radiadores
- Hornos de microondas, cámaras Web, monitores de bebés y teléfonos inalámbricos
- Paredes y muros
- Dispositivos arquitectónicos que funciones como una jaula de Faraday (es un contenedor recubierto por materiales conductores de electricidad como mallas metálicas,

inalámbricas en computadoras, permitiendo acceso a recursos remotos como internet e impresoras. Utiliza ondas de radio.

Ventajas

- La instalación de redes inalámbricas suele ser más económica.
- Su instalación también es más sencilla.
- Permiten gran alcance; las redes hogareñas inalámbricas suelen tener hasta 100 metros desde la base transmisora.
- Permite la conexión de gran cantidad de dispositivos móviles. En las redes cableadas mientras más dispositivos haya, más complicado será el entramado de cables.
- Posibilidad de conectar nodos a grandes distancias sin cableado, en el caso de las redes inalámbricas corporativas.
- Permiten más libertad en el movimiento de los nodos conectados, algo que puede convertirse en un verdadero problema en las redes cableadas.
- Permite crear una red en áreas complicadas donde, por ejemplo, resulta dificultoso o muy caro conectar cables.

Desventajas

- Calidad de Servicio: La velocidad que posee la red inalámbrica no supera la cableada, ya que esta puede llegar a los 10 Mbps, frente a 100 Mbps que puede alcanzar la cableada. Hay que tomar en cuenta la tasa de error debida a las interferencias.

papel aluminio, cajas o cestos de basura de acero que funciona como un blindaje contra los efectos de un campo eléctrico proveniente del exterior).

En caso de que varios dispositivos se conecten a la red, se puede optar por colocar el Router en un punto medio, para que ninguna zona quede sin cobertura. Por otra parte, para una mejor conexión, también procura que el Router se encuentre en una zona elevada, pues esto mejorará el alcance de la señal inalámbrica. Una excelente opción para mejorar la calidad del internet inalámbrico cuando hay obstáculos es utilizar un repetidor de WiFi, este dispositivo es muy útil para amplificar la señal y llevarla a más sitios de nuestra red.

- Costo: En algunos casos, puede ser más barato cablear una casa/oficina que colocar un servicio de red inalámbrica.
- La señal inalámbrica puede verse afectada e incluso interrumpida por objetos, árboles, paredes, espejos, entre otros.

La velocidad máxima de transmisión inalámbrica de la tecnología 802.11b es de 11 Mbps. Pero la velocidad típica es solo la mitad: entre 1.5 y 5 Mbps dependiendo de si se transmiten muchos archivos pequeños o unos pocos archivos grandes. La velocidad máxima de la tecnología 802.11g es de 54 Mbps. Pero la velocidad típica de esta última tecnología es solo unas 3 veces más rápida que la de 802.11b: entre 5 y 15 Mbps. Resumiendo, las velocidades típicas de los diferentes tipos de red son:

Estándar	V.Máxima	V.Practica	Frecuencia	Ancho Banda	Alcance
802.11	2Mbit/s	1Mbit/s	2.4Ghz	22MHz	330 metros
802.11a(WiFi5)	54Mbit/s	22Mbit/s	5.4Ghz	20MHz	390 metros
802.11b	11Mbit/s	6Mbit/s	2.4Ghz	22MHz	460 metros
802.11g	54Mbit/s	22Mbit/s	2.4Ghz	20MHz	460 metros
802.11n	600Mbit/s	100Mbit/s	2.4Ghzy5.4Ghz	20 y 40MHz	820 metros
802.11ac	6.93Gbps	100Mbit/s	5.4Ghz	80 o hasta 160MHz	Poco alcance, pero sin interferencias
802.11ad	7.13Gbit/s	Hasta6Gbit/s	60Ghz	2MHz	300 metros
802.11ah	35.6Mbps	26.7Mbps	0.9Ghz	2MHz	1000 metros
802.11ax(WiFi6)	9,6Gbps	6,9Gbps	2.4Ghz y 5.4Ghz	20MHz	1000 metros

Como puedes ver, los principales factores que influyen en la calidad de una conexión WiFi, son la frecuencia, el ancho de banda y el alcance total. Considerando que, todo esto junto con la velocidad máxima y la velocidad práctica, se congregan en lo que es cada versión de este tipo de conexión a la red. Además, la conexión se degradará inexorablemente con la cantidad de dispositivos conectados y su consumo de datos.

Velocidad de los Proveedores cuando se contrata el servicio de internet, la velocidad de interconexión del mismo depende de cuánto sea el cobro, pero en la mayoría de los casos se tendrá una velocidad de descarga mayor a la velocidad de carga y nuestra red será una intranet (dirección de IP dinámica) compartida con otros miles de usuarios abonados al servicio de internet del proveedor. También es posible contratar una interconexión con

velocidades homogéneas para carga y descarga dedicada, el costo del mismo se puede hasta triplicar con respecto a uno no homogéneo.

En caso de requerir una dirección de internet homologada o pública, el costo de contratar el servicio aumenta considerablemente, pero de esta forma nuestros equipos son visibles en el Internet (esto conlleva un aumento de riesgos al estar nuestros equipos más vulnerables a ataques informáticos).

Información de mi Computadora Si lo que deseamos es un listado detallado del Hardware de nuestro equipo de cómputo, entonces podemos usar cualquiera de estos comandos (que previamente deberemos instalar):

```
$ lscpu
# lshw
# dmidecode
# hwinfo
```

9.2 ¿Que es Computación Paralela?

En el sentido más simple, la computación paralela es el uso simultáneo de múltiples recursos computacionales para resolver un problema computacional:

- Un problema se divide en partes discretas que se pueden resolver simultáneamente
- Cada parte se descompone en una serie de instrucciones
- Las instrucciones de cada parte se ejecutan simultáneamente en diferentes procesadores
- Se emplea un mecanismo global de control/coordinación

¿Por qué se hace programación paralela? El hecho de que la programación paralela sea un paradigma da cuenta de que existe una razón por la cual no ha dejado de ser necesaria o no ha sido totalmente automatizable, igualmente hay otras razones interesantes detrás para entender la existencia, actualidad y contemporaneidad de la programación paralela:

- Ley de Moore: Esta ley propuesta por Gordon E. Moore en 1965 dice resumidamente que el número de transistores en un Chip determinado se doblaría cada dos años. Esto quiere decir un aumento del rendimiento en los procesadores del alrededor del 50%, esto se traduce en escalar la velocidad de reloj de los procesadores, pero esta ley no es fidedigna desde el 2002 dónde solo ha habido un 20%, lo cuál sigue siendo un aumento considerable, sin embargo, no sería suficiente para que todos los avances en computación que se han logrado hasta el día y las necesidades de procesamiento en crecimiento exponencial puedan satisfacerse totalmente.

- Overclocking infinito: El Overclokcing tiene un límite a pesar de que existiera una refrigeración perpetúa y adecuada del procesador. Esto es debido a las corrientes parásitas que impiden una velocidad teóricamente infinita a la cual los circuitos pueden cambiar entre estados, o de hecho sus transistores.

- Automatización del paralelismo: Se dice en este paradigma que el éxito es inversamente proporcional al número de cores precisamente porque existen complejidades en el corazón del paralelismo que implican cosas que todavía no se pueden predecir ni con inteligencia artificial, se menciona cuales son las posibles estrategias para atacar un problema de forma paralela, esto da cuenta de que existe una forma prácticamente determinada de abordarlos pero no de automatizarlos, a pesar de que sí existan algunas partes que son automatizables en el proceso.

- Solución en el Hardware: Un diseño adecuado del Hardware permitiría que la paralelización siempre estuviera presente con respecto a los procesadores que se están usando de tal modo que alguno los problemas que son inherentes al paradigma pudieran evitarse. Esto ha resultado imposible hasta la fecha, de hecho, solo diseñar solamente algo tan efectivo y tradicional como se ha hecho en programación secuencial es algo que no existe hasta ahora. Existen algunas aproximaciones como OpenMP y programación por hilos de las que hablaremos más adelante.

Ventajas

- Resuelve problemas que no se podrían realizar en una sola CPU

- Resuelve problemas que no se pueden resolver en un tiempo razonable

- Permite ejecutar problemas de un orden y complejidad mayor
- Permite ejecutar código de manera más rápida (aceleración)
- Permite ejecutar en general más problemas
- Obtención de resultados en menos tiempo
- Permite la ejecución de varias instrucciones en simultáneo
- Permite dividir una tarea en partes independientes

Desventajas

- Mayor consumo de energía
- Mayor dificultad a la hora de escribir programas
- Dificultad para lograr una buena sincronización y comunicación entre las tareas
- Retardos ocasionados por comunicación entre tareas
- Número de componentes usados es directamente proporcional a los fallos potenciales
- Condiciones de carrera
  - Múltiples procesos se encuentran en condición de carrera si el resultado de los mismos depende del orden de su llegada
  - Si los procesos que están en condición de carrera no son correctamente sincronizados, puede producirse una corrupción de datos

Paralelismo de Grano Fino, Grano Grueso y Paralelismo Vergonzoso las aplicaciones a menudo se clasifican según la frecuencia con que sus subtareas se sincronizan o comunican entre sí. Una aplicación muestra un paralelismo de grano fino si sus subtareas deben comunicarse muchas veces por segundo, se considera paralelismo de grano grueso si no se comunican muchas veces por segundo, y es vergonzosamente paralelo si nunca o casi nunca se tienen que comunicar. Aplicaciones vergonzosamente paralelas son consideradas las más fáciles de paralelizar.

El tipo de problemas complejos que aborda el cómputo de alto rendimiento (HPC por High-Performance Computing) no se pueden resolver con una computadora de escritorio y tienen una escala determinada: toma mucho tiempo resolverlos, se necesita una gran cantidad de memoria (RAM), se tienen que realizar muchísimos experimentos parecidos y hay restricciones concretas de tiempo para encontrar resultados. La mayoría de estas temáticas están relacionadas con la innovación en la industria y su aplicación en áreas como energía, medio ambiente, biotecnología, medicina, ingeniería, etc.

A su vez, la evolución del procesador la cantidad de transistores sigue creciendo pero la velocidad de los núcleos individuales (core) casi no aumenta. Podríamos comprar un procesador de 128 núcleos pero el desafío de aprovechar la potencia de un procesador es cómo distribuimos el tiempo de ejecución de una misma tarea computacional.

De este modo, cada tarea computacional se divide en dos partes: serial y paralelizable. Como todas las aplicaciones tienen una parte secuencial, el tiempo de cómputo de la aplicación paralela está acotado por esa sección serial (Ley de Ahmdahl). Esto implica dos cuestiones fundamentales:

1. Es necesario reducir lo más posible el tiempo de ejecución serial

2. En una tarea computacional existe un máximo de procesadores que conviene poner a trabajar en forma paralela, más allá de ese punto no se justifica seguir paralelizando la tarea.

¿Qué debe tener en cuenta un centro de HPC para aprovechar al máximo los recursos de cómputo? Cada centro de HPC tiene que configurar, optimizar aplicaciones y sistemas operativos. Allí el monitoreo es clave, con la escala de las infraestructuras también surgen aplicaciones con problemas de memoria (cuello de botella) y la necesidad de utilizar mejores redes (PARAMNet-3, Infiniband, GigE). Tener poder de cómputo no es barato y solamente con el Hardware no alcanza. El equipo de operaciones de HPC necesita un alto grado de conocimiento y de disciplina para encarar la tarea.

Con el uso del cómputo de alto desempeño es posible por ejemplo reducir el tiempo que nos lleva descubrir nuevos medicamentos de años a meses. Una computadora de alto rendimiento es capaz de resolver este y otros tipos de problemas científicos avanzados mediante simulaciones, modelos y análisis. Estos sistemas abren las puertas de la "Cuarta Revolución Industrial", ya que

ayudan a resolver muchas de las problemáticas más importantes del mundo. Los sistemas HPC ya se utilizan para las siguientes tareas:

- Descubrir nuevos componentes de drogas y probar los conocidos para combatir diferentes tipos de cáncer y otras enfermedades
- Simular dinámicas moleculares para crear nuevos materiales, como tejidos balísticos
- Pronosticar cambios climáticos considerables para mejorar la preparación de las comunidades afectadas

Las supercomputadoras representan lo último en sistemas HPC. La definición de supercomputadora depende de diferentes estándares que van cambiando a medida que las capacidades evolucionan. Un solo clúster de supercomputadoras puede incluir decenas de miles de procesadores, y los sistemas más caros y potentes del mundo pueden costar más de US$ 100 millones de dólares.

Casos de Uso Emergentes a medida que las tecnologías mejoraron, la computación de alto rendimiento comenzó a implementarse en una gama de capacidades más amplia. Ahora contamos con más potencia de procesamiento y memoria que nunca para solucionar problemas más complejos.

- Aprendizaje automático: como subconjunto de la inteligencia artificial (IA), el aprendizaje automático (AA) se refiere a un sistema que tiene la capacidad de aprender de forma activa por sí mismo, a diferencia de recibir, de forma pasiva, instrucciones para ejecutar. Los sistemas HPC pueden utilizarse en aplicaciones de AA altamente avanzadas donde se analizan grandes cantidades de datos, por ejemplo, investigación del cáncer para detectar el melanoma en imágenes.
- Análisis de grandes conjuntos de datos: se recurre a la comparación rápida y a la correlación de grandes conjuntos de datos para complementar investigaciones y resolver problemas académicos, científicos, financieros, comerciales, gubernamentales, de salud y de seguridad cibernética. Este trabajo requiere un rendimiento masivo y capacidades de cómputo de una potencia enorme. Con un estimado de 50 petabytes de datos al año generados por las misiones, la NASA se apoya en la

supercomputación para analizar observaciones y realizar simulaciones con grandes conjuntos de información.

- Modelado avanzado y simulación: al no tener que realizar un montaje físico en las primeras etapas del proceso, el modelado avanzado y la simulación permiten que las empresas ahorren tiempo, materiales y costos de contratación de personal para lanzar sus productos al mercado con mayor rapidez. El modelado y la simulación en HPC se aplican en el descubrimiento y la prueba de fármacos, diseños automotrices y aeroespaciales, pronóstico de sistemas climáticos o meteorológicos, y aplicaciones energéticas.

Cómo Funciona la Computación de Alto Rendimiento Existen dos métodos principales para procesar la información en HPC:

- Procesamiento en serie: es el que realizan las unidades de procesamiento central (CPU). Cada núcleo de CPU, por lo general, realiza solo una tarea a la vez. Las CPU son fundamentales para ejecutar diferentes funciones, como sistemas operativos y aplicaciones básicas (por ej., procesamiento de textos, productividad en la oficina).

- Procesamiento en paralelo: es el que se puede realizar mediante varias CPU o unidades de procesamiento de gráficos (GPU). Las GPU, diseñadas originalmente para gráficos independientes, son capaces de realizar diferentes operaciones aritméticas por medio de una matriz de datos (como píxeles de pantalla) de forma simultánea. La capacidad para trabajar en varios planos de datos al mismo tiempo hace que las GPU sean la elección natural para el procesamiento en paralelo en tareas de aplicaciones de aprendizaje automático (AA), como el reconocimiento de objetos en videos.

Para superar los límites de la supercomputación, se necesitan diferentes arquitecturas de sistemas. Para que el procesamiento en paralelo pueda llevarse a cabo, la mayoría de los sistemas HPC integran varios procesadores y módulos de memoria a través de interconexiones con un ancho de banda enorme. Ciertos sistemas HPC combinan varias CPU y GPU, lo que se conoce como computación heterogénea.

La potencia de procesamiento de las computadoras se mide en unidades llamadas "FLOPS"[173] (operaciones de punto flotante por segundo). A principios de 2019, la supercomputadora más potente que existe alcanzó los 143,5 peta-Flops (143×10^{15}). Este tipo de supercomputadora se llama equipo de petaescala y puede realizar más de mil billones de FLOPS. Por su parte, una computadora de escritorio para juegos de alta gama es más de un millón de veces más lenta y llega apenas a los 200 giga-FLOPS (1×10^{9}).

Gracias a los avances tanto en procesamiento como rendimiento, en el año 2020 fuimos testigos de un nuevo salto en la era de la supercomputación: la exaescala, que es casi 1000 veces más rápida que la petaescala. Esto significa que un sistema de exaescala realiza 10^{18} (o mil millones por mil millones) operaciones por segundo. Necesitaríamos 5 millones de computadoras de escritorio para alcanzar el nivel de rendimiento de procesamiento en supercomputación de 1 exa-Flops, suponiendo que cada computadora de escritorio es capaz de realizar 200 giga-FLOPS.

Arquitecturas de Software y Hardware En las dos siguientes secciones se explican en detalle las dos clasificaciones de computadoras más conocidas en la actualidad. La primera clasificación, es la clasificación clásica de Flynn en dónde se tienen en cuenta sistemas con uno o varios procesadores, la segunda clasificación es moderna en la que sólo se tienen en cuenta los sistemas con más de un procesador.

El objetivo es presentar de una forma clara los tipos de clasificación que existen en la actualidad desde el punto de vista de distintos autores, así como cuáles son las ventajas e inconvenientes que cada uno ostenta, ya que es común que al resolver un problema particular se usen una o más arquitecturas de Hardware interconectadas generalmente por red.

9.3 Clasificación Clásica de Flynn

La clasificación clásica de arquitecturas de computadoras que hace alusión a sistemas con uno o varios procesadores fue realizada por Michael J. Flynn y la publicó por primera vez en 1966 y por segunda vez en 1970.

[173] "FLOPS" describe una velocidad de procesamiento teórica: para hacer posible esa velocidad es necesario enviar datos a los procesadores de forma continua. Por lo tanto, el procesamiento de los datos se debe tener en cuenta en el diseño del sistema. La memoria del sistema, junto con las interconexiones que unen los nodos de procesamiento entre sí, impactan en la rapidez con la que los datos llegan a los procesadores.

Esta taxonomía se basa en el flujo que siguen los datos dentro de la máquina y de las instrucciones sobre esos datos. Se define como flujo de instrucciones al conjunto de instrucciones secuenciales que son ejecutadas por un único procesador y como flujo de datos al flujo secuencial de datos requeridos por el flujo de instrucciones. Con estas consideraciones, Flynn clasifica los sistemas en cuatro categorías:

Single Instruction stream, Single Data stream (SISD) Los sistemas Monoprocesador de este tipo se caracterizan por tener un único flujo de instrucciones sobre un único flujo de datos, es decir, se ejecuta una instrucción detrás de otra. Este es el concepto de arquitectura serie de Von Neumann donde, en cualquier momento, sólo se ejecuta una única instrucción, un ejemplo de estos sistemas son las máquinas secuenciales convencionales.

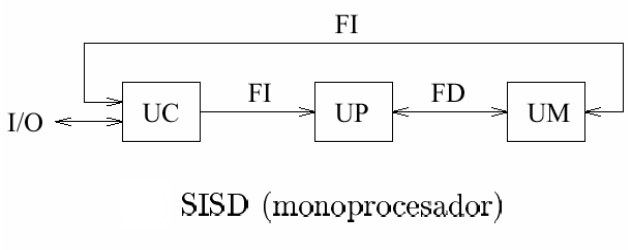

Figura 5: Ejemplo de máquina SISD

Single Instruction stream, Multiple Data stream (SIMD) Estos sistemas de procesador Matricial tienen un único flujo de instrucciones que operan sobre múltiples flujos de datos. Ejemplos de estos sistemas los tenemos en las máquinas vectoriales con Hardware escalar y vectorial.

El procesamiento es síncrono, la ejecución de las instrucciones sigue siendo secuencial como en el caso anterior, todos los elementos realizan una misma instrucción pero sobre una gran cantidad de datos. Por este motivo existirá concurrencia de operación, es decir, esta clasificación es el origen de la máquina paralela.

El funcionamiento de este tipo de sistemas es el siguiente. La unidad de control manda una misma instrucción a todas las unidades de proceso

(ALUs). Las unidades de proceso operan sobre datos diferentes pero con la misma instrucción recibida.

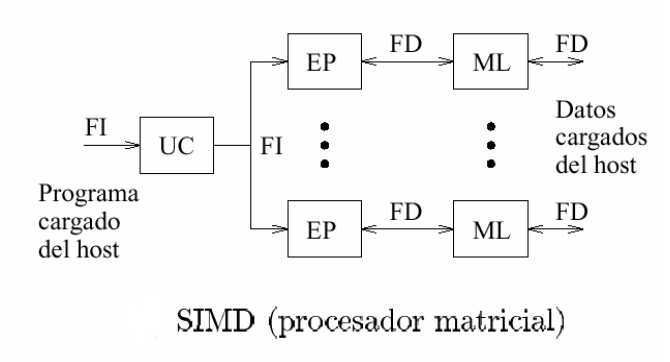

Figura 6: Ejemplo de máquina SIMD

Existen dos alternativas distintas que aparecen después de realizarse esta clasificación:

- Arquitectura Vectorial con segmentación, una CPU única particionada en unidades funcionales independientes trabajando sobre flujos de datos concretos.
- Arquitectura Matricial (matriz de procesadores), varias ALUs idénticas a las que el procesador da instrucciones, asigna una única instrucción pero trabajando sobre diferentes partes del programa.

Multiple Instruction stream, Single Data stream (MISD) Sistemas con múltiples instrucciones Array Sistólico que operan sobre un único flujo de datos. Este tipo de sistemas no ha tenido implementación hasta hace poco tiempo. Los sistemas MISD se contemplan de dos maneras distintas:

- Varias instrucciones operando simultáneamente sobre un único dato.
- Varias instrucciones operando sobre un dato que se va convirtiendo en un resultado que será la entrada para la siguiente etapa. Se trabaja de forma segmentada, todas las unidades de proceso pueden trabajar de forma concurrente.

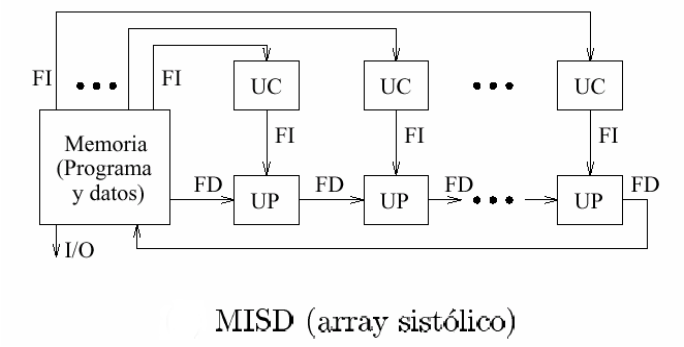

Figura 7: Ejemplo de máquina MISD

Multiple Instruction stream, Multiple Data stream (MIMD) Sistemas con un flujo de múltiples instrucciones Multiprocesador que operan sobre múltiples datos. Estos sistemas empezaron a utilizarse antes de la década de los 80s. Son sistemas con memoria compartida que permiten ejecutar varios procesos simultáneamente (sistema multiprocesador).

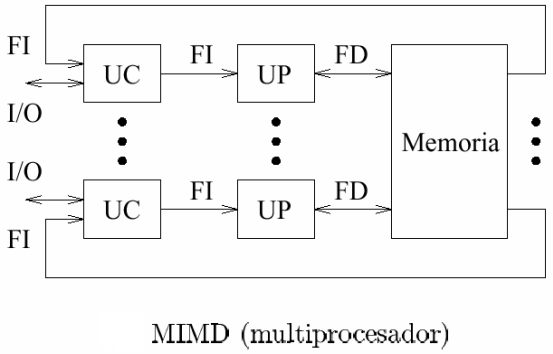

Figura 8: Ejemplo de máquina MIMD

Cuando las unidades de proceso reciben datos de una memoria no compartida estos sistemas reciben el nombre de MULTIPLE SISD (MSISD). En arquitecturas con varias unidades de control (MISD Y MIMD), existe otro nivel superior con una unidad de control que se encarga de controlar todas las unidades de control del sistema -ejemplo de estos sistemas son las máquinas paralelas actuales-.

9.4 Categorías de Computadoras Paralelas

Clasificación moderna que hace alusión única y exclusivamente a los sistemas que tienen más de un procesador (i.e máquinas paralelas). Existen dos tipos de sistemas teniendo en cuenta su acoplamiento:

- Los sistemas fuertemente acoplados son aquellos en los que los procesadores dependen unos de otros.
- Los sistemas débilmente acoplados son aquellos en los que existe poca interacción entre los diferentes procesadores que forman el sistema.

Atendiendo a esta y a otras características, la clasificación moderna divide a los sistemas en dos tipos: Sistemas multiprocesador (fuertemente acoplados) y sistemas multicomputadoras (débilmente acoplados).

9.4.1 Equipo Paralelo de Memoria Compartida

Un multiprocesador puede verse como una computadora paralela compuesta por varios procesadores interconectados que comparten un mismo sistema de memoria.

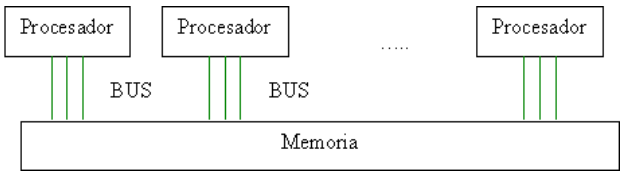

Figura 9: Arquitectura de una computadora paralela con memoria compartida

Los sistemas multiprocesadores son arquitecturas MIMD con memoria compartida. Tienen un único espacio de direcciones para todos los procesadores y los mecanismos de comunicación se basan en el paso de mensajes desde el punto de vista del programador.

Dado que los multiprocesadores comparten diferentes módulos de memoria, pueden acceder a un mismo módulo varios procesadores, a los multiprocesadores también se les llama sistemas de memoria compartida.

Para hacer uso de la memoria compartida por más de un procesador, se requiere hacer uso de técnicas de semáforos que mantienen la integridad

de la memoria; esta arquitectura no puede crecer mucho en el número de procesadores interconectados por la saturación rápida del bus o del medio de interconexión.

Dependiendo de la forma en que los procesadores comparten la memoria, se clasifican en sistemas multiprocesador UMA, NUMA, COMA y Pipeline, que explicamos a continuación:

Uniform Memory Access (UMA) Sistema multiprocesador con acceso uniforme a memoria. La memoria física es uniformemente compartida por todos los procesadores, esto quiere decir que todos los procesadores tienen el mismo tiempo de acceso a todas las palabras de la memoria. Cada procesador tiene su propia Caché privada y también se comparten los periféricos.

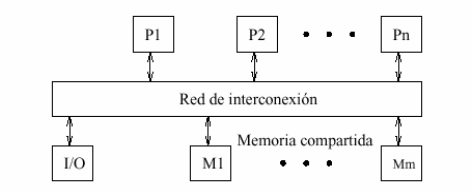

Figura 10: Acceso Uniforme a la memoria UMA

Los multiprocesadores son sistemas fuertemente acoplados (tightly-coupled), dado el alto grado de compartición de los recursos (Hardware o Software) y el alto nivel de interacción entre procesadores, lo que hace que un procesador depende de lo que hace otro. El sistema de interconexión debe ser rápido y puede ser de uno de los siguientes tipos: bus común, red Crossbar[174] y red Multietapa. Este modelo es conveniente para aplicaciones de propósito general y de tiempo compartido por varios usuarios, existen dos categorías de sistemas UMA.

- Sistema Simétrico: cuando todos los procesadores tienen el mismo tiempo de acceso a todos los componentes del sistema (incluidos los periféricos), reciben el nombre de sistemas multiprocesador simétrico. Los procesadores tienen el mismo dominio (prioridad) sobre los periféricos y cada procesador tiene la misma capacidad para procesar.

[174] Red reconfigurable que permite la conexión de cada entrada con cualquiera de las salidas, es decir, permite cualquier permutación.

- Sistema Asimétrico: son sistemas con procesador principal y procesadores subordinados, en donde sólo el primero puede ejecutar aplicaciones y dónde el tiempo de acceso para diferentes procesadores no es el mismo. Los procesadores subordinados (Attached) ejecutan código usuario bajo la supervisión del principal, por lo tanto cuando una aplicación es ejecutada en un procesador principal dispondrá de una cierta prioridad.

Non Uniform Memory Access (NUMA) Un sistema multiprocesador NUMA es un sistema de memoria compartida donde el tiempo de acceso varía según donde se encuentre localizado el acceso.

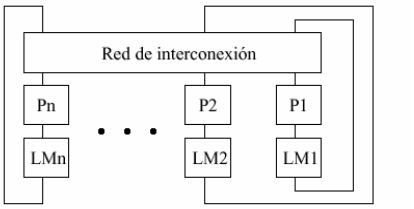

Figura 11: Acceso no uniforme a la memoria NUMA

El acceso a memoria, por tanto, no es uniforme para diferentes procesadores, existen memorias locales asociadas a cada procesador y estos pueden acceder a datos de su memoria local de una manera más rápida que a las memorias de otros procesadores, debido a que primero debe aceptarse dicho acceso por el procesador del que depende el módulo de memoria local.

Todas las memorias locales conforman la memoria global compartida y físicamente distribuida y accesible por todos los procesadores.

Cache Only Memory Access (COMA) Los sistemas COMA son un caso especial de los sistemas NUMA. Este tipo de sistemas no ha tenido mucha trascendencia, al igual que los sistemas SIMD.

Las memorias distribuidas son memorias Cachés, por este motivo es un sistema muy restringido en cuanto a la capacidad de memoria global. No hay jerarquía de memoria en cada módulo procesador. Todas las Cachés forman un mismo espacio global de direcciones. El acceso a las Cachés remotas se realiza a través de los directorios distribuidos de las Cachés.

Dependiendo de la red de interconexión utilizada, se pueden utilizar jerarquías en los directorios para ayudar a la localización de copias de bloques de Caché.

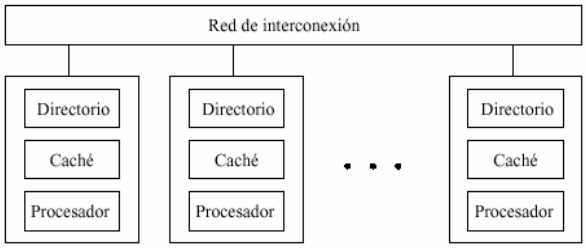

Figura 12: Ejemplo de COMA

Procesador Vectorial Pipeline En la actualidad es común encontrar en un solo procesador los denominados Pipeline o Procesador Vectorial Pipeline del tipo MISD. En estos procesadores los vectores fluyen a través de las unidades aritméticas Pipeline.

Las unidades constan de una cascada de etapas de procesamiento compuestas de circuitos que efectúan operaciones aritméticas o lógicas sobre el flujo de datos que pasan a través de ellas, las etapas están separadas por registros de alta velocidad usados para guardar resultados intermedios. Así la información que fluye entre las etapas adyacentes está bajo el control de un reloj que se aplica a todos los registros simultáneamente.

9.4.2 Equipo Paralelo de Memoria Distribuida

Los sistemas multicomputadoras (los más comunes son los Clusters) se pueden ver como una computadora paralela en el cual cada procesador tiene su propia memoria local. En estos sistemas la memoria se encuentra distribuida y no compartida como en los sistemas multiprocesador. Los procesadores se comunican a través de paso de mensajes, ya que éstos sólo tienen acceso directo a su memoria local y no a las memorias del resto de los procesadores.

La transferencia de los datos se realiza a través de la red de interconexión que conecta un subconjunto de procesadores con otro subconjunto. La trans-

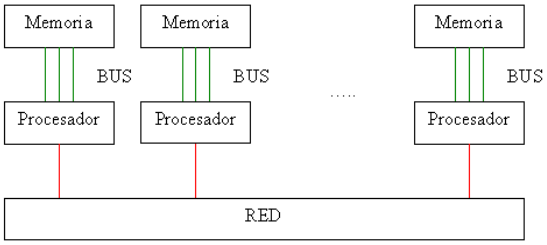

Figura 13: Arquitectura de una computadora paralela con memoria distribuida

ferencia de unos procesadores a otros se realiza por múltiples transferencias entre procesadores conectados dependiendo del establecimiento de dicha red.

Dado que la memoria está distribuida entre los diferentes elementos de proceso, estos sistemas reciben el nombre de distribuidos. Por otra parte, estos sistemas son débilmente acoplados, ya que los módulos funcionan de forma casi independiente unos de otros. Este tipo de memoria distribuida es de acceso lento por ser peticiones a través de la red, pero es una forma muy efectiva de tener acceso a un gran volumen de memoria.

9.4.3 Equipo Paralelo de Memoria Compartida-Distribuida

La tendencia actual en las máquinas paralelas es de aprovechar las facilidades de programación que ofrecen los ambientes de memoria compartida y la escalabilidad de los ambientes de memoria distribuida. En este modelo se conectan entre sí módulos de multiprocesadores, pero se mantiene la visión global de la memoria a pesar de que es distribuida.

Clusters El desarrollo de sistemas operativos y compiladores del dominio público (Linux y Software GNU), estándares para interfaz de paso de mensajes (Message Passing Interface MPI), conexión universal a periféricos (Peripheral Component Interconnect PCI), entre otros, han hecho posible tomar ventaja de los recursos económicos computacionales de producción masiva (procesadores, discos, redes).

La principal desventaja que presenta a los proveedores de multicomputadoras es que deben satisfacer una amplia gama de usuarios, es decir, deben ser generales. Esto aumenta los costos de diseños y producción de equipos, así

como los costos de desarrollo de Software que va con ellos: sistema operativo, compiladores y aplicaciones. Todos estos costos deben ser añadidos cuando se hace una venta. Por supuesto alguien que sólo necesita procesadores y un mecanismo de pase de mensajes no debería pagar por todos estos añadidos que nunca usará. Estos usuarios son los que están impulsando el uso de Clusters principalmente de computadoras personales (PC), cuya arquitectura se muestra a continuación:

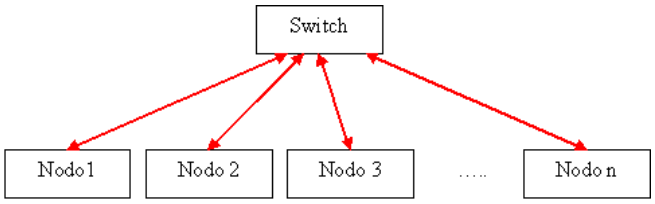

Figura 14: Arquitectura de un cluster

Los Cluster[175] se pueden clasificar en dos tipos según sus características físicas:

- Cluster homogéneo: si todos los procesadores y/o nodos participantes en el equipo paralelo son iguales en capacidad de cómputo -es permitido variar la cantidad de memoria o disco duro en cada procesador-
- Cluster heterogéneo: es aquel en que al menos uno de los procesadores y/o nodos participantes en el equipo paralelo son de distinta capacidad de cómputo

El Clustering no presenta dependencias a nivel de Hardware (no todos los equipos necesitan el mismo Hardware) ni a nivel de Software (no necesitan el mismo sistema operativo). Este tipo de sistemas disponen de una interfaz que permite dirigir el comportamiento de los Clusters. Dicha interfaz es la encargada de la interacción con usuarios y procesos, realizando la división de la carga entre los diversos servidores que compongan el Cluster.

[175] Existe el Ranking de las 500 supercomputadoras más poderosas del mundo (esta se actualiza cada seis meses en junio y noviembre) y puede ser consultada en:
https://top500.org

Los Clusters pueden formarse de diversos equipos; los más comunes son los de computadoras personales, pero es creciente el uso de computadoras multiprocesador de más de un procesador de memoria compartida interconectados por red con los demás nodos del mismo tipo, incluso el uso de computadoras multiprocesador de procesadores vectoriales Pipeline. Los Clusters armados con la configuración anterior tienen grandes ventajas para procesamiento paralelo:

- La reciente explosión en redes implica que la mayoría de los componentes necesarios para construir un Cluster son vendidos en altos volúmenes y por lo tanto son económicos. Ahorros adicionales se pueden obtener debido a que sólo se necesitará una tarjeta de vídeo, un monitor y un teclado por Cluster. El mercado de los multiprocesadores es más reducido y más costoso
- El reemplazar un componente defectuoso en un Cluster es relativamente trivial comparado con hacerlo en un multiprocesador, permitiendo una mayor disponibilidad de Clusters cuidadosamente diseñados

Desventajas del uso de Clusters de computadoras personales para procesamiento paralelo:

- Con raras excepciones, los equipos de redes generales producidos masivamente no están diseñados para procesamiento paralelo y típicamente su latencia es alta y los anchos de banda pequeños comparados con multiprocesadores. Dado que los Clusters explotan tecnología que sea económica, los enlaces en el sistema no son veloces implicando que la comunicación entre componentes debe pasar por un proceso de protocolos de negociación lentos, incrementando seriamente la latencia. En muchos y en el mejor de los casos (debido a costos) se recurre a una red tipo Fast Ethernet restringimiento la escalabilidad del Cluster
- Hay poco soporte de Software para manejar un Cluster como un sistema integrado
- Los procesadores no son tan eficientes como los procesadores usados en los multiprocesadores para manejar múltiples usuarios y/o procesos. Esto hace que el rendimiento de los Clusters se degrade con relativamente pocos usuarios y/o procesos

- Muchas aplicaciones importantes disponibles en multiprocesadores y optimizadas para ciertas arquitecturas, no lo están en Clusters

Sin lugar a duda los Clusters presentan una alternativa importante para varios problemas particulares, no sólo por su economía, si no también porque pueden ser diseñados y ajustados para ciertas aplicaciones. Las aplicaciones que pueden sacar provecho de Clusters son en donde el grado de comunicación entre procesos es de bajo a medio.

Tipos de Cluster Básicamente existen tres tipo de Clusters, cada uno de ellos ofrece ventajas y desventajas, el tipo más adecuado para el cómputo científico es del de alto-rendimiento, pero existen aplicaciones científicas que pueden usar más de un tipo al mismo tiempo.

- Alta disponibilidad (Fail-over o High-Availability): este tipo de Cluster está diseñado para mantener uno o varios servicios disponibles incluso a costa de rendimiento, ya que su función principal es que el servicio jamás tenga interrupciones como por ejemplo un servicio de bases de datos de transacciones bancarias
- Alto rendimiento (HPC o High Performance Computing): este tipo de Cluster esta diseñado para obtener el máximo rendimiento de la aplicación utilizada incluso a costa de la disponibilidad del sistema, es decir el Cluster puede sufrir caídas, este tipo de configuración está orientada a procesos que requieran mucha capacidad de cálculo.
- Balanceo de Carga (Load-Balancing): este tipo de Cluster está diseñado para balancear la carga de trabajo entre varios servidores, lo que permite tener, por ejemplo, un servicio de cálculo intensivo multiusuarios que detecte tiempos muertos del proceso de un usuario para ejecutar en dichos tiempos procesos de otros usuarios.

Grids Son cúmulos (grupo de Clusters) de arquitecturas en paralelo interconectados por red, los cuales distribuyen tareas entre los Clusters que lo forman, estos pueden ser homogéneos o heterogéneos en cuanto a los nodos componentes del cúmulo. Este tipo de arquitecturas trata de distribuir

cargas de trabajo acorde a las características internas de cada Cluster y las necesidades propias de cada problema, esto se hace a dos niveles, una en la parte de programación en conjunto con el balance de cargas y otra en la parte de Hardware que tiene que ver con las características de cada arquitectura que conforman el cúmulo.

Balance de Carga A la hora de diseñar un sistema paralelo con compartición de recursos, es necesario considerar cómo balancear la carga de trabajo. Se entiende este concepto, como la técnica usada para dividir el trabajo a compartir entre varios procesos, equipos de cómputo, u otros recursos. Está muy relacionada con los sistemas multiproceso, que trabajan o pueden trabajar con más de una unidad para llevar a cabo su funcionalidad.

Para evitar los cuellos de botella, el balance de la carga de trabajo se reparte de forma equitativa a través de un algoritmo que estudia las peticiones del sistema y las redirecciona a la mejor opción.

Balance de Carga por Hardware presenta las siguientes características:

- A partir de un algoritmo de planificación de procesos -Round Robin, LRU-, examina las peticiones entrantes y selecciona el más apropiado entre los distintos elementos del sistema
- La selección del siguiente libre del sistema está basada en el algoritmo de sustitución y es aleatoria
- La sesión debe de ser mantenida por el desarrollador
- Al ser un proceso de Hardware, es muy rápido

Balance de Carga por Software presenta las siguientes características:

- Examinan la solicitud para garantizar que se puede satisfacer la demanda de los usuarios
- Distintas peticiones del mismo usuario son servidas simultáneamente o secuencialmente según se definan

- Más lentos que los balanceadores de Hardware
- Normalmente son soluciones baratas

Escalabilidad Se entiende por escalabilidad a la capacidad de adaptación y respuesta de un sistema con respecto al rendimiento del mismo a medida que aumentan de forma significativa la carga computacional del mismo. Aunque parezca un concepto claro, la escalabilidad de un sistema es un aspecto complejo e importante del diseño de sistemas paralelos.

La escalabilidad está íntimamente ligada al diseño de sistemas paralelos, influye en el rendimiento de forma significativa. Si una aplicación está bien diseñada, la escalabilidad no constituye un problema. Analizando la escalabilidad, se deduce de la implementación y del diseño general del sistema. No es atributo del sistema configurable.

La escalabilidad supone un factor crítico en el crecimiento de un sistema paralelo. Si un sistema tiene como objetivo crecer la carga computacional -en el número de usuarios o procesos- manteniendo su rendimiento actual, tiene que evaluar dos posibles opciones:

- Con un Hardware de mayor potencia o
- Con una mejor combinación de Hardware y Software

Se pueden distinguir dos tipos de escalabilidad, vertical y horizontal:

- El escalar verticalmente o escalar hacia arriba, significa el añadir más recursos a un solo nodo en particular dentro de un sistema, tal como el añadir memoria o un disco duro más rápido a una computadora.
- La escalabilidad horizontal, significa agregar más nodos a un sistema, tal como añadir una computadora nueva a un programa de aplicación para espejo.

Escalabilidad Vertical El escalar hacia arriba de un sistema viene a significar una migración de todo el sistema a un nuevo Hardware que es más potente y eficaz que el actual. Una vez se ha configurado el sistema futuro, se realizan una serie de validaciones y copias de seguridad y se pone en funcionamiento. Las aplicaciones que estén funcionando bajo la arquitectura

Hardware antigua no sufren con la migración, el impacto en el código es mínimo.

Este modelo de escalabilidad vertical tiene un aspecto negativo. Al aumentar la potencia en base a ampliaciones de Hardware, llegará un momento que existirá algún tipo de limitación de Hardware. Además a medida que se invierte en Hardware de muy altas prestaciones, los costos se disparan tanto de forma temporal -ya que si se ha llegado al umbral máximo, hay componentes de Hardware que tardan mucho tiempo en ampliar su potencia de forma significativa- como económicos. Sin embargo a nivel estructural no supone ninguna modificación reseñable, lo que la convierte en una buena opción si los costos anteriores son asumibles.

Escalabilidad Horizontal La escalabilidad horizontal consiste en potenciar el rendimiento del sistema paralelo desde un aspecto de mejora global, a diferencia de aumentar la potencia de una única parte del mismo. Este tipo de escalabilidad se basa en la modularidad de su funcionalidad, por ello suele estar conformado por una agrupación de equipos que dan soporte a la funcionalidad completa. Normalmente, en una escalabilidad horizontal se añaden equipos para dar más potencia a la red de trabajo.

Con un entorno de este tipo, es lógico pensar que la potencia de procesamiento es directamente proporcional al número de equipos en la red. El total de la potencia de procesamiento es la suma de la velocidad física de cada equipo transferida por la partición de aplicaciones y datos extendida a través de los nodos.

Si se aplica un modelo de escalabilidad basado en la horizontalidad, no existen limitaciones de crecimiento a priori. Como principal defecto, este modelo de escalabilidad supone una gran modificación en el diseño, lo que conlleva a una gran trabajo de diseño y reimplantación. Si la lógica se ha concebido para un único servidor, es probable que se tenga que estructurar el modelo arquitectónico para soportar este modelo de escalabilidad.

El encargado de cómo realizar el modelo de partición de datos en los diferentes equipos es el desarrollador. Existen dependencias en el acceso a la aplicación. Es conveniente, realizar una análisis de actividad de los usuarios para ir ajustando el funcionamiento del sistema. Con este modelo de escalabilidad, se dispone de un sistema al que se pueden agregar recursos de manera casi infinita y adaptable al crecimiento de cargas de trabajo y nuevos usuarios.

La escalabilidad cuenta como factor crítico en el crecimiento de usuarios. Es mucho más sencillo diseñar un sistema con un número constante de usuarios -por muy alto que sea este- que diseñar un sistema con un número creciente y variable de usuarios. El crecimiento relativo de los números es mucho más importante que los números absolutos.

9.4.4 Cómputo Paralelo en Multihilos

En una computadora, sea secuencial o paralela, para aprovechar las capacidades crecientes del procesador, el sistema operativo divide su tiempo de procesamiento entre los distintos procesos, de forma tal que para poder ejecutar a un proceso, el Kernel les asigna a cada uno una prioridad y con ello una fracción del tiempo total de procesamiento, de forma tal que se pueda atender a todos y cada uno de los procesos de manera eficiente.

En particular, en la programación en paralelo usando MPI, cada proceso -que eventualmente puede estar en distinto procesador- se lanza como una copia del programa con datos privados y un identificador del proceso único, de tal forma que cada proceso sólo puede compartir datos con otro proceso mediante paso de mensajes.

Esta forma de lanzar procesos por cada tarea que se desee hacer en paralelo es costosa, por llevar cada una de ellas toda una gama de subprocesos para poderle asignar recursos por parte del sistema operativo. Una forma más eficiente de hacerlo es que un proceso pueda generar bloques de subprocesos que puedan ser ejecutados como parte del proceso (como subtareas), así en el tiempo asignado se pueden atender a más de un subproceso de manera más eficiente, esto es conocido como programación multihilos.

Los hilos realizarán las distintas tareas necesarias en un proceso. Para hacer que los procesos funcionen de esta manera, se utilizan distintas técnicas que le indican al Kernel cuales son las partes del proceso que pueden ejecutarse simultáneamente y el procesador asignará una fracción de tiempo exclusivo al hilo del tiempo total asignado al proceso.

Los datos pertenecientes al proceso pasan a ser compartidos por los subprocesos lanzados en cada hilo y mediante una técnica de semáforos el Kernel mantiene la integridad de estos. Esta técnica de programación puede ser muy eficiente si no se abusa de este recurso, permitiendo un nivel más de paralelización en cada procesador. Esta forma de paralelización no es exclusiva de equipos multiprocesadores o multicomputadoras, ya que pueden ser implementados a nivel de sistema operativo.

9.4.5 Cómputo Paralelo en CUDA

Son las siglas de arquitectura unificada de dispositivos de cómputo (Compute Unified Device Architecture CUDA) que hace referencia a una plataforma de computación en paralelo (que incluye su propia RAM conocida como GRAM para hacer óptima la solicitud de dicho recurso por sus múltiples cores) incluyendo un compilador y un conjunto de herramientas de desarrollo que permiten a los programadores usar una variación del lenguaje de programación C -Por medio de Wrappers se puede usar MatLab, Python, Julia, Fortran y Java en vez de C/C++- para codificar algoritmos en las unidades de procesamiento de gráficos (Graphics Processing Unit GPU).

CUDA intenta explotar las ventajas de las GPU (de decenas a centenas de cores[176]) frente a las CPU (decenas de cores[177]) de propósito general utilizando el paralelismo que ofrecen sus múltiples cores o núcleos, que permiten el lanzamiento de un altísimo número de hilos simultáneos. Si una aplicación está bien diseñada utilizando numerosos hilos que realizan tareas independientes usando la memoria compartida de la tarjeta (que es lo que hacen las GPU al procesar gráficos, su tarea natural) hará un uso eficiente de este dispositivo.

Por ejemplo, la tarjeta gráfica de NVidia GEFORCE RTX 3090 proporciona 10,496 cores y 24 GB de GDDR6x, mientras que la tarjeta NVidia Titan RTX proporciona 130 Tensor[178] TFLOP de rendimiento, 576 núcleos tensores y 24 GB de memoria GDDR6.

Ahora, miles de desarrolladores, científicos e investigadores están encontrando innumerables aplicaciones prácticas para esta tecnología en campos como el procesamiento de vídeo e imágenes, la biología y la química computa-

[176] La GRAM de las tarjetas CUDA es relativamente pequeña (a lo más de algunas decenas de Gigabytes), pero está diseñada para que pueda ser compartida por sus cores de forma óptima. Pero si es necesario que múltiples cores de la GPU soliciten datos a la RAM del equipo de cómputo, pueden ocasionar degradación en la eficiencia de la GPU porque la RAM tiene una arquitectura DDR4 o DDR5 (lo que significa que solo puede atender peticiones de datos de sólo 4 o 5 cores simultáneamente).

[177] Los actuales CPUs pueden tener decenas de cores y comparten hasta Terabyates de RAM. Pero la arquitectura actual de la RAM es de tipo DDR4 o DDR5, esto significa, que sólo hay 4 o 5 canales para satisfacer las solicitudes de petición de datos de/hacia los cores, si más cores necesita acceso a la RAM tendrán que esperar que los otros cores concluyan su uso, mermando la eficiencia de la CPU.

[178] Un Tensor core (o núcleos Tensor) calculan la operación de una matriz 4x4 completa, la cual se calcula por reloj. Estos núcleos pueden multiplicar dos matrices FP16 4x4 y sumar la matriz FP32 al acumulador.

cional, la simulación de la dinámica de fluidos, la reconstrucción de imágenes de Tomografía Axial Computarizada TAC, el análisis sísmico o el trazado de rayos, entre otros.

Procesamiento paralelo con CUDA Los sistemas informáticos están pasando de realizar el «procesamiento central» en la CPU a realizar «coprocesamiento» repartido entre la CPU y la GPU. Para posibilitar este nuevo paradigma computacional, NVIDIA ha inventado la arquitectura de cálculo paralelo CUDA, que ahora se incluye en las GPUs GeForce, ION Quadro y Tesla GPUs, lo cual representa una base instalada considerable para los desarrolladores de aplicaciones.

CUDA ha sido recibida con entusiasmo por la comunidad científica. Por ejemplo, se está utilizando para acelerar AMBER, un simulador de dinámica molecular empleado por más de 60,000 investigadores del ámbito académico y farmacéutico de todo el mundo para acelerar el descubrimiento de nuevos medicamentos. En el mercado financiero, Numerix y CompatibL introdujeron soporte de CUDA para una nueva aplicación de cálculo de riesgo de contraparte y, como resultado, se ha multiplicado por 18 la velocidad de la aplicación. Cerca de 400 instituciones financieras utilizan Numerix en la actualidad.

Un buen indicador de la excelente acogida de CUDA es la rápida adopción de la GPU Tesla para aplicaciones de GPU Computing. En la actualidad hay más de 700 Clusters de GPUs instalados en compañías publicadas en Fortune 500 de todo el mundo, lo que incluye empresas como Schlumberger y Chevron en el sector energético o BNP Pariba en el sector bancario. Y en el 2021 se puso en funcionamiento el cluster Perlmutter que esta formado por 6,159 NVIDIA A100 Tensor Core GPUs para ciencias astrofísicas y atmosféricas.

Por otra parte, la reciente llegada de los últimos sistemas operativos de Microsoft y Apple está convirtiendo el GPU Computing en una tecnología de uso masivo. En estos nuevos sistemas, la GPU no actúa únicamente como procesador gráfico, sino como procesador paralelo de propósito general accesible para cualquier aplicación.

Plataforma de Cálculo Paralelo CUDA proporciona unas cuantas extensiones de MatLab, Fortran, Python, Julia, C, C++, etc. Que permiten implementar el paralelismo en el procesamiento de tareas y datos con diferentes niveles de granularidad. El programador puede expresar ese paralelismo

mediante diferentes lenguajes de alto nivel como Fortran, Python, Julia, C y C++ o mediante estándares abiertos como las directivas de OpenACC -que es un estándar de programación para el cómputo en paralelo desarrollado por Cray, CAPS, Nvidia y PGI diseñado para simplificar la programación paralela de sistemas heterogéneos de CPU/GPU-. En la actualidad, la plataforma CUDA se utiliza en miles de aplicaciones aceleradas en la GPU y en miles de artículos de investigación publicados, en las áreas de:

- Bioinformática
- Cálculo financiero
- Dinámica de fluidos computacional (CFD)
- Ciencia de los datos, analítica y bases de datos
- Defensa e Inteligencia
- Procesamiento de imágenes y visión computarizadas
- EDA (diseño automatizado)
- Aprendizaje automático, Inteligencia artificial
- Ciencia de los materiales
- Medios audiovisuales y entretenimiento
- Imágenes médicas
- Dinámica molecular
- Análisis numérico
- Química cuántica
- Exploración sísmica
- Mecánica estructural computacional
- Visualización e interacción de proteínas
- Modelos meteorológicos y climáticos

Algunas bibliotecas aceleradas en la GPU:

- Thrust C++ Template
- cuBLAS
- cuSPARSE
- NPP
- cuFFT

Algunos lenguajes de programación:

- CUDA C/C++, Fortran, Python, Julia
- .NET, Java

Algunos compiladores disponibles:

- OpenACC, paraleliza automáticamente los bucles de código Fortran o C utilizando directivas
- Compilador de autoparalelización de C y Fortran (de PGI) para CUDA C
- Compilador de autoparalelización HMPP de CAPS para CUDA C basado en C y Fortran
- Fortran, Python, Julia, Java, C++ y C
- Compilador de Fortran para CUDA de PGI
- Traductor de Fortran a C para CUDA
- FLAGON: librería de Fortran 95 para cálculo en la GPU
- Interfaz (Wrapper) de Python para CUDA: PyCUDA
- Wrapper de Java
- jCUDA: Java para CUDA
- Vínculos para las librerías BLAS y FFT de CUDA

- JaCUDA
- Integración de .NET para CUDA
- Thrust: librería de plantillas de C++ para CUDA
- CuPP : infraestructura de C++ para CUDA
- Libra: capa de abstracción de C/C++ para CUDA
- F# para CUDA
- Librería ArrayFire para acelerar el desarrollo de aplicaciones de GPU Computing en C, C++, Fortran y Python

Soporte de MATLAB, Mathematica, R, LabView:

- MATLAB, Mathematica, R, LabView
- MathWorks: Librerías MATLAB procesadas con GPUs NVIDIA
- Plugin Jacket basado en CUDA para MATLAB
- GPULib: librería de funciones matemáticas con vínculos para IDL y MATLAB
- Programación con CUDA en Mathematica de Wolfram
- Plugin de Mathematica para CUDA
- Habilitación del GPU Computing en el entorno estadístico de R
- Librería CUDA para LabVIEW de National Instruments
- Formas de usar CUDA desde LabVIEW CVI

Además existen herramientas de productividad y Cluster:

- Soporte de Eclipse para CUDA
- CUDA Occupancy Calculator
- Administrador de Clusters de cálculo para GPUs Tesla
- PBS Works para GPUs Tesla
- Scyld de Penguin Computing

9.5 Métricas de Desempeño

Las métricas de desempeño del procesamiento de alguna tarea en paralelo es un factor importante para medir la eficiencia y consumo de recursos al resolver una tarea con un número determinado de procesadores y recursos relacionados de la interconexión de éstos.

Entre las métricas para medir desempeño en las cuales como premisa se mantiene fijo el tamaño del problema, destacan las siguientes:

- Factor de aceleración
- Eficiencia
- Fracción serial

Cada una de ellas mide algo en particular y sólo la combinación de estas dan un panorama general del desempeño del procesamiento en paralelo de un problema en particular en una arquitectura determinada al ser comparada con otras.

Factor de Aceleración (o Speed-Up) Se define como el cociente del tiempo que se tarda en completar el cómputo de la tarea usando un sólo procesador entre el tiempo que necesita para realizarlo en p procesadores trabajando en paralelo:

$$s = \frac{T(1)}{T(p)} \tag{9.1}$$

se asume que se usará el mejor algoritmo tanto para un solo procesador como para p procesadores.

Esta métrica en el caso ideal debería de aumentar de forma lineal al aumento del número de procesadores.

Eficiencia Se define como el cociente del tiempo que se tarda en completar el cómputo de la tarea usando un solo procesador entre el número de procesadores multiplicado por el tiempo que necesita para realizarlo en p procesadores trabajando en paralelo:

$$e = \frac{T(1)}{pT(p)} = \frac{s}{p}. \tag{9.2}$$

Este valor será cercano a la unidad cuando el Hardware se esté usando de manera eficiente, en caso contrario el Hardware será desaprovechado.

Fracción serial Se define como el cociente del tiempo que se tarda en completar el cómputo de la parte secuencial de una tarea entre el tiempo que se tarda en completar el cómputo de la tarea usando un solo procesador:

$$f = \frac{T_s}{T(1)} \tag{9.3}$$

pero usando la ley de Amdahl:

$$T(p) = T_s + \frac{T_p}{p}$$

y reescribiéndola en términos de factor de aceleración, obtenemos la forma operativa del cálculo de la fracción serial que adquiere la forma siguiente:

$$f = \frac{\frac{1}{s} - \frac{1}{p}}{1 - \frac{1}{p}}. \tag{9.4}$$

Esta métrica permite ver las inconsistencias en el balance de cargas, ya que su valor debierá de tender a cero en el caso ideal, por ello un incremento en el valor de f es un aviso de granularidad fina con la correspondiente sobrecarga en los procesos de comunicación.

Costo o Trabajo Se define el costo o trabajo de resolver un problema en paralelo como el producto del tiempo de cálculo en paralelo T_p por el número de procesadores usando p y se representa por:

$$C_p = p * T_p.$$

Aceleración y Eficiencia Relativa Cuando se trabaja en más de un equipo paralelo -supongamos con p y p' procesadores con $p \geq p'$- es común comparar el desempeño entre dichos equipos. Para ello se define la aceleración relativa como:

$$S_p^{p'} = \frac{T_{p'}}{T_p}$$

para $p \geq p'$, en la cual se espera que:

$$S_p^{p'} \simeq \frac{p}{p'}$$

y eficiencia relativa como:

$$E_p^{p'} = \frac{p'}{p} S_p^{p'} = \frac{p'}{p} \frac{T_{p'}}{T_p}.$$

Análisis de Rendimiento Usando Métricas Suponiendo un Cluster con 17 Cores o núcleos[179], se muestra una ejemplificación de las métricas para un problema de ejemplo:

Cores	Tiempo	Aceleración	Eficiencia	Frac. Ser.
1	5295			
3	2538	2.08	0.69	0.218
5	1391	3.80	0.76	0.078
9	804	6.58	0.73	0.045
17	441	12.00	0.70	0.025

Nótese que en todos los casos la fracción serial disminuye sustancialmente con el aumento del número de procesadores, pero la aceleración está por debajo del valor esperado.

Suponiendo un Cluster A con 100 Cores y un Cluster B con 128 Cores, se muestra una ejemplificación de las métricas para un problema de ejemplo en ambos Clusters con los siguientes resultados:

Cores	Cluster A	Cluster B
16	9158 seg	ND
32	5178 seg	5937 seg
64	3647 seg	4326 seg
100	2661 seg	
128		2818 seg

Como se muestra en la tabla, en todos los casos el Cluster A usando como máximo 100 Cores obtiene un tiempo de cálculo inferior al que requiere Cluster B usando a lo más los 128 Cores.

Haciendo uso de las métricas de aceleración y eficiencia relativa[180] se tiene que para el Cluster B:

$$S_{128}^{32} = 5937/2818 = 2.10$$

[179] A cada procesador encapsulado se le llama Core o núcleo, logrando que la comunicación entre ellos se realiza de una forma más rápida a través de un bus interno integrado en la propia pastilla de silicio sin tener que recurrir por tanto al bus del sistema mucho más lento. Al contrario del caso de la tecnología HyperThreading que el sistema operativo los reconoce como un core en el equipo, pero estos cores virtuales distan mucho del desempeño de uno real.

[180] Aceleración relativa es $S_p^{p'} = \frac{T_{p'}}{T_p}$ para $p \geq p'$, en la cual se espera que $S_p^{p'} \simeq \frac{p}{p'}$ y eficiencia relativa es $E_p^{p'} = \frac{p'}{p} S_p^{p'} = \frac{p'}{p} \frac{T_{p'}}{T_p}$.

donde lo esperado sería:

$$S_{128}^{32} = 32/128 = 4.00,$$

para el caso de la eficiencia:

$$E_{128}^{32} = (32/128) * (5937/2818) = 0.52.$$

En el caso del Cluster A se tiene que:

$$S_{100}^{16} = 9158/2661 = 3.44$$

dónde lo esperado sería:

$$S_{100}^{16} = 16/100 = 6.35,$$

para el caso de la eficiencia:

$$E_{100}^{16} = (16/100) * (9158/2661) = 0.55.$$

Haciendo uso del mismo número de Cores base para el Cluster A que para Cluster B, se tiene que:

$$S_{100}^{32} = 5178/2661 = 1.94$$

dónde lo esperado sería:

$$S_{100}^{16} = 32/100 = 3.12,$$

para el caso de la eficiencia:

$$E_{100}^{16} = (32/100) * (5178/2661) = 0.62.$$

De todo lo anterior, se desprende que el Cluster A obtiene valores de una aceleración y eficiencias relativas ligeramente mejores que el Cluster B, pero esto no se refleja en la disminución de casi 6% del tiempo de ejecución y del uso de 28 Cores menos.

Además, el costo computacional:

$$C_p = P * T_p,$$

que para el caso del Cluster B es:

$$C_{128} = 360,704$$

y en Cluster A es:

$$C_{100} = 266,100$$

que representa una disminución de 27%; además de un factor muy importante, el Cluster A tuvo un costo monetario mucho menor con respecto del Cluster B.

9.6 Programación de Cómputo de Alto Rendimiento

Hay muchas aplicaciones a las herramientas computacionales, pero nos interesan aquellas que permitan resolver problemas concomitantes en Ciencia e Ingeniería. Muchas de estas aplicaciones caen en lo que comúnmente se llama cómputo científico. La computación científica es el campo de estudio relacionado con la construcción de modelos matemáticos, técnicas numéricas para resolver problemas científicos y de ingeniería; y su respectiva implementación computacional. La solución de estos problemas genera un alto consumo de memoria, espacio de almacenamiento o tiempo de cómputo; por ello nos interesa trabajar en computadoras que nos puedan satisfacer estas demandas.

La computación de alto rendimiento (véase 9) -High performance Computing o HPC en inglés- es la agregación de potencia de cálculo para resolver problemas complejos en Ciencia e Ingeniería o Gestión. Para lograr este objetivo, la computación de alto rendimiento se apoya en tecnologías computacionales como los Clusters, las supercomputadoras o la computación paralela. La mayoría de las ideas actuales de la computación distribuida se han basado en la computación de alto rendimiento.

La computación paralela o de alto rendimiento es una forma de cómputo en la que muchas instrucciones se ejecutan simultáneamente, operando sobre el principio de que problemas grandes, a menudo se pueden dividir en unos más pequeños, que luego son resueltos simultáneamente (en paralelo). Hay varias formas diferentes de computación paralela: paralelismo a nivel de bits, paralelismo a nivel de instrucción, paralelismo de datos y paralelismo de tareas.

El paralelismo se ha empleado durante muchos años, sobre todo en la computación de altas prestaciones, pero el interés en ella ha crecido últimamente debido a las limitaciones físicas que impiden el aumento de la frecuencia de los actuales procesadores comerciales. Como el consumo de energía –y por consiguiente la generación de calor– de las computadoras constituye una preocupación en los últimos años, la computación en paralelo se ha convertido en el paradigma dominante en la arquitectura de computadoras, principalmente en forma de procesadores multinúcleo.

Las computadoras paralelas pueden clasificarse según el nivel de paralelismo que admite su Hardware:

- Equipos con procesadores multinúcleo

- Equipos con multiprocesador (múltiples procesadores multinúcleo en una sola máquina)
- Procesadores masivamente paralelos
- Equipos con una o más tarjetas CUDA (Compute Unified Device Architecture)
- Cluster de equipos multinúcleo, multiprocesador y/o CUDAS
- Cúmulos de Clusters (Grids)

Muchas veces, para acelerar tareas específicas, se utilizan arquitecturas especializadas de computación en paralelo junto a procesadores tradicionales.

Existen múltiples vertientes en el cómputo en paralelo, algunas de ellas son:

- Cómputo en memoria compartida usando OpenMP
- Cómputo en memoria distribuida usando MPI

Como es de esperarse, los programas informáticos paralelos son más complejos y difíciles de escribir que los secuenciales, porque la concurrencia introduce nuevos tipos de errores de Software -por ello existe una creciente gama de herramientas que coadyuvan a mejorar la escritura, depuración y desempeño de los programas en paralelo-, pero la comunicación y sincronización entre diferentes subtareas son algunos de los mayores obstáculos para obtener un buen rendimiento del programa paralelo.

Además, el tiempo de programación necesario para desarrollar una aplicación paralela eficiente y eficaz para la gran mayoría de los programadores puede ser de semanas o meses en el mejor de los casos. Por ello, es necesario hacer un balance entre las diferentes opciones de paralelización para no invertir un tiempo precioso que puede no justificar dicha inversión económica, de recursos computacionales y sobre todo de tiempo.

Actualmente, en muchos centros de cómputo es una práctica común usar directivas de compilación en equipos paralelos sobre programas escritos de forma secuencial, con la esperanza que sean puestos por el compilador como programas paralelos. Esto en la gran mayoría de los casos genera códigos poco eficientes, pese a que corren en equipos paralelos y pueden usar toda

la memoria compartida de dichos equipos, el algoritmo ejecutado continúa siendo secuencial en una gran parte del código.

Si la arquitectura paralela donde se implemente el programa es UMA de acceso simétrico, los datos serán accesados a una velocidad de memoria constante. En caso contrario, al acceder a un conjunto de datos es común que una parte de estos sean locales a un procesador (con un acceso del orden de nanosegundos), pero el resto de los datos deberán ser accesados mediante red (con acceso del orden de milisegundos), siendo esto muy costoso en tiempo de procesamiento.

Por lo anterior, si se cuenta con computadoras con memoria compartida o que tengan interconexión por bus, salvo en casos particulares no será posible explotar éstas características eficientemente. Pero en la medida en que se adecuen los programas para usar bibliotecas y compiladores acordes a las características del equipo disponible -algunos de ellos sólo existen de manera comercial- la eficiencia aumentará de manera importante.

9.6.1 Programando con OpenMP para Memoria Compartida

OpenMP es una interfaz de programación de aplicaciones (API) para la programación multiproceso de memoria compartida en múltiples plataformas. Permite añadir concurrencia a los programas escritos en C, C++, Julia, Python y Fortran sobre la base del modelo de ejecución Fork-join. Está disponible en muchas arquitecturas, incluidas las plataformas de Unix, Linux y de Microsoft Windows. Se compone de un conjunto de directivas de compilador, rutinas de biblioteca, y variables de entorno que influyen en el comportamiento en tiempo de ejecución.

Definido conjuntamente por proveedores de Hardware y de Software, OpenMP es un modelo de programación portable y escalable que proporciona a los programadores una interfaz simple y flexible para el desarrollo de aplicaciones paralelas, para plataformas que van desde las computadoras de escritorio hasta supercomputadoras. Una aplicación construida con un modelo de programación paralela híbrido se puede ejecutar en un Cluster de computadoras utilizando OpenMP y MPI, o a través de las extensiones de OpenMP para los sistemas de memoria distribuida.

OpenMP se basa en el modelo *Fork-join*, paradigma que proviene de los sistemas Unix, donde una tarea muy pesada se divide en K hilos (Fork) con menor peso, para luego «recolectar» sus resultados al final y unirlos en un solo resultado (Join).

Cuando se incluye una directiva de compilador OpenMP esto implica que se incluye una sincronización obligatoria en todo el bloque. Es decir, el bloque de código se marcará como paralelo y se lanzarán hilos según las características que nos dé la directiva, y al final de ella habrá una barrera para la sincronización de los diferentes hilos (salvo que implícitamente se indique lo contrario con la directiva Nowait). Este tipo de ejecución se denomina *Fork-join*.

OpenMP también soporta el modelo de paralelismo de tareas. El equipo de hilos del bloque de código paralelo ejecuta las tareas especificadas dentro de dicho bloque. Las tareas permiten un paralelismo asíncrono. Desde la versión 4.0 lanzada en 2013 admite la especificación de dependencias entre tareas, relegando a la biblioteca de tiempo de ejecución de OpenMP el trabajo de planificar las tareas y ponerlas en ejecución. Los hilos de ejecución irán ejecutando las tareas a medida que estas estén disponibles (sus dependencias ya estén satisfechas). El uso de tareas da lugar a sincronización con una granularidad más fina. El uso de barreras no es estrictamente necesario, de manera que los hilos pueden continuar ejecutando tareas disponibles sin necesidad de esperar a que todo el equipo de hilos acabe un bloque paralelo. El uso de tareas con dependencias crea un grafo, pudiéndose aplicar propiedades de grafos a la hora de escoger tareas para su ejecución.

Salvo el uso de implementaciones de Hardware de la biblioteca de tiempo de ejecución OpenMP (p.ej. en una matriz de puertas programables FPGAs), los sobrecostes de las tareas es mayor, este sobrecoste ha de ser amortizado mediante el potencial paralelismo adicional que las tareas exponen.

Estructura del Programa en C++ Ejemplo de cálculo de Pi usando OpenMP:

```
#include <stdio.h>
// Indica si se carga lo referente a OpenMP
#ifdef _OPENMP
#include <omp.h>
int threads=omp_get_num_threads();
#else
int threads=0;
#endif
#define STEPCOUNTER 1000000000
int main (void)
```

```
{
  long i;
  double pi=0;
  printf("threads %d", threads);
#pragma omp parallel for reduction(+:pi)
  for (i=0; i < STEPCOUNTER; i++)
  {
    pi += 1.0/(i*4.0 +1.0);
    pi -= 1.0/(i*4.0 +3.0);
  }
  pi = pi*4.0;
  printf("PI = %2.16lf ",pi);
  return 0;
}
```

El compilador de OpenMP es el mismo que para los lenguajes C, C++ o Fortran respectivamente (véase 7.7). Por ello, para usarlo en C++ en línea de comandos (véase 3.4), instalamos el compilador g++, mediante:

```
# apt install g++
```

Así, para compilar con g++[181], sin usar OpenMP, usamos:

```
$ g++ pi.cpp -o pi
```

Ejecutar midiendo el tiempo[182]:

```
$ time ./pi
```

[181] Compilar fuentes en C++ solicitando que el ejecutable tenga el nombre *ejemp*:

```
$ g++ *.cpp -o ejemp
```

en este caso no se usa ninguna directiva para optimizar el ejecutable generado. Para compilar usando diversas optimizaciones (O1, -O2 o -O3) usar por ejemplo:

```
$ g++ -O1 *.cpp
```

[182] Los sistemas operativos tipo Linux/Unix cuentan con un comando básico que mide el tiempo de ejecución de cualquier comando, usando:

```
$ time ls
```

Ahora, usando el compilar para OpenMP usamos:

```
$ g++ -o pi -fopenmp pi.cpp
```

Indicar el número de hilos, por ejemplo 2:

```
$ export OMP_NUM_THREADS=2
```

Ejecutar:

```
$ time ./pi
```

9.6.2 Programando con MPI para Memoria Distribuida

Para poder intercomunicar dos o más Cores en una o en múltiples computadoras se usa la «interfaz de paso de mensajes (Message Passing Interface MPI)» (véase [58], [59], [62] y [60]), una biblioteca de comunicación para procesamiento en paralelo que puede ser usada desde lenguajes de programación como: C, C++, Pytho, Julia, Fortran. MPI ha sido desarrollado como un estándar para el paso de mensajes y operaciones relacionadas.

Este enfoque es adoptado por usuarios e implementadores de bibliotecas, en el cual se proveen a los programas de procesamiento en paralelo de portabilidad y herramientas necesarias para desarrollar aplicaciones que puedan usar el cómputo paralelo de alto desempeño.

El modelo de paso de mensajes posibilita a un conjunto de procesos que tienen solo memoria local, la comunicación con otros procesos (usando Bus o red) mediante el envío y recepción de mensajes. Por definición el paso de mensajes posibilita transferir datos de la memoria local de un proceso a la memoria local de cualquier otro proceso que lo requiera.

En el modelo de paso de mensajes para equipos paralelos, los procesos se ejecutan en paralelo, teniendo direcciones de memoria separada para cada proceso, la comunicación ocurre cuando una porción de la dirección de memoria de un proceso es copiada mediante el envío de un mensaje dentro de otro proceso en la memoria local mediante la recepción del mismo.

---

pero podemos instalar el paquete que además de medir el tiempo de ejecución nos diga que recursos se usaron el la ejecución del comando indicado, para instalarlo usamos:

```
# apt install time
```

el uso es el mismo del comando interno time.

Las operaciones de envío y recepción de mensajes es cooperativa y ocurre sólo cuando el primer proceso ejecuta una operación de envío y el segundo proceso ejecuta una operación de recepción, los argumentos base de estas funciones son:

- Para el que envía, la dirección de los datos a transmitir y el proceso destino al cual los datos se enviarán.
- Para el que recibe, debe tener la dirección de memoria donde se pondrán los datos recibidos, junto con la dirección del proceso que los envío.

Es decir:
Send(dir, lg, td, dest, etiq, com)

$\{dir, lg, td\}$ describe cuántas ocurrencias lg de elementos del tipo de dato td se transmitirán empezando en la dirección de memoria dir.

$\{des, etiq, com\}$ describe el identificador $etiq$ de destino des asociado con la comunicación com.

Recv(dir, mlg, td, fuent, etiq, com, st)

$\{dir, lg, td\}$ describe cuántas ocurrencias lg de elementos del tipo de dato td se transmitirán empezando en la dirección de memoria dir.

$\{fuent, etiq, com, est\}$ describe el identificador $etiq$ de la fuente $fuent$ asociado con la comunicación com y el estado est.

El conjunto básico de directivas (en nuestro caso sólo se usan estas) en C++ de MPI son (véase [58] y [59]):

MPI::Init	Inicializa al MPI
MPI::COMM_WORLD.Get_size	Busca el número de procesos existentes
MPI::COMM_WORLD.Get_rank	Busca el identificador del proceso
MPI::COMM_WORLD.Send	Envía un mensaje
MPI::COMM_WORLD.Recv	Recibe un mensaje
MPI::Finalize	Termina al MPI

Estructura del Programa en C++ Ejemplo de Hola_Mundo en MPI:

```
#include <stdio.h>
#include <mpi.h>
int main(int argc, char *argv[])
{
int rank, size;
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &size);
printf("Hola! Soy el %d de %d\n", rank, size);
MPI_Finalize();
return 0;
}
```

Otro ejemplo, para realizar una suma en MPI:

```
#include <iostream>
#include <iomanip>
#include <mpi.h>
using namespace std;
int main(int argc, char ** argv){
int mynode, totalnodes;
int sum = 0,startval,endval,accum;
MPI_Status status;
MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD, &totalnodes);
MPI_Comm_rank(MPI_COMM_WORLD, &mynode);
startval = 1000*mynode/totalnodes+1;
endval =1000*(mynode+1)/totalnodes;
for(int i=startval;i<=endval;i=i+1) sum = sum + i;
if(mynode!=0)
MPI_Send(&sum, 1, MPI_INT, 0, 1, MPI_COMM_WORLD);
else
for(int j=1;j<totalnodes;j=j+1){
MPI_Recv(&accum, 1, MPI_INT, j, 1,
MPI_COMM_WORLD, &status);
sum = sum + accum;
```

```
}
if(mynode == 0)
cout << "The sum from 1 to 1000 is: "<< sum << endl;
MPI_Finalize();
}
```

Existe una gran variedad de compiladores de MPI en línea de comandos (véase 3.4), algunos disponibles en Debian GNU/Linux son instalados mediante:

```
# apt install lam-runtime libmpich-dev mpi-default-dev mpich\
mpi-default-bin openmpi-bin valgrind-mpi xmpi
```

Para compilar y ejecutar es posible usar alguna de estas opciones:

mpic++, mpic++.openmpi, mpiexec.mpich, mpif90.openmpi, mpirun.lam, mpicc, mpicxx, mpiexec.openmpi, mpifort, mpirun.mpich, mpiCC, mpicxx.mpich, mpif77, mpifort.mpich, mpirun.openmpi, mpicc.mpich, mpicxx.openmpi, mpif77.mpich, mpifort.openmpi, mpitask, mpicc.openmpi, mpiexec, mpif77.openmpi, mpimsg, mpivars, mpiCC.openmpi, mpiexec.hydra, mpif90, mpipython, mpichversion, mpipython, mpiexec.lam, mpif90.mpich, mpirun

Por ejemplo, para compilar *ejemp.cpp* en *mpic++* solicitando que el ejecutable tenga el nombre *ejemp*, usamos:

```
$ mpic++ ejemp.cpp -o ejemp
```

en este caso no se uso ninguna opción de optimización en tiempo de compilación (véase 7.7), se puede hacer uso de ellas (-O1, -O2 o -O3), mediante:

```
$ mpic++ -O3 ejemp.cpp -o ejemp
```

para ejecutar el programa ya compilado en 4 procesos y medir el tiempo de ejecución (véase 7.7), usamos:

```
$ time mpirun -np 4 ejemp
```

También podemos compilar *ejemp.c* en *mpicc* solicitando que el ejecutable tenga el nombre *ejemp*:

```
$ mpicc ejemp.cpp -o ejemp
```

en este caso no se uso ninguna opción de optimización en tiempo de compilación, se puede hacer uso de ellas (-O1, -O2 o -O3), mediante:

```
$ mpicc -O3 ejemp.c -o ejemp
```

para ejecutar el programa ya compilado en 4 procesos, usamos:

```
$ mpirun -np 4 ejemp
```

Un último ejemplo, en el caso de usar *mpiCC.mpic* y *lamboot*, entoces es necesario compilar usando:

```
$ mpiCC.mpich -O2 ejemp.cpp -o ejemp -lm
```

iniciar ambiente de ejecución paralelo:

```
$ lamboot -v
```

correr usando 8 procesos por ejemplo:

```
$ mpirun.mpich -np 8 ejemp
```

correr usando 5 procesos segun lista *machines.lolb* por ejemplo:

```
$ mpirun.mpich -machinefile machines.lolb -np 5 ejemp
```

terminar ambiente de ejecución paralelo:

```
$ lamhalt -v
```

Observación 5 *Para que en la ejecución de MPI no pida la clave de usuario:*
$ ssh-keygen -t rsa
En cada pregunta responder con ENTER, para después copiar usando:
$ ssh-copy-id usuario@servidor
Ojo: Si continúa pidiendo clave, es que esta instalado rsh o lsh.

9.6.3 Esquema de Paralelización Principal-Subordinados

El esquema de paralelización Principal-Subordinados -también conocido como Maestro-Esclavo-, permite sincronizar por parte del nodo principal las tareas que se realizan en paralelo usando varios nodos subordinados, éste modelo puede ser explotado de manera eficiente si existe poca comunicación entre el principal y el subordinado -entre los subordinados no debe de existir comunicación- y los tiempos consumidos en realizar las tareas asignadas son mayores que los períodos involucrados en las comunicaciones para la asignación de dichas tareas. De esta manera se garantiza que la mayoría de los procesadores estarán trabajando de manera continua y existirán pocos tiempos muertos.

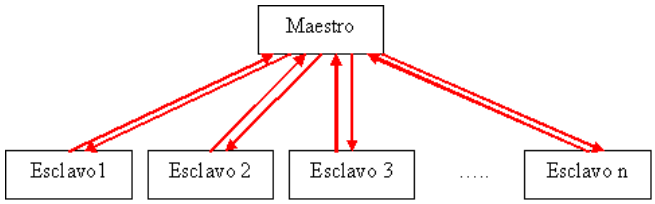

Figura 15: Esquema del Maestro-Esclavo

Donde, tomando en cuenta la implementación en estrella del Cluster, el modelo de paralelismo de MPI (véase 9.6.2) y las necesidades propias de comunicación del programa, el nodo principal tendrá comunicación sólo con cada nodo subordinado y no existirá comunicación entre los nodos subordinados, esto reducirá las comunicaciones y optimizará el paso de mensajes, algunos ejemplos de este esquema se pueden consultar en:

> Herramientas
> http://132.248.181.216/Herramientas/

Un factor limitante en este esquema es que el nodo principal deberá de atender todas las peticiones hechas por cada uno de los nodos subordinados, esto toma especial relevancia cuando todos o casi todos los nodos subordinados compiten por ser atendidos por el nodo principal.

Se recomienda implementar este esquema en un Cluster heterogéneo en donde el nodo principal sea más poderoso computacionalmente que los nodos subordinados. Si a este esquema se le agrega una red de alta velocidad y de

baja latencia, se le permitirá operar al Cluster en las mejores condiciones posibles, pero este esquema se verá degradado al aumentar el número de nodos subordinados inexorablemente.

Pero hay que ser cuidadosos en cuanto al número de nodos subordinados que se usan en la implementación en tiempo de ejecución versus el rendimiento general del sistema al aumentar estos, algunas observaciones posibles son:

- El esquema Principal-Subordinados programado usando MPI lanza P procesos (uno para el nodo principal y $P-1$ para los nodos subordinados), estos en principio corren en un solo procesador pero pueden ser lanzados en múltiples procesadores usando una directiva de ejecución, de esta manera es posible que en una sola máquina se programe, depure y sea puesto a punto el código usando mallas pequeñas (del orden de cientos de nodos) y cuando este listo puede mandarse a producción en un Cluster.
- El esquema Principal-Subordinados no es eficiente si sólo se usan dos procesadores (uno para el nodo principal y otro para el nodo subordinado), ya que el nodo principal en general no realiza los cálculos pesados y su principal función será la de distribuir tareas; los cálculos serán delegados al nodo subordinado.

Estructura del Programa Principal-Subordinados La estructura del programa se realizo para que el nodo principal mande trabajos de manera síncrona a los nodos subordinados. Cuando los nodos subordinados terminan la tarea asignada, avisan al nodo principal para que se les asigne otra tarea (estas tareas son acordes a la etapa correspondiente del método de descomposición de dominio ejecutándose en un instante dado). En la medida de lo posible se trata de mandar paquetes de datos a cada nodo subordinado y que estos regresen también paquetes al nodo principal, a manera de reducir las comunicaciones al mínimo y tratar de mantener siempre ocupados a los nodos subordinados para evitar los tiempos muertos, logrando con ello una granularidad gruesa, ideal para trabajar con Clusters.

La estructura básica del programa bajo el esquema Principal-Subordinados codificada en C++ y usando MPI (véase 9.6.2) es:

```
main(int argc, char *argv[])
```

```
{
   MPI::Init(argc,argv);
   ME_id = MPI::COMM_WORLD.Get_rank();
   MP_np = MPI::COMM_WORLD.Get_size();
   if (ME_id == 0) {
     // Operaciones del Principal
   } else {
     // Operaciones del Subordinado con identificador ME_id
    }
   MPI::Finalize();
}
```

En este único programa se deberá de codificar todas las tareas necesarias para el nodo principal y cada uno de los nodos subordinados, así como las formas de intercomunicación entre ellos usando como distintivo de los distintos procesos a la variable *ME_id*. Para más detalles de esta forma de programación y otras funciones de MPI (véase 9.6.2, [58] y [59]).

El principal factor limitante para el esquema Principal-Subordinados es que se presupone contar con un nodo principal lo suficientemente poderoso para atender simultáneamente las tareas síncronas del método, ya que este distribuye tareas acorde al número de nodos subordinados, estas si son balanceadas, ocasionaran que muchos de los procesadores subordinados terminen aproximadamente al mismo tiempo y el nodo principal tendrá que atender múltiples comunicaciones simultáneamente, degradando su rendimiento al aumentar el número de nodos subordinados que atender.

Para los factores limitantes inherente al propio esquema Principal-Subordinados, es posible implementar algunas operaciones del nodo principal en paralelo, ya sea usando equipos multiprocesador o en más de un nodo distinto a los nodos subordinados.

9.6.4 Opciones de Paralelización Híbridas

En la actualidad, casi todos los equipos de cómputo usados en estaciones de trabajo y Clusters cuentan con dos o más Cores, en ellos siempre es posible usar MPI para intercambiar mensajes entre procesos corriendo en

el mismo equipo de cómputo, pero no es un proceso tan eficiente como se puede querer. En estas arquitecturas llamadas de memoria compartida es mejor usar OpenMP o cualquiera de sus variantes para trabajar en paralelo. Por otro lado es común contar con las cada vez más omnipresentes tarjetas NVIDIA, y con los cada vez más numerosos Cores CUDA -que una sola tarjeta NVIDIA TESLA puede tener del orden de miles de ellos- y que en un futuro serán cada vez más numerosos.

Para lograr obtener la mayor eficiencia posible de estos tres niveles de paralelización, se están implementando procesos híbridos (véase [63] y [61]), en donde la intercomunicación de equipos con memoria compartida se realiza mediante MPI y la intercomunicación entre Cores que comparten la misma memoria se realiza con OpenMP, además las operaciones matriciales, vectoriales, etc. se le encargan a los numerosos Cores CUDA de las tarjetas NVIDIA.

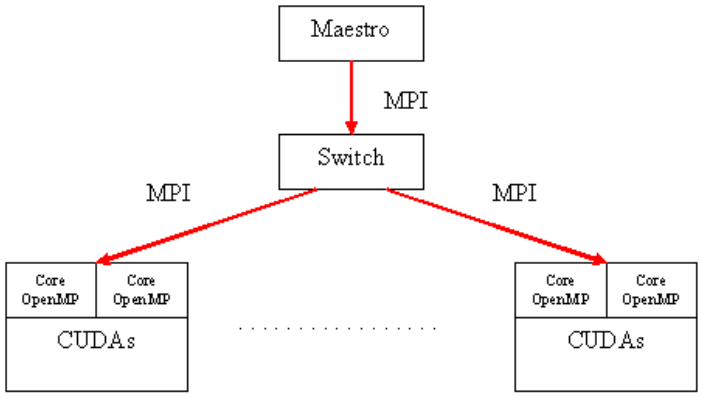

Figura 16: Paralelización Híbrida

Se han desarrollado múltiples programas que sus algoritmos resuelven ecuaciones diferenciales parciales concomitantes en Ciencias e Ingenierías que hacen uso de esta forma integradora de paralelismo. Para ello, la interconexión de equipos de memoria compartida se realizaría mediante MPI y en cada equipo de memoria compartida se manipularían uno o más subdominios mediante OpenMP -ya que cada subdominio es independiente de los demás- y la manipulación de matrices y operaciones entre matrices y vectores que requiere cada subdominio se realizarían en las tarjetas NVIDIA mediante los numerosos Cores CUDA sin salir a la RAM de la computadora.

Permitiendo así, tener una creciente eficiencia de paralelización que optimizan en gran medida los recursos computacionales, ya que todas las matrices y vectores se generarían en la GRAM de la tarjeta NVIDIA. De forma tal que sea reutilizable y que pueda usarse en problemas en los que el número de grados de libertad sea grande, permitiendo hacer uso de equipos de cómputo cada vez más asequibles y de menor costo, pero con una creciente eficiencia computacional que compiten con los grandes equipos de cómputo de alto desempeño.

9.7 Programando Desde la Nube

Existen diferentes servicios Web[183] que permiten editar, compilar y ejecutar código de diversos lenguajes y paquetes desde el navegador, esto en aras de que los estudiantes y profesores que cuenten con algún sistema de acceso a red y un navegador puedan programar en los más diversos lenguajes, IDEs y terminales sin hacer instalación alguna en su equipo de cómputo, tableta o teléfono celular.

Google Colaboratory Integrante de la G Suite for Education de Google permite a los usuarios que pertenezcan a esta Suite (como gran parte de los estudiantes de la UNAM) tener acceso desde el navegador para escribir y ejecutar código de Python (Jupyter), es posible elegir correr nuestro Notebook en una CPU, GPU o en una TPU de forma gratuita. Tiene algunas restricciones, como por ejemplo que una sesión dura 12 hrs, pasado ese tiempo se limpia nuestro ambiente y perdemos las variables y archivos que tengamos almacenados allí.

Es conveniente para principiantes que requieran experimentar con Machne Learning y Deep Learning pero sin recurrir en costos de procesamiento Cloud. Además el ambiente de trabajo ya viene con muchas librerías instaladas y listas para utilizar (como por ejemplo *Tensorflow, Sclkit-learn, Pytorch, Keras* y *OpenCV*), ahorrándonos el trabajo de configurar nuestro ambiente

[183] Cuando se trabaja desde la Web es recomendable usar el modo Privado o Incógnito para no guardar el historial de navegación, información introducida en los formularios y borrar al cerrar el navegador los datos de los sitios visitados. Pero recuerda que los sitios Web que visitamos sí guardan información de nuestra visita, nuestro proveedor de Internet también guarda constancia de nuestra visita y si descargamos algo, esto no se borra al igual que el historial de descargas, además de las marcas de páginas o favoritos se conservarán al cerrar el navegador.

de trabajo. Podemos importar nuestros archivos y datos desde *Google Drive, GitHub*, etc.

Más información sobre Google Colaboratory en:

https://colab.research.google.com/notebooks/intro.ipynb

10 Máquinas Virtuales

Entendamos por una máquina virtual a un programa de cómputo (véase [31], [37], [30] y [29]) que simula a una computadora, en la cual se puede instalar y usar otros sistemas operativos de forma simultánea como si fuese una computadora real sobre nuestro sistema operativo huésped[184].

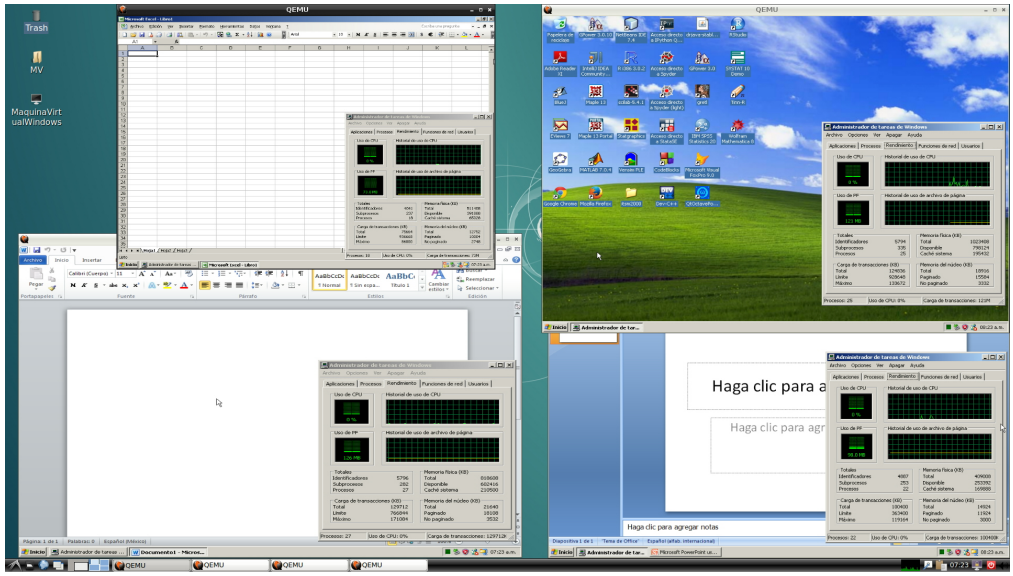

Figura 17: Sobre un equipo AMD de gama baja y 4 GB de RAM, usando como sistema operativo huésped un Linux Debian estable, se ejecutan 4 máquinas virtuales (mediante KVM) de Windows XP con diferentes aplicaciones y dentro de cada una de ellas se muestra la RAM asignada, la usada en ese momento, el uso de CPU de cada máquina virtual, entre otros datos.

Una característica esencial de las máquinas virtuales es que los procesos que ejecutan están limitados por los recursos y abstracciones proporcionados por ellas. Estos procesos no pueden escaparse de esta "computadora virtual". Uno de los usos más extendidos de las máquinas virtuales es ejecutar sistemas operativos nuevos u obsoletos adicionales a nuestro sistema habitual.

[184] Tal y como puede verse reflejado en la definición de máquina virtual, en este texto nos estamos focalizando en las máquinas virtuales de sistema. Existen otro tipo de máquinas virtuales, como por ejemplo las máquinas virtuales de proceso o los emuladores.

De esta forma podemos ejecutar uno o más sistemas operativos -Linux, Mac OS, Windows XP, 7 ó 8- desde nuestro sistema operativo habitual -GNU/Linux, Unix o Windows- sin necesidad de instalarlo directamente en nuestra computadora y sin la preocupación de que se desconfigure el sistema operativo huésped o a las vulnerabilidades del sistema virtualizado, ya que podemos aislarlo para evitar que se dañe.

10.1 Tipos de Máquinas Virtuales

Las máquinas virtuales se pueden clasificar en dos grandes categorías según su funcionalidad y su grado de equivalencia a una verdadera máquina:

- Máquinas virtuales de sistema (en inglés System Virtual Machine). También llamadas máquinas virtuales de Hardware[185], permiten a la má-quina física subyacente compartirse entre varias máquinas virtuales, cada una ejecutando su propio sistema operativo[186]. A la capa de Software que permite la virtualización se le llama monitor de máquina virtual o Hypervisor. Un monitor de máquina virtual puede ejecutarse o bien directamente sobre el Hardware o bien sobre un sistema operativo ("Host Operating System").

- Máquinas virtuales de proceso (en inglés Process Virtual Machine). A veces llamada "máquina virtual de aplicación", se ejecuta como un proceso normal dentro de un sistema operativo y soporta un solo proceso. La máquina se inicia automáticamente cuando se lanza el proceso que se desea ejecutar y se detiene para cuando éste finaliza. Su objetivo es el de proporcionar un entorno de ejecución independiente de la plataforma de Hardware y del sistema operativo, que oculte los detalles de la plataforma subyacente y permita que un programa se ejecute siempre de la misma forma sobre cualquier plataforma.

[185] La virtualización puede se por Software o con apoyo mediante el Hardware, en este último caso se obtienen varios órdenes de magnitud de rendimiento que por Software.

[186] Que sus componentes sean virtuales no quiere decir necesariamente que no existan, por ejemplo una máquina virtual puede tener unos recursos reservados de 2 GB de RAM y 20 GB de disco que se obtienen del equipo donde este ejecutando la máquina virtual. Otros dispositivos podrían realmente ser inexistentes físicamente como por ejemplo un CD-ROM que en verdad es el contenido de una imagen ISO en vez de un lector de CD de verdad.

10.2 Técnicas de Virtualización

Las Máquinas Virtuales ya tienen más de 60 años de vida su origen en los primeros días de la informática en la década de 1960, cuando el tiempo compartido para los usuarios de la computadora central era un medio de separar el Software de un sistema anfitrión físico. La máquina virtual se definió a principios de los 70 como "un duplicado eficiente y aislado de una máquina de computación real".

Las máquinas virtuales tal como las conocemos hoy en día han cobrado fuerza en los últimos 20 años a medida que las empresas adoptaron la virtualización de servidores para utilizar la potencia de computación de sus servidores físicos de manera más eficiente, reduciendo la necesidad de servidores físicos y ahorrando así espacio en el centro de datos. Debido a que las aplicaciones con diferentes requisitos de sistema operativo podían funcionar en un solo Host físico, no se requería un Hardware de servidor diferente para cada una.

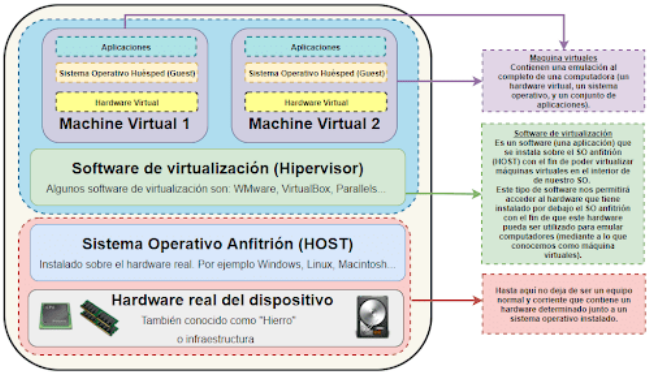

Figura 18: Qué es una Máquina Virtual

Básicamente se reconocen tres tipos de virtualización, algunas de las cuales son usadas actualmente en la gran mayoría de los sistemas operativos, ge-neralmente sin que el usuario esté consciente de que usa virtualización[187], estos son:

[187] El ejemplo más común y omnipresente es la máquina virtual del lenguaje de programación de JAVA o .Net Framework.

Emulación del Hardware Subyacente (ejecución nativa) Esta técnica se suele llamar virtualización completa -Full Virtualization- del Hardware, y se puede implementar usando un Hipervisor de Tipo I o de Tipo II:

1. Monitor de tipo I, se ejecuta directamente sobre el Hardware.
2. Monitor de tipo II, se ejecuta sobre otro sistema operativo.

Cada máquina virtual puede ejecutar cualquier sistema operativo soportado por el Hardware subyacente. Así los usuarios pueden ejecutar dos o más sistemas operativos distintos simultáneamente en computadoras "privadas" virtuales. Actualmente tanto Intel como AMD han introducido prestaciones a sus procesadores x86_64 para permitir la virtualización de Hardware.

Emulación de un Sistema no Nativo Las máquinas virtuales también pueden actuar como emuladores de Hardware, permitiendo que aplicaciones y sistemas operativos concebidos para otras arquitecturas de procesador se puedan ejecutar sobre un Hardware que en teoría no soportan. Esta técnica permite que cualquier computadora pueda ejecutar Software escrito para la máquina virtual. Sólo la máquina virtual en sí misma debe ser portada a cada una de las plataformas de Hardware.

Virtualización a Nivel de Sistema Operativo Esta técnica consiste en dividir una computadora en varios compartimentos independientes de manera que en cada compartimento podamos instalar un servidor. A estos compartimentos se les llama "entornos virtuales". Desde el punto de vista del usuario, el sistema en su conjunto actúa como si realmente existiesen varios servidores ejecutándose en varias máquinas distintas.

10.3 Otras Formas de Virtualización

El éxito de las máquinas virtuales en la virtualización de servidores llevó a aplicar la virtualización a otras áreas, como son el almacenamiento, las redes o las computadoras de escritorio. Lo más probable es que, si hay un tipo de Hardware que se está utilizando en el centro de datos, se esté explorando el concepto de virtualizarlo.

Virtualización de escritorios Implementar los escritorios como un servicio gestionado permite a las organizaciones de tecnología de la información responder más rápido a las necesidades cambiantes del entorno de trabajo y a las nuevas oportunidades. Los escritorios y las aplicaciones virtualizados también pueden distribuirse de forma rápida y sencilla a sucursales, empleados subcontratados o en otros países y trabajadores móviles que utilizan dispositivos móviles como tabletas iPad y Android.

Virtualización de redes las empresas han explorado opciones de red como servicio y la virtualización de funciones de red -NFV, Network Functions Virtualization-, que utiliza servidores de productos básicos para sustituir los aparatos de red especializados y permitir servicios más flexibles y escalables. Esto difiere un poco de la red definida por los programas informáticos, que separa el plano de control de la red del plano de reenvío para permitir un aprovisionamiento más automatizado y una gestión de los recursos de la red basada en políticas.

Una tercera tecnología, las funciones de red virtual, son servicios basados en programas informáticos que pueden ejecutarse en un entorno NFV, incluidos procesos como el enrutamiento, el cortafuegos, el equilibrio de la carga, la aceleración de la WAN y el cifrado.

Máquinas Virtuales y Contenedores El crecimiento de las máquinas virtuales ha dado lugar a un mayor desarrollo de tecnologías como los contenedores, que llevan el concepto un paso más allá y está ganando atractivo entre los desarrolladores de aplicaciones Web. En el entorno de un contenedor, una sola aplicación, junto con sus dependencias, puede ser virtualizada. Con muchos menos gastos generales que una máquina virtual, un contenedor sólo incluye binarios, bibliotecas y aplicaciones.

Mientras que algunos piensan que el desarrollo de contenedores puede "matar" a la máquina virtual, hay suficientes capacidades y beneficios de las máquinas virtuales que mantienen la tecnología en movimiento. Por ejemplo, las máquinas virtuales siguen siendo útiles cuando se ejecutan múltiples aplicaciones juntas o cuando se ejecutan aplicaciones heredadas en sistemas operativos más antiguos.

Además, algunos opinan que los contenedores son menos seguros que los hipervisores de las máquinas virtuales porque los contenedores tienen un solo sistema operativo que las aplicaciones comparten, mientras que las máquinas

virtuales pueden aislar la aplicación y el sistema operativo.

Máquinas Virtuales, 5G y Edge Computing Las máquinas virtuales son vistas como parte de nuevas tecnologías como el 5G y el Edge Computing. Por ejemplo, los proveedores de infraestructura de escritorio virtual -VDI, Virtual Desktop Infrastructure- como Microsoft, VMware y Citrix están buscando formas de extender sus sistemas VDI a los empleados que ahora trabajan en casa a causa de la pandemia.

Como muchas otras tecnologías que se utilizan hoy en día, éstas no se habrían desarrollado si no fuera por los conceptos originales de máquinas virtuales introducidos hace décadas.

10.4 Aplicaciones de las Máquinas Virtuales de Sistema

Cada uno de los sistemas operativos que virtualizamos -con su propio sistema operativo llamado sistema operativo «invitado (Guest)»- es independiente de los otros sistemas operativos. De este modo, en caso que una de las máquinas virtuales deje de funcionar, el resto seguirá funcionando. Una máquina virtual dispone de todos los elementos de un equipo de cómputo real, de disco duro, memoria RAM, unidad de CD o DVD, tarjeta de red, tarjeta de vídeo, etc., pero a diferencia de un equipo de cómputo real estos elementos en vez de ser físicos son virtuales. Así, una vez instalado un sistema operativo en la máquina virtual, podemos usar el sistema operativo virtualizado del mismo modo que lo usaríamos si lo hubiéramos instalado en nuestro equipo de cómputo.

Varios sistemas operativos distintos pueden coexistir sobre la misma computadora trabajando simultáneamente, en sólido aislamiento el uno del otro, por ejemplo para probar un sistema operativo nuevo sin necesidad de instalarlo directamente. La máquina virtual puede proporcionar una arquitectura de instrucciones que sea algo distinta de la verdadera máquina, es decir, podemos simular Hardware. Además, todos los elementos de una máquina virtual se encapsulan en un conjunto pequeño de archivos -en KVM/QEMU es solo un archivo-, esto permite que podamos pasar un sistema operativo virtual de un equipo de cómputo a otro y realizar copias de seguridad de forma fácil y rápida.

La gran mayoría de los manejadores de máquinas virtuales -KVM, Vir-

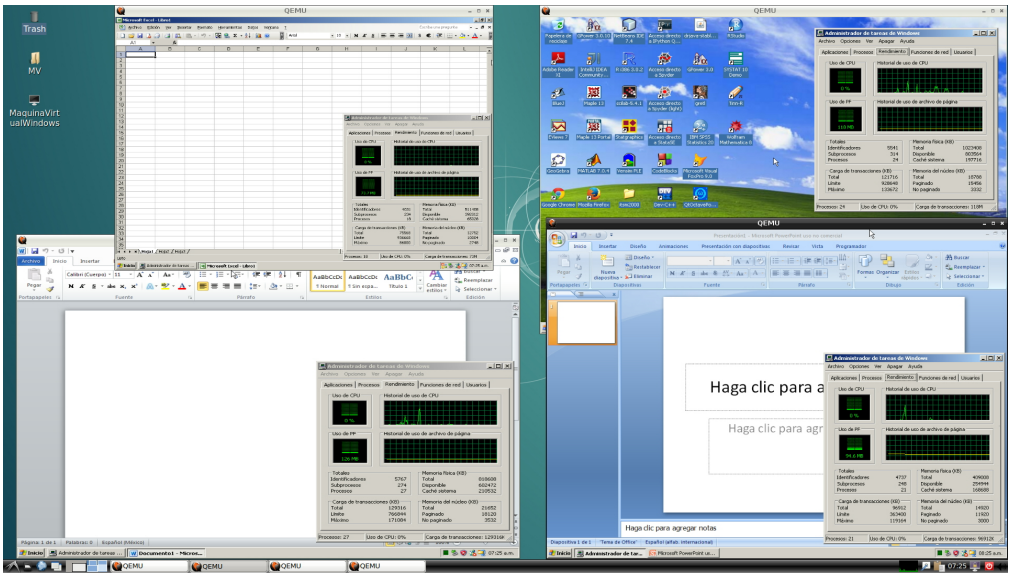

Figura 19: Al poder correr diferentes sistemas operativos y/o versiones del mismo en donde podemos instalar diversas aplicaciones antagónicas que no podrían coexistir en un sólo sistema operativo, nos permite ampliar el uso de nuestro equipo de cómputo.

tualBox o VMWare- permiten instalar prácticamente cualquier sistema operativo -por ejemplo Linux, Android, Mac OS X, Windows, Chrome OS, etc.-. Sin embargo existen otros manejadores de máquinas virtuales -Virtual PC, Hiper-V o Parallels- que están principalmente destinados a virtualizar Windows.

La virtualización es una excelente opción hoy en día, ya que las máquinas actuales -Laptops, Desktops y servidores- en la mayoría de los casos están siendo "subutilizados" -estos cuentan con una gran capacidad de cómputo, disco duro y memoria RAM- ya que no se utilizan todos los recursos todo el tiempo, teniendo un uso promedio que oscila entre 30% a 60% de su capacidad total. Permitiendo así, ejecutar varias máquinas virtuales en un sólo equipo físico aumentando el porcentaje de uso de los recursos de cómputo disponibles -en el caso de virtualizar servidores, a este proceso se le conoce como consolidación de servidores-. Así, la consolidación de servidores contribuye a reducir el coste total de las instalaciones necesarias para mantener los servicios, permitiendo un ahorro considerable de los costos asociados -

energía, mantenimiento, espacio, administración, etc.-, esto se hace patente en la «computación en la nube (Cloud Computing)» muy en boga actualmente.

10.5 Ventajas y Desventajas

Como toda tecnología, la virtualización tiene ventajas y desventajas, las cuales deben de ser sopesadas en cada ámbito de implementación. Lo que es un hecho que permite en un mismo equipo de cómputo ejecutar más de un sistema operativo o distintas versiones del mismo.

Pero queda claro que uno de los inconvenientes de las máquinas virtuales, es que agregan gran complejidad al sistema en tiempo de ejecución. Esto tiene como efecto la ralentización del sistema, es decir, el programa no alcanzará la misma velocidad de ejecución que si se instalase directamente en el sistema operativo «anfitrión (Host)» o directamente sobre la plataforma de Hardware, sin embargo, a menudo la flexibilidad que ofrecen compensa esta pérdida de eficiencia.

Si la virtualización es por Hardware, la velocidad de ejecución es más que aceptable para la mayoría de los casos, por ejemplo, en el caso de usar KVM/QEMU soporta máquinas virtuales de hasta 255 CPUs y 4 TB de RAM, y el rendimiento de aplicaciones como Oracle, SAP, LAMP, MS Exchange sobre máquinas virtuales puede oscilar entre el 95% y el 135% comparado con su ejecución en servidores físicos. Además, se ha conseguido ejecutar hasta 600 máquinas virtuales en un solo servidor físico.

10.5.1 Ventajas

Además de permitir ejecutar múltiples sistemas operativos, diferentes versiones de un mismo sistema pero con diferente Software que en principio puede ser incompatible entre sí. Para usuarios de Windows, el hecho en sí, de no tener porque lidiar con problemas derivados de virus y antivirus le confiere una gran ventaja desde el punto de vista administrativo y del usuario final. Además, permite una administración centralizada, ya que todas las máquinas virtuales tendrían la misma configuración y paquetes sin importar el Hardware subyacente en las que se ejecute el sistema operativo huésped.

En el caso de instituciones educativas de cualquier nivel académico, es común que en un mismo equipo de cómputo sea necesario ejecutar por un lado diferentes versiones de sistemas operativos -por ejemplo Linux, Windows XP,

Windows 7, etc.- y por otro lado, en un sistema operativo, ejecutar diferentes versiones de un mismo paquete -generalmente no se pueden tener instalados simultáneamente más de una versión-.

Las máquinas virtuales son una verdadera opción para coexistir simultáneamente diferentes versiones de sistemas operativos y en un mismo sistema máquinas virtuales ejecutando las diversas versiones de un mismo programa de cómputo, además se pueden configurar para que al momento de iniciarlas siempre se ejecuten a partir de una configuración e instalación base, de tal forma que al ser lanzadas, el usuario pueda instalar, configurar e inclusive dañar la máquina virtual, pero al reiniciarse la máquina virtual en una nueva sesión, se regresa a la configuración de la versión base, de esta forma no hay posibilidad de infección de virus entre diversos lanzamientos de sesiones de la máquina virtual, la actualización es centralizada y se puede hacer por red, sin intervención del usuario.

Por ello, es una opción viable y común tener en una máquina un sistema huésped como Debian GNU/Linux Estable y dentro de él, un grupo de máquinas virtuales de Windows -Windows XP, Windows 7, etc.-, en los que cada máquina virtual tenga instalado Software agrupado por las características del sistema operativo necesario para ejecutar a todas las aplicaciones seleccionadas -por ejemplo agrupados por la versión de Service Pack-.

Por otro lado, si se desconfigura un sistema operativo virtualizado es sumamente fácil de restaurar si lo comparamos con una máquina real. Si tomamos las precauciones necesarias podemos restaurar el estado que tenía un sistema operativo virtualizado, de forma fácil y rápida. Si hablamos del entorno empresarial, la virtualización de sistemas operativos supone un ahorro económico y de espacio considerable. Ya que mediante el uso de la virtualización evitamos la inversión en multitud de equipos físicos, esto supone un ahorro importante en mantenimiento, en consumo energético, espacio y procesos administrativos.

Por otro lado, mediante la virtualización y el balanceo dinámico podemos incrementar las tasas de prestación de servicios de un servidor del siguiente modo. Si disponemos de un servidor Web podemos asignar recursos adicionales al servidor, como por ejemplo memoria RAM y CPU en los picos de carga para evitar que el servidor se caiga y de este modo incrementar la tasa de eficiencia. Una vez finalizado el pico de carga podemos desviar los recursos aplicados al servidor Web a otra necesidad que tengamos. Por lo tanto, aparte de mejorar la tasa de servicio se pueden optimizar los recursos.

Si estamos usando una máquina virtual en un entorno de producción,

podemos ampliar los recursos de un sistema operativo o servidor de una forma muy sencilla, tan solo tenemos que acceder al Software de virtualización y asignar más recursos. Además, es fácil crear un entorno para realizar pruebas de todo tipo aislado del resto del sistema. Así, las máquinas virtuales y la virtualización permiten usar un solo servicio por servidor virtualizado de forma sencilla, de este modo aunque se caiga uno de los servidores virtualizados los otros seguirán funcionando.

En resumen, la virtualización permite ofrecer un servicio más rápido, sencillo a usuarios (académicos, estudiantes, clientes, etc.) y es un pilar que debe ser considerado en una escuela, universidad o compañía en su proceso de transformación o consolidación, permitiendo escalonar y ser creativos a la hora de atender las necesidades crecientes y cambiantes de los usuarios; y contar con servicios agregados, ágiles y adaptables a los constantes cambios de tecnología de Hardware y Software permitiendo escalar a la hiperconvergencia hacia la nube.

10.5.2 Desventajas

Entre las principales desventajas de virtualizar sistemas propietarios[188] como Windows (véase 14.1) -no así los sistemas libres como Debian GNU/Linux (véase 14.2)- es que se puede violar el sistema de licenciamiento (véase 14.5) del Software instalado en las máquinas virtuales, esto es especialmente importante cuando se usa en más de una máquina, pues la licencia usada para la instalación es violada cuando se tiene más de una copia de la máquina virtual o se ejecutan múltiples instancias de la máquina virtual.

En el caso de Windows XP Home, no se infringe la licencia mientras se cuente con número de licencias igual al máximo número de máquinas virtuales lanzadas simultáneamente. Para otras versiones del sistema operativo Windows como es Windows XP Profesional, la virtualización se maneja con licencias adicionales a la del sistema operativo original y se debe de contar con tantas licencias como el máximo número de máquinas virtuales lanzadas simultáneamente. Además, es necesario contar con el tipo de licencia adecuada para virtualizar a todos y cada uno de los paquetes de cómputo instalados en cada máquina virtual y en las instancias para el número de máquinas

[188] Según la Free Software Foundation (véase [21]), el «Software libre» se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, y estudiar el mismo, e incluso modificar el Software y distribuirlo modificado. Así, un Software que no es libre, es llamado «Software privativo o propietario».

virtuales lanzadas simultáneamente en uno o más equipos.

Para usar una máquina virtual en condiciones favorables, necesitamos un equipo de cómputo potente. Debemos tener en cuenta que si usamos dos sistemas operativos de forma simultánea estamos empleando hasta el doble de recursos. No obstante, cualquier equipo de cómputo doméstico de gama baja actual dispone de los recursos suficientes para usar una o más máquinas virtuales.

Los sistemas operativos y los programas se ejecutarán con mayor lentitud en las máquinas virtuales. Esto es debido a que las máquinas virtuales no pueden sacar un rendimiento ideal del Hardware que tenemos en nuestro equipo. Cuanto más potente sea nuestro equipo de cómputo menos se notará la pérdida de rendimiento.

Si tenemos un problema -de Hardware o Software- en el equipo de cómputo que aloja el sistema operativo anfitrión puede caerse el servicio en la totalidad de máquinas virtuales. Por lo tanto el equipo de cómputo que hace funcionar la máquina virtual es una parte crítica del proceso de virtualización.

A pesar de los inconvenientes que se citan en este apartado, bajo nuestro punto de vista, la virtualización y las máquinas virtuales proporcionan unas ventajas y una flexibilidad que compensan claramente los inconvenientes que acabamos de citar.

10.5.3 Consideraciones Técnicas y Legales de la Virtualización

Como se mostrará en la siguiente sección, virtualizar sistemas operativos -Linux, Unix, Windows entre otros- no representa ningún problema técnico, pero no es el caso en cuanto a las implicaciones legales de hacer la virtualización que involucra el almacenamiento, distribución y el número de veces que se ejecuta simultáneamente una máquina virtual en uno o múltiples equipos, ya que en general, la máquina virtual está contenida en un sólo archivo que facilita su distribución y almacenamiento, violando de esta forma la licencia de algunos sistemas operativos y/o programas instalados en el mismo.

En el caso de virtualizar cualquier sistema operativo libre como Debian GNU/Linux (véase 14.2), el tipo de licencia que tiene, permite y alienta su uso para cualquier fin que uno desee, por ello no hay ningún problema en virtualizarlo, no así el caso de hacerlo en sistemas operativos propietarios tipo Windows, la licencia (véase 14.1) restringe su uso a un sólo equipo de cómputo y en muchos casos prohíbe expresamente su virtualización. Además hay que

tomar en cuenta el resto del Software instalado en el sistema operativo, ya que estos también tienen sus propias restricciones legales a su uso y número de veces que se puede ejecutar simultáneamente un paquete dado.

Esto es especialmente importante cuando se usa en más de una máquina física, la misma máquina virtual, pues la licencia usada para la instalación es violada cuando se tiene más de una copia de la máquina virtual o se ejecutan múltiples instancias de la máquina virtual, esta violación de licencia es suficiente para ser sujeto a multas o incluso cárcel por dicho ilícito (véase 14.5).

Por otro lado, cada vez que se adquiere una licencia de uso de algún Software que no caduque -la cual implica un alto costo monetario-, esta pueda seguir siendo usada en una máquina virtual con una versión tal vez obsoleta del sistema operativo que la soporte, pero corriendo en un sistema huésped moderno y protegido en Hardware de última generación de forma lícita y con el consiguiente ahorro económico.

10.6 Máquinas Virtuales en la Educación, Ciencias e Ingeniería

Como hemos visto en las secciones anteriores, el uso de las máquinas virtuales es variado, flexible y permite ser usado en diversos ámbitos de la educación, del desarrollo y prueba de programas de cómputo y en general, en Ciencias e Ingeniería. Algunas de las utilidades y beneficios que podemos sacar de una máquina virtual son los siguientes:

- Para aprender a instalar, configurar y usar diversos sistemas operativos, además de poder probar diversas opciones de configuración en ellos. El proceso de instalación de la máquina virtual no requiere crear particiones adicionales en nuestro disco ni alterar la configuración de la máquina anfitriona y podemos trasladar la máquina virtual a uno o más equipos de cómputo que la soporte.

- Para usar un Software que no está disponible en nuestro sistema operativo habitual. Por ejemplo, si somos usuarios de Linux y queremos usar Photoshop, lo podemos hacer a través de una máquina virtual.

- En ocasiones tenemos que usar Software que únicamente se puede ejecutar en sistemas operativos obsoletos -Windows 98 por ejemplo-,

podemos crear una máquina virtual con dicho sistema y usar el Software de forma aislada sin preocuparnos de sus vulnerabilidades.

- Podemos experimentar en el sistema operativo que corre dentro de la máquina virtual haciendo cosas que no nos atreveríamos a realizar con nuestro sistema operativo habitual, como por ejemplo, instalar Software no seguro que consideramos sospechoso, etc.

- En muchos casos se quiere aprender a instalar, administrar y usar equipo al que no tenemos acceso como un equipo multiCore, con tarjeta CUDA instalada o un Cluster de múltiples nodos multiCore. Esto es posible hacer mediante el uso de máquinas virtuales en un equipo de gama media.

- Si se hace el adecuado aislamiento de una máquina virtual en la que se instale alguna versión de Windows, esta puede ser inmune a los virus y no requiere el uso de antivirus.

- En el caso de instituciones educativas de cualquier nivel académico, es común que en un mismo equipo de cómputo sea necesario ejecutar por un lado diferentes versiones de sistemas operativos -Linux, Windows XP, Windows 7, etc.- y por otro lado, en un sistema operativo, ejecutar diferentes versiones de un mismo paquete -generalmente no se puede tener instalada simultáneamente más de una versión- esto se logra con máquinas virtualizadas ad hoc coexistiendo en una misma máquina física.

- Podemos crear/simular una red de equipos de cómputo con tan solo un equipo de cómputo. Esta red de equipos de cómputo virtualizados la podemos usar con fines formativos y de este modo adquirir pericia sobre administración de redes.

- Si eres un desarrollador de Software puedes revisar si el programa que estas desarrollando funciona correctamente en varios sistemas operativos y/o navegadores de Web.

- Podemos usar las máquinas virtuales para hacer SandBox[189] con el fin

[189] Un sistema de aislamiento de procesos o entorno aislado, a menudo usando como medida de seguridad para ejecutar programas con seguridad y de manera separada del sistema anfitrión.

de ejecutar aplicaciones maliciosas o abrir correos sospechosos en un ambiente controlado y seguro.

- Para probar versiones Alfa, Beta y Release Candidate de ciertos programas y sistemas operativos.
- Para montar un servidor Web, un servidor VPN, un servidor de correo o cualquier otro tipo de servidor.
- Para probar multitud de programas en Windows y evitar que se ensucie el registro mediante las instalaciones y desinstalaciones de los programas
- Consolidar servidores, i.e. lo que ahora hacen varias máquinas, se pueden poner en un solo equipo físico dentro de varias máquinas virtuales independientes o interactuando entre ellas según se requiera.
- Mantenimiento y pruebas de aplicaciones sin necesidad de adaptar nuevas versiones del sistema operativo.
- Aumentar la disponibilidad al reducir tiempo de parada y mantenimiento. Ya que la máquina virtual está hecha, se pueden poner a trabajar una o más copias en un equipo o en múltiples máquinas físicas en cuestión de segundos, permitiendo la continuidad de un negocio o servicio y de recuperación ante desastres.
- Reducir costos de administración ya que se reducen y agilizan las políticas de respaldo y recuperación, además de optimizar los recursos disponibles permitiendo escalabilidad al crecer con contención de costos, mejorando la eficiencia energética al usar un menor número de equipos de cómputo.
- Permite incursionar en la estrategia de nube híbrida proactiva creando un conjunto de marcos de decisión en la nube y procesos para evaluar las oportunidades de computación en la nube en función de las necesidades y cargas de trabajo de los usuarios, por ejemplo el uso de supercómputo rentado.
- Establecer las habilidades, herramientas y procesos para un entorno dinámico e híbrido al asociarse los educadores y los especialistas en tecnologías de información para realizar un inventario de habilidades

y competencias, e identificar oportunidades de capacitación y áreas de vulnerabilidad potencial.

10.7 ¿Qué Necesito para Crear y Usar una Máquina Virtual?

Actualmente la virtualización de un sistema operativo se puede implementar por Software o por Hardware, lo único que precisamos para poder usar una máquina virtual es un equipo de cómputo e instalar y configurar el programa manejador de la máquina virtual. Cuanto más potente y actual sea el equipo de cómputo del que dispongamos, mejor experiencia obtendremos trabajando con sistemas operativos virtualizados.

Algunos de los puntos importantes para obtener un rendimiento óptimo del sistema operativo virtualizado son los siguientes:

- Preferentemente disponer de un procesador que disponga de capacidad de virtualización por Hardware (Intel VTx/AMD-V). Casi cualquier equipo de cómputo actual dispone de un procesador apto para virtualizar sistemas operativos por Hardware.
- Disponer de espacio suficiente en el disco duro[190], es preferible disponer de un disco de estado sólido (SSD) por sus velocidades de lectura-escritura.
- Necesitamos disponer de memoria RAM suficiente y adecuada[191]. Cuanta más memoria RAM y cuanto más rápida sea, mejores resultados de virtualización obtendremos.
- Sin duda el hecho de tener una buena tarjeta gráfica también ayudará a disponer de una mejor experiencia de virtualización.

Para empezar, debemos decidir qué manejador de máquinas virtuales usar, si trabajamos en Debian GNU/Linux, podemos usar QEMU/KVM y lo instalamos mediante:

[190] Una máquina virtual con Windows XP ocupa por lo menos 2 GB en disco y una con Windows 7 ocupa por lo menos 4 GB en disco.

[191] La cantidad de memoria RAM ideal dependerá del sistema operativo que queremos virtualizar y del número de sistemas operativos que queramos virtualizar de forma simultánea. Si tan solo queremos virtualizar un sistema operativo con 2 o 3 GB de RAM debería ser suficiente.

```
# apt install qemu-kvm qemu qemu-utils
```

QEMU/KVM soporta emulación de IA-32 (x86) PC, AMD64 PC, MIPS R4000, Sun's SPARC sun4m, Sun's SPARC sun4u, ARM development boards (Integrator/CP y Versatile/PB), SH4 SHIX board, PowerPC (PReP y Power Macintosh), y arquitecturas ETRAX CRIS.

Ejemplo 1 *Si hemos descargado la imagen ISO de algún sistema operativo, por ejemplo la de Knoppix*[192]*, la podemos usar mediante:*

```
$ kvm -m 1024 -cdrom KNOPPIX_ V8.2-2018-05-10-EN.iso
```

Aquí se usa la arquitectura por omisión y memoria de 1024 MB.

Ejemplo 2 *Instalación y uso de una máquina virtual para Debian GNU/Linux estable a partir del archivo ISO*[193]*, para ello, primero necesitamos:*

Generar el disco virtual que la contendrá, por ejemplo de 10 GB con el nombre debianStable.img mediante:

```
$ qemu-img create -f qcow2 debianStable.img 10G
```

Después, instalar la imagen de Debian estable en el disco virtual genera-do en el paso anterior, solicitando a KVM que una vez terminada la instalación no haga el reinicio de la máquina virtual, esto mediante:

```
$ kvm -no-reboot -boot d -cdrom debian-802-i386-netinst.iso \
-hda debianStable.img -m 400
```

Ahora podemos usar la máquina virtual mediante:

```
$ kvm -hda debianStable.img -m 800
```

Ejemplo 3 *Instalación y uso de una máquina virtual para Windows XP, en este caso necesitamos:*

Crear el disco virtual, por ejemplo de 10 GB mediante:

```
$ qemu-img create -f qcow2 WindowsXP.img 10G
```

Hacer la instalación básica a partir del ISO, mediante:

```
$ kvm -no-reboot -boot d -hda WindowsXP.img -m 400 \
-localtime -cdrom es_ winxp_ pro_ with_ sp2.iso
```

Y concluir la instalación mediante:

```
$ kvm -no-reboot -boot c -hda WindowsXP.img -m 400 \
```

[192] Knoppix es una versión Live ampliamente conocida y completa de GNU/LINUX, se puede descargar de: https://www.knopper.net/knoppix/

[193] Diversas imágenes ISO del proyecto Linux Debian se pueden descargar de: https://www.debian.org/CD/

-localtime -cdrom es_winxp_pro_with_sp2.iso
Ahora podemos usar la máquina virtual mediante:
$ kvm -boot c -hda WindowsXP.img -m 400 -localtime

10.8 ¿Cómo Funciona una Máquina Virtual?

Explicar el funcionamiento a detalle de una máquina virtual es engorroso y está fuera del alcance del propósito de este texto. No obstante a grandes rasgos podemos decir que una máquina virtual es un Software que mediante una capa de virtualización[194] se comunica con el Hardware que tenemos disponible en nuestro equipo de cómputo consiguiendo de este modo emular la totalidad de componentes de un equipo de cómputo real. De este modo la máquina virtual será capaz de emular un disco duro, una memoria RAM, una tarjeta de red, un procesador, etc.

Una vez que sabemos esto, cuando abrimos una máquina virtual, como por ejemplo Virtualbox (véase [32]), nos encontramos con un entorno gráfico (en el caso de usar QEMU/KVM se usa la línea de comandos para crear y usar las máquinas virtuales, pero también se cuentan con entornos gráficos que usan la línea de comandos internamente) que nos permitirá configurar y asignar recursos a cada uno de los componentes físicos que emula la máquina virtual. En prácticamente la totalidad de máquinas virtuales debemos definir detalles del siguiente tipo:

- Tipo de procesador a usar
- Espacio que queramos asignar al disco duro.
- Memoria RAM que queremos asignar a la máquina virtual.
- La memoria de nuestra tarjeta gráfica.
- La configuración de red.
- etc.

[194] La capa de virtualización es un sistema de archivos propietario y una capa de abstracción de servicio del Kernel que garantiza el aislamiento y seguridad de los recursos entre distintos contenedores. La capa de virtualización hace que cada uno de los contenedores aparezca como servidor autónomo. Finalmente, el contenedor aloja la aplicación o carga de trabajo.

Una vez configurados estos parámetros habremos creado una máquina virtual para instalar un sistema operativo, de este modo tan solo tendremos que abrir la máquina virtual que se acaba de crear e instalar el sistema operativo tal y como si se tratará de un equipo de cómputo real.

10.9 Aplicaciones y Paquetes Disponibles

A continuación mencionaremos algunas de las aplicaciones más conocidas y universalmente disponibles y/o usadas en los Sistemas Operativos GNU Linux/BSD, tanto en el ámbito personal, es decir, distribuciones usadas para fines particulares (hogar), como en el ámbito profesional, es decir, en el área de servidores de organizaciones y empresas.

Es importante destacar que, en este listado no se incluirán aquellas tecnologías de virtualización que vienen como una solución integrada, todo en uno o llave en mano, tales como *Promox*.

VirtualBox Software desarrollado por Oracle multiplataforma, capaz de virtualizar prácticamente la totalidad de sistemas operativos con arquitectura x86/amd64. La base de este Software dispone de una licencia GPL2, mientras que el Pack de extensiones que añaden funcionalidades están bajo licencia privativa. Virtualbox es gratuito para un uso no comercial.

Es un Hipervisor de Tipo 2 multiplataforma, es decir, solo debe y puede ser ejecutado (instalado) en cualquier Host (Ordenador) con alguna de las versiones vigentes o antiguas de Sistemas Operativos Windows, Linux, Macintosh, Solaris, OpenSolaris, OS/2 y OpenBSD.

Posee un continuo y progresivo ciclo de desarrollo con lanzamientos frecuentes, que la convierten en una excelente alternativa a otras soluciones parecidas, pero con una muy apreciable cantidad de características y funciones, sistemas operativos invitados compatibles y plataformas en las que se puede ejecutar.

En la mayoría de las distribuciones GNU Linux/BSD se encuentra dicha aplicación incluida en los repositorios, por lo que, con la siguiente orden de comando suele instalarse en todas ellas:

```
# apt install virtualbox
```

Vale destacar para VirtualBox que, al usar esta aplicación siempre es ideal la instalación de las «Guest Additions» y el «Extension Pack». Por ende,

para esto y otras formas de instalación lo ideal es visitar el enlace oficial de VirtualBox.

Vmware Workstation Player Software privativo multiplataforma desarrollado por EMC Corporation y que es utilizado ampliamente en el entorno profesional en las áreas del Cloud Computing entre muchas otras. Al igual que Virtualbox, esta máquina virtual nos permite virtualizar infinidad de sistemas operativos. Vmware dispone de muchas soluciones de virtualización y prácticamente todas son de pago, no obstante Vmware Workstation Player es totalmente gratuita para un uso no comercial.

Parallels aunque se trata de una máquina virtual multiplataforma, acostumbra a ser usado por los usuarios del sistema operativo OS X de Apple que desean virtualizar el sistema operativo Windows. Esta máquina virtual es de pago y únicamente puede virtualizar los sistemas operativos Windows y Mac OS. Quien quiera probar Parallels lo puede hacer descargando la versión de prueba.

Windows Virtual PC Software gratuito y privativo propiedad de Microsoft que se puede usar tanto en Windows como en Mac OS. Virtual PC está destinado únicamente a virtualizar sistemas operativos Windows.

Virtualización: GNOME Boxes es una aplicación nativa del entorno de Escritorio GNOME, que se utiliza para acceder a sistemas remotos o virtuales. Boxes o Cajas, utiliza las tecnologías de virtualización de *QEMU*, *KVM* y *Libvirt*.

Además, requiere que la CPU sea compatible con algún tipo de virtualización asistida por Hardware (Intel VT-x o AMD-V, por ejemplo); por lo tanto, GNOME Boxes no funciona en las CPUs con procesador Intel Pentium/Celeron, ya que, carecen de esta característica.

En la mayoría de las distribuciones GNU Linux/BSD se encuentra dicha aplicación incluida en los repositorios, por lo que, con la siguiente orden de comando suele instalarse en todas ellas:

```
# apt install gnome-boxes
```

Vale destacar para GNOME Boxes que, es una herramienta muy sencilla dirigida a usuarios novatos, permite descargar desde la aplicación diversas

imágenes y no incorpora demasiadas opciones de configuración que suelen ser muy conocidas y usadas en otras, tales como VirtualBox.

Virt-Manager es una interfaz de usuario de escritorio para la administración de un Gestor de Máquinas Virtuales a través de *Libvirt*. Está dirigida principalmente a las máquinas virtuales gestionadas por *KVM*, pero también maneja las gestionadas por *Xen* y *LXC*.

Virt-Manager presenta una vista resumida de los dominios en ejecución, su rendimiento en vivo y estadísticas de utilización de recursos. Los asistentes permiten la creación de nuevos dominios, y la configuración y ajuste de la asignación de recursos de un dominio y el Hardware virtual. Un visor de cliente *VNC* y *SPICE* integrado presenta una consola gráfica completa para el dominio invitado.

En la mayoría de las distribuciones GNU Linux/BSD se encuentra dicha aplicación incluida en los repositorios, por lo que, con la siguiente orden de comando suele instalarse en todas ellas:

```
# apt install virt-manager
```

Vale destacar para Virt-Manager que, es también es una herramienta sencilla, aunque mucho más completa que GNOME Boxes, por lo que se puede considerar para usuarios medios o avanzados de primer nivel, dado que, fácilmente es capaz de permitir la gestión de todo el ciclo de vida de las máquinas virtuales existentes.

QEMU / KVM es un emulador y virtualizador de máquinas genérico y de código abierto, capaz de ejecutar sistemas operativos y programas hechos para una máquina en una máquina diferente con muy buen rendimiento, y capaz de lograr un rendimiento casi nativo ejecutando el código del huésped directamente en la CPU del Host.

KVM es una solución de virtualización completa para Linux en Hardware x86 que contiene extensiones de virtualización (Intel VT o AMD-V) que consiste en un módulo de Kernel cargable, que proporciona la infraestructura de virtualización del núcleo y un módulo específico del procesador. Y actualmente funciona inmerso dentro de QEMU.

En la mayoría de las distribuciones GNU Linux/BSD se encuentra dicha aplicación incluida en los repositorios, por lo que, con la siguiente orden de comando suele instalarse en todas ellas:

```
# apt install qemu-kvm
```

además, en caso necesario podemos hacer uso de una interfaz gráfica para trabajar con QEMU/KVM, tenemos varias opciones, algunas de ellas son:

```
# apt install qemu-system-gui
# apt install aqemu
# apt install qemuctl
# apt install virt-manager
# apt install gnome-boxes
```

Vale destacar que QEMU/KVM es también es una herramienta muy completa, ya que no solo emula (x86, x86-AMD64, MIPS, Arm, PowerPC, SPARC, etc.) sino que virtualiza, a diferencias de otros iguales de avanzadas como WMWare, que solo permite virtualizar. Para conocer las opciones de emulación usamos:

```
$ apt search qemu-system-
```

Cockpit es una interfase Web Open Source que permite manejar máquinas virtuales de KVM que provee acceso a sistemas Linux permitiendo la administración, manejo y monitoreo a través de una interfaz gráfica intuitiva, para usarlo hay que instalar primero KVM y luego hacer:

```
# apt install cockpit cockpit-machines
```

Librerías y Paquetes relacionados Estos últimos 3 paquetes mencionados suelen instalar otros conexos (relacionados) como dependencias, por lo que, en caso de ser necesario, pueden instalar los mismos, junto a sus dependencias y otros paquetes útiles necesarios, como:

```
gnome-boxes virt-manager virt-goodies virt-sandbox virt-top
virt-viewer virtinst libvirt-clients libvirt-daemon libvirt-daemon-
system qemu qemu-kvm qemu-utils qemu-system qemu-system-
gui qemu-block-extra freerdp2-x11 bridge-utils ovirt-guest-agent
systemd-container
```

Otros en caso de querer instalar otras tecnologías de virtualización disponibles sobre Linux/BSD se puede optar por:

Xen instalándolo con la orden de comando siguiente:

```
# apt install xen-system-amd64 xen-utils-4.11 xen-tools
```

LXC instalándolo con la orden de comando siguiente:

```
# apt install lxc
```

Docker instalándolo con la orden de comando siguiente:

```
# apt install docker-ce docker-ce-cli containerd.io
```

Diferencias entre KVM y QEMU Cuando comenzamos en el mundo de la virtualización, la opción más recurrente para comenzar es KVM. Después nos damos cuenta de que se comienza a usar también otro componente muy a menudo, QEMU y siempre hay muchas preguntas en torno a cómo funciona KVM y QEMU o cual es la diferencia entre ellos.

Sobre KVM es un Software de código abierto y significa Kernel Virtual Machine (Máquina Virtual basada en el Kernel) es una solución de virtua-lización para Linux en hardware x86 que contiene extensiones de virtualización (Intel VT o AMD-V). KVM forma parte del Kernel de Linux desde la versión 2.6.20. Específicamente, con KVM podemos convertir a Linux en un hipervisor para que nuestro Host ejecute entornos virtuales, es decir, máquinas virtuales. Cada máquina virtual se implementa como un proceso regular de Linux

KVM ha jugado un papel clave en el entorno de virtualización de código abierto basado en Linux. De hecho, KVM es el único hipervisor para todos los productos de virtualización de Red Hat, tanto para RHOSP - Red Hat Openstack Platform, como para Red Hat Virtualization o abreviado, RHV.

KVM en general es 2 cosas, un módulo del Kernel pero también KVM es es un Fork del ejecutable de QEMU (más adelante hablaremos de eso). Entonces como mencionamos, KVM es un módulo Kernel que permite el uso de tecnologías de extensiones de virtualización Intel o AMD. En pocas palabras, estas extensiones permiten que múltiples sistemas operativos compartan una CPU física sin interferir entre ellos. Por otro lado, no resuelven compartir todos los dispositivos de Hardware, para esto KVM requiere una lógica extra y aquí es dónde comenzamos a hablar de QEMU.

Sobre QEMU es un emulador de procesadores basado en la traducción dinámica de binarios, es decir, realiza la conversión del código binario de la arquitectura fuente o Host, en código entendible por la arquitectura huésped o la máquina virtual.

El resultado de usar QEMU es poder ejecutar el código original como si se estuviera ejecutando en la máquina emulada. Por ejemplo, se podría ejecutar código escrito para el procesador ARM en su máquina basada en Intel.

QEMU es capaz de emular varias plataformas de Hardware diferentes, incluida la x86, plataformas PowerPC, sistemas basados en ARM y también sistemas SUN SPARC. Además del Hardware básico de estos sistemas, QEMU también proporciona emulación de varios módulos adicionales, como tarjetas gráficas, tarjetas de sonido, dispositivos de red, dispositivos de almacenamiento y controladores, dispositivos serie/paralelo/USB y dispositivos de memoria. Esto significa que en muchos casos las computadoras pueden ser completa y totalmente emuladas y utilizadas para ejecutar su Software original.

¿Cómo trabajan juntos? en el Hardware real, el sistema operativo traduce las instrucciones de los programas para que sean ejecutadas por el CPU físico. En una máquina virtual es lo mismo, pero la diferencia es que el CPU está virtualizado por el hipervisor y el hipervisor tiene que traducir las instrucciones del CPU virtual y convertirlo en instrucciones para el CPU físico. Esta traducción tiene una gran sobrecarga de rendimiento.

Para minimizar esta sobrecarga, los procesadores admiten extensiones de virtualización. Intel soporta una tecnología llamada VT-X y el equivalente AMD es AMD-V. Con el uso de estos, una rebanada de CPU físico se puede asignar directamente al CPU virtual. Así que las instrucciones de la CPU virtual se pueden ejecutar directamente en la rebanada del CPU físico. Evitando así la traducción que tendría que hacer el hipervisor.

KVM es el módulo del Kernel de Linux que permite esta asignación de CPU físico para CPU virtual. Esta asignación proporciona la aceleración de Hardware para la máquina virtual y aumenta su rendimiento. De hecho QEMU utiliza esta aceleración cuando el tipo de virtualización es elegido como KVM.

Los desarrolladores de KVM aprovecharon la arquitectura QEMU y básicamente crearon un nuevo modelo de CPU en QEMU. Este nuevo tipo de modelo tiene una lógica específica de KVM. Así las llamadas al sistema que

se harían de forma nativa pasan por del módulo KVM para que la ejecución se ejecute de forma nativa en la CPU, mientras que QEMU se utiliza para proporcionar el resto funcionalidad (emular los dispositivos).

Al trabajar juntos, KVM accede directamente al CPU físico y a la memoria, a su vez QEMU emula los recursos de Hardware, como el disco duro, video, USB, etc. Hoy, cuando las personas se refieren al hipervisor KVM, en realidad se refieren a la combinación QEMU/KVM.

KVM necesita QEMU (emulador) para la funcionalidad completa como hipervisor. QEMU es autosuficiente y KVM es realmente un módulo del Kernel de Linux para la explotación de extensiones VT que actúa como controlador de las capacidades de CPU físicas.

Así puedes notar entonces que QEMU necesita a KVM para aumentar su rendimiento... Por otro, lado KVM por sí solo no puede proporcionar la solución de virtualización completa, necesita de QEMU.

10.10 Acceso a Datos Desde una Máquina Virtual

Para acceder a los datos almacenados en máquinas virtuales, disponemos de las siguientes opciones:

a) Mediante el uso de algún navegador Web, se puede acceder a su cuenta de correo electrónico y al almacenamiento en la nube como *Google Drive, Dropbox, HubiC, pCloud, MediaFire, FlipDrive, Mega*, entre otros.

b) En el sistema operativo Linux, se puede acceder a cualquier servidor de internet mediante los protocolos *SSH*, *SAMBA* o montar un sistema de archivos mediante *NFS* o *SSHFS*, entre otros.

c) En cualquier sistema operativo podemos usar algún navegador gráfico de *FTP, FTPS o SFTP* como *FileZilla, WinSCP, PSCP, PSFTP, FireFTP, CoreFTP*, entre muchos otros, para transportar archivos y carpetas.

d) En las máquinas virtuales de Windows usamos el protocolo *SAMBA*, para tener acceso a este, hay que conectarse a una unidad de red dentro del explorador de archivos de Windows.

e) En Linux, por ejemplo con *PCManFM, Dolphin, Nautilus, Thunar, Konqueror*, entre otros, podemos acceder a una máquina que tenga un servidor de la siguiente forma:

1) Para acceder a un servidor *SAMBA*, escribir la ruta de archivos en el manejador de archivos:

$ smb://estud@192.168.13.230/estud/

2) Para acceder a un servidor *SSH*, escribir la ruta de archivos en el manejador de archivos:

$ sftp://usuario@192.168.13.230/home/usuario/

En línea de comandos, podemos:

3) Montar con *SSHFS* un directorio de otra máquina con servidor *SSH*:

$ sshfs usuario@192.168.13.230:/home/usuario/ /home/algun/lugar

4) Montar con mount un directorio de otra máquina con servidor *NFS*:

mount 10.0.2.2:/directorio ./punto_montaje

5) Usar *SCP* y *SFTP* de *SSH* para transferir archivos, para copiar un archivo, usamos:

$ scp archivo.dat usuario@192.168.13.230:~/Datos/

para copiar un subdirectorio, usamos:

$ scp -r Directorio usuario@192.168.13.230:.

para copiar un archivo de una máquina remota a nuestra máquina, usamos:

$ scp usuario@192.168.13.230:/home/usuario/archivo .

6) Montar con CURLFTPFS un directorio de otra máquina con servidor *FTP*:

curlftpfs -o allow_other usuario:password@192.168.13.230:puerto /home/algun/lugar

10.11 Desde la Nube

Existen diferentes servicios Web[195] que permiten instalar, configurar y usar cientos de sistemas operativos Linux y Unix -máquinas virtuales usando servicios Web en Debian GNU/Linux y QEMU- desde el navegador, esto en

[195] Cuando se trabaja desde la Web es recomendable usar el modo Privado o Incógnito para no guardar el historial de navegación, información introducida en los formularios y borrar al cerrar el navegador los datos de los sitios visitados. Pero recuerda que los sitios Web que visitamos sí guardan información de nuestra visita, nuestro proveedor de internet también guarda constancia de nuestra visita y si descargamos algo, esto no se borra al igual que el historial de descargas, además de las marcas de páginas o favoritos se conservarán al cerrar el navegador.

aras de que los usuarios que cuenten con algún sistema de acceso a red y un navegador puedan usar, configurar e instalar algún sistema operativo y su respectiva paquetería sin hacer instalación alguna en su equipo de cómputo, tableta o teléfono celular[196].

Una muestra de estos proyectos son: Distrotest (https://distrotest.net) y JSLinux (https://bellard.org/jslinux).

Algunas versiones listas para usar son:

4mLinux, AbsoluteLinux, Academix, AlpineLinux, Antergos, antiX Linux, Aptosid, ArchBang, ArchLabs, ArchLinux, Archman, ArchStrike, ArcoLinux, ArtixLinux, AryaLinux, AV Linux, BackBoxLinux, BigLinux, Bio-Linux, BlackArch, BlackLab, BlackPantherOS, BlackSlash, blag, BlankOn, Bluestar, Bodhi, BunsenLabs, ByzantineOS, Caine, Calculate Linux Desktop, CentOS, Chakra, ChaletOS, ClearOS, Clonezilla, ConnochaetOS, Cucumber, Damn Small Linux, Damn Small Linux Not, Debian, DebianEdu, deepin, DEFT, Devil-Linux, Devuan, DragonFly BSD, Dragora, DuZeru, Dyne:bolic, Edubuntu, elementaryOS, Elive Linux, Emmabuntüs, Emmi OS, Endless OS, EnsoOS, Exe GNU/Linux, ExTiX, Fatdog64, Fedora Atomic, Fedora Server, Fedora Workstation, FerenOS, FreeBSD, FreeDOS, Frugalware, G4L, GeckoLinux, Gentoo, GNewSense, GoboLinux, Gparted, GreenieLinux, GRML, GuixSD, Haiku, Heads, Kali Linux, Kanotix, KaOS, Knoppix, Kodachi, KolibriOS, Korora, Kubuntu, Kwort, Linux Lite, Linux Mint, LiveRaizo, LMDE, Lubuntu, LXLE OS, Macpup, Mageia, MakuluLinux, Manjaro, Matriux, MauiLinux, MenuetOS, MinerOS, MiniNo, Modicia, Musix, MX Linux, Nas4Free, Neptune, NetBSD, Netrunner, NixOs, NST, NuTyX, OpenIndiana, OpenMandriva, openSUSE, OracleLinux, OSGeo live, OviOS, Parabola CLI, Parabola LXDE, Pardus, Parrot Home, Parrot Security, Parrot Studio, Parsix, PCLinuxOS, PeachOSI, Pentoo, Peppermint, PeppermintOS, Pinguy, PinguyOS, plopLinux, PointLinux, Pop!_OS, PORTEUS, Puppy Linux, PureOS, Q4OS, QubesOS, Quirky, Raspberry Pi Desktop, ReactOS, Redcore, Rescatux, RevengeOS, RoboLinux, Rockstor, ROSA FRESH, Runtu, Sabayon, SalentOS, Salix, ScientificLinux, Siduction, Slackware, Slax, SliTaz, Solus, SolydK, SolydX, SparkyLinux, Springdale, StressLinux, SubgraphOS, SwagArch, Tails, Tanglu, Tiny Core, Trisquel, TrueOS, TurnKey Linux, Ubuntu, Ubuntu Budgie, Ubuntu Studio, UbuntuKylin, Uruk, VectorLinux, VineLinux, VoidLinux, Voyager, VyOS, WattOs, Xubuntu, Zentyal, Zenwalk, Zevenet, Zorin OS

Usar Linux en Formato Live Linux es uno de los sistemas operativos pioneros en ejecutar de forma autónoma o sin instalar en la computadora, existen diferentes distribuciones Live -descargables para formato CD, DVD, USB[197]- de sistemas operativos y múltiples aplicaciones almacenados en un medio extraíble, que pueden ejecutarse directamente en una computadora,

[196] Estos servicios son conocidos como computación en la nube (Cloud Computing).

[197] Para generar un dispositivo USB con la imagen contenida en un archivo ISO podemos usar el Software ETCHER, descargable para Linux, Windows y Mac OS desde https://etcher.io/.

estos se descargan de la Web generalmente en formato ISO[198], una de las listas más completas de versiones Live está en:

https://livecdlist.com

En el caso de tener un archivo ISO de algún sistema operativo (por ejemplo ubuntu-11.10-desktop-i386.iso) y se quiere ejecutar su contenido desde una máquina virtual con QEMU/KVM sólo es necesario usar:

```
$ kvm -m 512 -cdrom ubuntu-11.10-desktop-i386.iso
```

en este ejemplo usamos en KVM la arquitectura por omisión y memoria de 512 MB (-m 512).

Knoppix es una versión Live ampliamente conocida y completa, esta se puede descargar de:

https://www.knopper.net/knoppix/

y usar mediante:

```
$ kvm -m 1024 -cdrom KNOPPIX_V8.2-2018-05-10-EN.iso
```

aquí se usa la arquitectura por omisión y memoria de 1024 MB.

Descarga de Máquinas Virtuales de Sistemas Operativos Existen diversos proyectos que permiten descargar decenas de máquinas virtuales listas para ser usadas, para los proyectos VirtualBox y VMWare (y por ende para KVM/QEMU), estas se pueden descargar de múltiples ligas, algunas de ellas son:

https://www.osboxes.org
https://www.virtualbox.org

Si desargamos y descomprimimos el archivo lubuntu1210.7z, esto dejará la imagen de VirtualBox de LUBUNTU cuyo nombre es lubuntu1210.vdi. Entonces esta imagen la usaremos directamente en KVM/QEMU, mediante:

```
$ kvm -m 2000 -hda lubuntu1210.vdi
```

Nota: esta imagen usa como usuario y clave de acceso: lubuntu/lubuntu

[198] Una imagen ISO es un archivo informático donde se almacena una copia exacta de un sistema de archivos y de esta se puede generar una imagen para CDROM, DVD o USB.

11 Creación, Uso y Optimización de Máquinas Virtuales Usando QEMU/KVM

Entendamos por una máquina virtual (véase sección 10) a un programa de cómputo que simula a una computadora, en la cual se puede instalar y usar otros sistemas operativos de forma simultánea como si fuese una computadora real sobre nuestro sistema operativo huésped. Hoy en día, tenemos a nuestra disposición varios manejadores de máquinas virtuales (MV), algunos de ellos son los siguientes:

- Virtualbox (véase [32]) es un Software desarrollado por Oracle, se trata de un Software multiplataforma capaz de virtualizar prácticamente la totalidad de sistemas operativos con arquitectura x86/AMD64. La base de este Software dispone de una licencia GPL2 (véase 14.4), mientras que el Pack de extensiones que añaden funcionalidades están bajo licencia privativa. Virtualbox es gratuito para un uso no comercial.

- Vmware Workstation Player (véase [33]) es un Software privativo multiplataforma desarrollado por EMC corporation y es ampliamente usado en el entorno profesional en las áreas del Cloud Computing entre muchas otras. Al igual que Virtualbox, esta máquina virtual nos permite virtualizar una gran diversidad de sistemas operativos. Vmware dispone de muchas soluciones de virtualización y prácticamente todas son de pago, no obstante Vmware Workstation Player es gratuita para un uso no comercial.

- Parallels (véase [36]) es un Software multiplataforma, es usado frecuentemente por los usuarios del sistema operativo OS X de Apple que desean virtualizar el sistema operativo Windows. Esta máquina virtual es de pago y únicamente puede virtualizar los sistemas operativos Windows y Mac OS.

- Windows Virtual PC (véase [34]) es un Software gratuito y privativo propiedad de Microsoft que se puede usar tanto en Windows como en Mac OS. Virtual PC está destinado únicamente a virtualizar sistemas operativos Windows.

- QEMU/KVM (véase [29]) es un Software libre multiplataforma que dispone de licencia GPL (véase 14.4). Además de virtualizar un gran

número de sistemas operativos, permite emular diversas arquitecturas como por ejemplo X86, x86-AMD64, MIPS, Arm, PowerPC, etc.

¿Qué Manejadores Libres de Máquinas Virtuales Podemos Instalar?

QEMU Es un emulador de procesadores basado en la traducción dinámica de binarios -conversión del código binario de la arquitectura fuente en código entendible por la arquitectura huésped-. QEMU también tiene capacidades de virtualización dentro de un sistema operativo, ya sea GNU/Linux, Windows, o cualquiera de los sistemas operativos admitidos; de hecho es la forma más común de uso. Esta máquina virtual puede ejecutarse en cualquier tipo de Microprocesador o arquitectura (x86, x86-64, PowerPC, MIPS, SPARC, etc.). está licenciado en parte con la LGPL y la GPL de GNU (véase 14.4).

El objetivo principal es emular un sistema operativo dentro de otro, sin tener que reparticionar el disco duro, empleando para su ubicación cualquier directorio dentro de éste. El programa no dispone de GUI[199], pero existe otro programa llamado QEMU Manager que puede hacer de interfaz gráfica si se utiliza QEMU desde Windows. También existe una versión para GNU/Linux llamada Qemu-Launcher. En Mac OS X puede utilizarse el programa Q que dispone de una interfaz gráfica para crear y administrar las máquinas virtuales.

Kernel-based Virtual Machine (KVM) Máquina virtual basada en el núcleo es una solución para implementar virtualización completa con Linux, está formada por un módulo del núcleo (con el nombre kvm.ko) y herramientas en el espacio de usuario, siendo en su totalidad Software libre (véase 14.4). El componente KVM para el núcleo está incluido en Linux desde la versión 2.6.20. El proyecto KVM está incluido en el proyecto QEMU.

Las características principales de KVM/QEMU son:

- Cada máquina virtual se implementa como un proceso
- KVM/QEMU aprovecha el modelo de seguridad estándar de Linux: SELinux/AppArmor[200]. Estos modelos proporcionan el aislamiento y

[199] Graphical User Interface (Interfaz gráfica del usuario).
[200] AppArmor fue creado en parte como alternativa a SELinux.

el control de recursos necesarios

- Hereda las características de gestión de memoria de Linux. La memoria utilizada por una MV se gestionará de la misma forma que la de otro proceso, podrá ser guardada en disco, utilizada en páginas grandes y soporte NUMA[201] de Linux permitiendo el uso de MVs de grandes cantidades de memoria

- Soporta las últimas características de virtualización de memoria proporcionada por los fabricantes como EPT (Extended Page Table de Intel) ó RVI (Rapid Virtualization Indexing de AMD). Estas tecnologías persiguen reducir el uso de CPU y aumentar el rendimiento de los Hipervisores

- El compartir páginas de memoria se consigue a través de la característica añadida a Linux llamada Kernel Same-page Merging (KSM). Escaneando las páginas de memoria de cada MV, si dos páginas coinciden, KSM las une en una sola página que se comparte entre las dos máquinas, almacenando únicamente una copia y si en cualquier momento, una de las MV modifica la página, le da una copia privada

- Permite utilizar cualquier tipo de almacenamiento soportado por Linux para las imágenes de las MVs

- Soporta el almacenamiento de ficheros distribuidos como GFS2[202], OCFS[203] o GlusterFS[204]. De esta forma las imágenes de las MV pueden ser compartidas por varios Hipervisores

- Las imágenes de disco soportan aprovisionamiento bajo demanda evitando tener que reservar todo el espacio inicialmente asignado. El formato nativo de KVM es QCOW2[205], el cual permite la realización de

[201] Non-Uniform Memory Access (acceso a memoria no uniforme).
[202] Global File System 2 es un sistema de archivos compartido para clusters en Linux.
[203] Oracle Cluster File System es un sistema de archivos de discos compartidos o sistema de archivos distribuidos para clústers de servidores de sistemas GNU/Linux desarrollado por Oracle Corporation distribuidos bajo los términos de la GNU General Public License.
[204] Gluster File System es un sistema multiescalable de archivos NAS desarrollado inicialmente por Gluster Inc.
[205] QEMU Copy-On-Write el formato de imagen para máquinas virtuales segunda versión de QCOW.

Snapshots[206], compresión y cifrado

- Permite migraciones en vivo (Live Migrations), estas características permiten mover una MV en ejecución entre servidores físicos (Hipervisores) sin interrupción del servicio. Estas migraciones son transparentes para el usuario, ya que la MV permanece encendida, las conexiones de red activas y las aplicaciones en ejecución mientras la máquina se reacomoda en un nuevo servidor
- KVM/QEMU soporta MV de hasta 255 CPUs y 4 TB de RAM. Y el rendimiento de aplicaciones como Oracle, SAP, LAMP, MS Exchange sobre MV puede oscilar entre el 95% y el 135% comparado con su ejecución en servidores físicos, se ha conseguido ejecutar hasta 600 máquinas virtuales en un sólo servidor físico
- Soporte de sistemas operativos invitados como Windows, Linux, Android, Familia BDS (OpenBSD, FreeBSD, NetBSD), Solaris, etc.
- Es ampliamente usado en varios proyectos sobre Cloud Computing como OpenStack, CloudStack, OpenNebula, etc.

En esta sección mostraremos cómo crear, configurar, optimizar y trabajar con las máquinas virtuales mediante KVM/QEMU en Debian GNU/Linux para probar imágenes ISO[207] descargadas de la red, instalar y usar máquinas virtuales para Windows y Linux entre otros.

11.1 Tipo de Virtualización Soportado por la Máquina Huésped

Primero es necesario saber si nuestro equipo soporta la virtualización por Hardware o debemos usar la virtualización por Software, suponiendo que tenemos acceso a una máquina con Linux o ha sido inicializada usando una versión «viva (Live)»[208] de CD, DVD o USB de Linux para iniciar la computadora.

[206] Es una copia instantánea del sistema de archivos que contiene a la máquina virtual.
[207] Una imagen ISO es un archivo informático donde se almacena una copia exacta de un sistema de archivos y de esta se puede generar una imagen para CDROM, DVD o USB.
[208] Una opción es KNOPPIX, es una distribución de Linux basada en Debian y usa LXDE como entorno de escritorio, pude ser descargada de http://www.knopper.net/knoppix/

Entonces, para revisar si hay soporte en Hardware para la virtualización, podemos usar cualquiera de estas cuatro opciones:

1) Buscar información en: */proc/cpuinfo*, mediante:

```
$ egrep "vmx|svm" /proc/cpuinfo
```

si se soporta la virtualización por Hardware aparecerá -entre otras[209]- la bandera

```
Procesadores INTEL: vmx
Procesadores AMD: svm
```

2) Podemos usar *lscpu* que nos indicará las características de nuestro equipo, en particular en el apartado Virtualization nos indicará si es soportada o no, para ello usamos:

```
$ lscpu | grep Virtualization
```

3) Podemos usar cpu-checker, instalando:

```
# apt install cpu-checker
```

y para conocer si es soportado, usamos:

```
$ kvm-ok
```

4) Podemos usar libvirt-client, instalando:

```
# apt installlibvirt-clients
```

y para conocer si es soportado, usamos:

```
$ virt-Host-validate
```

[209] El significado de las banderas de /proc/cpuinfo esta descrita en la sección 11.14.

Instalar y Usar Máquinas Virtuales Por omisión, los equipos de tecnología de bajo desempeño no soporta la virtualización a nivel Hardware, pero siempre es posible su emulación mediante QEMU.

Si la computadora soporta virtualización a nivel Hardware es posible usar KVM (o en QEMU la bandera -enable-kvm). Según la versión de Linux, KVM puede existir como un paquete real o como uno virtual, si es virtual, al instalar KVM lo que realmente se instala es QEMU y al ejecutar KVM por ejemplo

```
$ kvm ...
```

es reemplazado por

```
$ qemu-system-x86_64 -enable-kvm ...
```

Estos tienen la misma sintaxis de uso, y para nuestros ejemplos sólo es necesario reemplazar qemu-system-x86_64 por kvm y en ambos siempre se usará qemu-img para manipular las imágenes.

Instalación de KVM en Debian GNU/Linux (recomendado para virtualización por Hardware) es mediante:

```
# apt install qemu-kvm
```

Instalación de QEMU en Debian GNU/Linux (permite emular diversas arquitecturas de Hardware) es mediante:

```
# apt install qemu
```

Observación 6 *El desempeño de la emulación versus virtualización por Hardware es de varios órdenes de magnitud menor, pero una imagen creada con cualquiera de ellos se puede usar con los otros virtualizadores. KVM sólo soporta virtualizar arquitecturas X86 y 64 de INTEL y AMD, QEMU emula diversas arquitecturas, como son ARM, CRIS, i386, M68k, MicroBlaze, MIPS, PowerPC, SH4, SPARC y x86-64.*

Problemas Comunes al Virtualizar Si se detecta la bandera para virtualización por Hardware y al tratar de usar KVM marca:

> open /dev/kvm: Permission denied
> Could not initialize KVM, will disable KVM support

sólo hay que agregar, el login del usuario al grupo *kvm* en el archivo */etc/group*.

Si marca:

> open /dev/kvm: No such file or directory
> Could not initialize KVM, will disable KVM support

sólo hay que activar en el BiOS la virtualización por Hardware

En KVM, al usar un procesador y solicitar la emulación de otro, es común que marque que ciertas banderas no son soportadas, por ejemplo al usar un procesador AMD y solicitar la emulación de un procesador Nehalem Intel Core i7 9xx (Nehalem Class Core i7) mediante:

```
$ kvm -cpu Nehalem -cdrom TinyCore-current.iso[210]
```

ó

```
$ qemu-system-x86_64 -enable-kvm -cpu Nehalem -cdrom \
TinyCore-current.iso
```

es común que marque:

warning: Host doesn't support requested feature:

CPUID.01H:ECX.ssse3 [bit 9]

warning: Host doesn't support requested feature:

CPUID.01H:ECX.sse4.1.sse4_1 [bit 19]

[210] TinyCoreLinux es un sistema operativo minimalista centrado en proveer un sistema base con núcleo Linux —es de tamaño de 11,16 MB y 106 Mb—, puede ser descargado de https://distro.ibiblio.org/tinyCorelinux

warning: Host doesn't support requested feature:

CPUID.01H:ECX.sse4.2.sse4_2 [bit 20]

si es necesario usar dichas banderas en el CPU, entonces usar:

```
$ qemu-system-x86_64 -cpu Nehalem -cdrom TinyCore-current.iso
```

en este caso avisará que:

warning: TCG doesn't support requested feature:

CPUID.01H:EDX.vme [bit 1]

i.e. soporta el chip, pero no la virtualización (vme: Virtual Mode Extensions [8086 mode]), se sacrifica velocidad en aras de tener las prestaciones del chip emulado.

11.2 Salida Gráfica de la Virtualización Usando VNC

Si se usa el protocolo de Computación Virtual en Red (Virtual Network Computing VNC[211]) como visualizador de la salida gráfica de KVM/QEMU, debemos agregar *-vnc :n a la línea de comandos*, donde *n* es el número de pantalla a usar, esto se hace mediante:

```
$ kvm -m 128 -cdrom TinyCore-current.iso -cpu kvm64 \
-vnc :0 &
```

[211] Virtual Network Computing (VNC) es un programa de Software libre basado en una estructura cliente-servidor que permite observar las acciones del ordenador servidor remotamente a través de un ordenador cliente. VNC no impone restricciones en el sistema operativo del ordenador servidor con respecto al cliente, es posible compartir la pantalla de una máquina con cualquier sistema operativo que admita VNC conectándose desde otro ordenador o dispositivo que disponga de un cliente VNC portado, además permite usar Internet de baja velocidad en la visualización.

Para visualizar la salida gráfica usando VNC se debe instalar algún cliente de VNC, en Debian Linux existen una gran variedad de clientes, uno de ellos es tigervnc-viewer, y se instala mediante:

```
# apt install tigervnc-viewer
```

Otras opciones son *vncviewer* y *xtightvncviewer*. Ninguno de ellos requiere configuración adicional al ser instalados.

y para ver la salida gráfica[212] en la misma máquina, usamos:

```
$ vncviewer 0
```

Si se desea ver la salida gráfica en otro equipo conectado en red (puede ser con sistema operativo Windows, Linux o MAC OS que tenga instalado *vncviewer*[213]), es recomendable hacer ajustes en la calidad de la salida gráfica a mostrar y que no se vea afectada por la velocidad del internet, si suponemos que el servidor de la máquina virtual es *192.168.13.230*, entonces lanzamos la máquina virtual mediante:

```
$ kvm -m 128 -cdrom TinyCore-current.iso -cpu kvm64 \
-vnc :0,lossy &
```

y para ver la salida gráfica en cualquier otro equipo interconectado por red, usamos:

```
$ vncviewer 192.168.13.230:0 QualityLevel=3
```

donde la calidad del video *QualityLevel=n*[214], es de 0 a 9, donde 0 es la más pobre y 9 la más alta calidad de la salida gráfica.

Nota: En caso que el cursor del Mouse de la máquina virtual no coincida con el del equipo anfitrión es necesario agregar: *-usb -device usb-tablet*, al lanzar la máquina virtual:

```
$ kvm -m 128 -cdrom TinyCore-current.iso -cpu kvm64 \
-usb -device usb-tablet -vnc :0 &
```

y para ver la salida gráfica (como se comentó antes) usamos:

```
$ vncviewer 0
```

[212] Después de que es ejecutado el comando *vncviewer*, aparecerá la ventana de la máquina virtual (optimizada para ser usada en conexiones de red de baja velocidad), en ella se puede usar la máquina virtual como si estuviese instalada en su equipo. Se puede cerrar la ventana de visualización de VNC y la máquina virtual seguirá trabajando en el servidor; de ser necesario se puede abrir el cliente de VNC tantas veces como requiera. Para apagar la máquina virtual en el servidor, se debe de solicitar a esta que se apague mediante el menú de inicio de la virtualización.

[213] Otros proyectos multiplataforma son: Zoho Assist, TigerVNC, RealVNC, TeamViewer, Remmina, NoMachine, Apache Guacamole, XRDP, FreeNX, X2Go, Xpra, entre otros.

[214] El valor por omisión es de 3 para una para una conexión de Internet de baja velocidad común en los hogares, en caso necesario usar un valor de 0, que permite usar VNC en conexiones de muy baja velocidad.

Uso de SSH para Interactuar con una Máquina Virtual de Forma Remota Si se tiene acceso a un servidor mediante SSH [215] en el cual esté activo X11 Forwarding[216] e instalado KVM/QEMU, entonces es posible ejecutar una máquina remota en el servidor y visualizar la salida gráfica en la máquina donde se ejecuta el comando SSH[217].

Primero, al hacer la conexión mediante SSH, es necesario solicitar en la conexión se habilite X11 Forwarding mediante:

```
$ ssh -X -l usr 192.168.13.230
```

donde <usr> es el nombre del usuario en el equipo <192.168.13.230>. Después de hacer la conexión, ya podemos ejecutar la máquina virtual como se indicó antes:

```
$ kvm -m 128 -cdrom TinyCore-current.iso &
```

y la salida gráfica de la máquina virtual se trasmitirá por red de forma segura usando la tunelización de SSH.

11.3 Usando un Sistema Operativo Live como una Máquina Virtual

Linux es uno de los sistemas operativos pioneros en ejecutar de forma autónoma o sin instalar en la computadora, existen diferentes distribuciones Live - descargables para formato CD, DVD, USB[218]- de sistemas operativos y múltiples aplicaciones almacenados en un medio extraíble, que pueden ejecutarse

[215] SSH (o Secure SHell) es un protocolo que facilita las comunicaciones seguras entre dos sistemas usando una arquitectura cliente/servidor y que permite a los usuarios conectarse a un host remotamente. A diferencia de otros protocolos de comunicación remota tales como FTP o Telnet, SSH encripta la sesión de conexión, haciendo imposible que alguien pueda obtener contraseñas no encriptadas.

[216] Es el servidor gráfico que usan casi todas las distribuciones Linux. Este servidor permite, entre otras cosas, Forwarding a través de SSH. Esto significa que es posible ejecutar aplicaciones gráficas de una máquina remota exportando el Display a nuestro escritorio. Es decir, la aplicación se ejecuta en el servidor remoto, pero la interfaz gráfica la visualizamos en nuestro escritorio local.

[217] Es recomendable usar VNC en conjunción con SSH, en lugar de SSH puro, ya que el consumo de red en la salida gráfica sin VNC en la conexión SSH es excesivo para la mayoría de las infraestructuras de Internet.

[218] Para generar un dispositivo USB con la imagen contenida en un archivo ISO podemos usar el Software ETCHER, descargable para Linux, Windows y Mac OS desde https://etcher.io/

directamente en una computadora, estos se descargan de la Web generalmente en formato ISO[219], una de las listas más completas de versiones Live esta en https://livecdlist.com

En el caso de tener un CD, DVD o USB Live y se quiera ejecutar su contenido desde una máquina virtual con QEMU/KVM solo es necesario montar el dispositivo. Para ello, primero es necesario conocer donde es montado por el sistema operativo, mediante:

```
$ df
```

suponiendo que el dispositivo es /dev/sddx, entonces usar ese dispositivo en KVM mediante:

```
$ kvm -m 512 -usb /dev/sddx
```

en este ejemplo usamos el virtualizador con la arquitectura por omisión y memoria de 512 MB (-m 512).

11.4 Usando un Archivo ISO como una Máquina Virtual

En el caso de tener un archivo ISO[220] de algún sistema operativo (ubuntu-11.10-desktop-i386.iso) y se quiera ejecutar su contenido desde una máquina virtual con QEMU/KVM solo es necesario usar:

```
$ kvm -m 512 -cdrom ubuntu-11.10-desktop-i386.iso
```

en este ejemplo usamos al virtualizador con la arquitectura por omisión y memoria de 512 MB (-m 512).

[219] Una imagen ISO es un archivo informático donde se almacena una copia exacta de un sistema de archivos y de esta se puede generar una imagen para CDROM, DVD o USB.

[220] Una imagen ISO es un archivo informático donde se almacena una copia exacta de un sistema de archivos y de esta se puede generar una imagen para CDROM, DVD o USB.

11.5 Creación de Máquinas Virtuales

En esta sección mostraremos varios ejemplos completos para crear máquinas virtuales de Linux y Windows mediante el uso de KVM/QEMU.

Ejemplo 4 *Instalación y uso de una máquina virtual para Debian GNU/Linux estable a partir del archivo ISO*[221]*, para ello, primero necesitamos:*

Generar el disco virtual que la contendrá, por ejemplo de 10 GB con el nombre debianStable.img mediante:

```
$ qemu-img create -f qcow2 debianStable.img 10G
```

Después, instalar la imagen de Debian estable en el disco virtual genera-do en el paso anterior, solicitando a KVM que una vez terminada la instalación no haga el reinicio de la máquina virtual, esto mediante:

```
$ kvm -no-reboot -boot d -cdrom debian-802-i386-netinst.iso \
-hda debianStable.img -m 400
```

Después de la instalación, es conveniente compactar y desfragmentar la imagen usando:

```
$ qemu-img convert -c debianStable.img -O qcow2 debian.img
```

Ahora podemos usar la máquina virtual con la imagen desfragmentada y compactada de Debian estable solicitando que tenga 800 MB de RAM, mediante:

```
$ kvm -hda debian.img -m 800
```

Observación 7 *La definición de la toda la máquina virtual -el disco virtual que contiene el sistema operativo instalado y su respectiva paquetería- está contenida en un único archivo que puede ser copiado, almacenado o distribuido. En esto radica el poder de las máquinas virtuales, una vez hecha y configurada, se puede usar en donde se requiera y la cantidad de veces que lo necesitemos.*

Al ser un solo archivo la máquina virtual, es común tener múltiples archivos que conserven los distintos estados conforme se instalen o configuren paquetes de la misma. De esta forma se agiliza la recuperación tras algún fallo y el poder hacer modificaciones de la máquina base o restaurar una máquina a algún punto de configuración anterior, con tan solo usar la respectiva copia almacenada.

[221] Diversas imágenes ISO del proyecto Linux Debian se pueden descargar de: https://www.debian.org/CD/

Además, al usar KVM/QEMU se tiene la certeza de que la máquina virtual creada en una distribución de Linux puede ser usada en cualquier otra distribución bajo cualquier arquitectura de Hardware soportada por Linux -que tenga instalada una versión igual o superior de KVM/QEMU- sin cambio alguno.

Ejemplo 5 *Instalación y uso de una máquina virtual para Windows XP, en este caso necesitamos:*

Crear el disco virtual, por ejemplo de 10 GB mediante:

```
$ qemu-img create -f qcow2 WindowsXP.img 10G
```

Hacer la instalación básica de Windows XP a partir, por ejemplo del ISO, mediante:

```
$ kvm -no-reboot -boot d -hda WindowsXP.img -m 400 \
-cdrom es_winxp_pro_with_sp2.iso
```

Y concluir la instalación de Windows XP mediante:

```
$ kvm -no-reboot -boot c -hda WindowsXP.img -m 400 \
-cdrom es_winxp_pro_with_sp2.iso
```

Después de la instalación, es conveniente compactar y desfragmentar la imagen usando:

```
$ qemu-img convert -c WindowsXP.img -O qcow2 Windows.img
```

Ahora podemos usar la máquina virtual con la imagen desfragmentada y compactada de Windows XP usando:

```
$ kvm -boot c -hda Windows.img -m 400
```

Ejemplo 6 *Una vez que se cuenta con una imagen de Windows, podemos instalar por ejemplo Windows Office, donde tenemos dos opciones a saber:*

1) Instalar Windows Office 2003 a partir del ISO de Office mediante:

```
$ kvm -m 300 -boot c -hda Windows.img \
-cdrom Office-2003.iso
```

2) Si se tiene el CD o DVD, entonces podemos usar:

```
$ kvm -m 300 -boot c -hda Windows.img \
-cdrom /dev/cdrom/
```

Observación 8 *En el caso de Windows hay que usar el mismo Hardware siempre, en caso contrario marca que es necesario registrar el sistema operativo nuevamente al ejecutarlo en otra arquitectura, para evitar este problema, es necesario usar la bandera -cpu al momento de crearlo y usarlo, por ejemplo:*

Usar máquina virtual de Windows en QEMU y KVM usando el mismo Hardware mediante alguna de estas opciones:

```
$ kvm -m 400 -boot c -hda Windows.img -cpu qemu32
$ qemu-system-x86_64 -m 400 -boot c -hda \
Windows.img -cpu qemu32
$ qemu-system-x86_64 -enable-kvm -m 400 -boot c \
-hda Windows.img -cpu qemu32
```

Para conocer los CPUs soportados usar:

```
$ kvm -cpu ?
```

Para conocer las máquinas soportadas usar:

```
$ kvm -machines ?
```

Ejemplo 7 *Otro ejemplo completo de instalación y uso de una máquina virtual para Windows 7, en este caso necesitamos:*

Crear el disco virtual, por ejemplo de 15 GB mediante:

```
$ qemu-img create -f qcow2 Windows7.img 15G
```

Hacer la instalación básica de Windows 7 a partir, por ejemplo del DVD mediante:

```
$ kvm -no-reboot -cdrom /dev/cdrom -boot d -hda Windows7.img \
-m 500
```

Concluir la instalación de Windows 7 mediante:

```
$ kvm -no-reboot -boot c -hda Windows7.img -cdrom /dev/cdrom \
-m 500
```

Después de la instalación, es conveniente compactar y desfragmentar la imagen usando:

```
$ qemu-img convert -c Windows7.img -O qcow2 Windows.img
```

Ahora podemos usar la máquina virtual con la imagen desfragmentada y compactada de Windows 7 mediante:

```
$ kvm -boot c -hda Windows.img -m 500
```

Ejemplo 8 *Otro ejemplo completo de instalación y uso de una máquina virtual para Windows 11, en este caso necesitamos:*

Crear el disco virtual, por ejemplo de 80 GB mediante:

$ qemu-img create -f qcow2 Win.img 80G

Hacer la instalación de Windows 11 a partir, por ejemplo del iso mediante:

*$ kvm -boot d -cdrom tiny11\ b1.iso *
*-cpu host -smp cores=4,threads=1 -m 4096 *
*-machine q35,smm=on -usb -device usb-tablet -k es *
-drive file=Win.img,cache=writeback,media=disk

Después de la instalación, es conveniente compactar y desfragmentar la imagen usando:

$ qemu-img convert -c Win.img -O qcow2 Windows.img

Ahora podemos usar la máquina virtual con la imagen desfragmentada y compactada de Windows 11 mediante:

*$ kvm -cpu host -smp cores=4,threads=1 -m 4096 *
*-machine q35,smm=on -usb -device usb-tablet -k es *
-drive file=Windows.img,cache=writeback,media=disk

11.6 Uso de Virtualización Dentro de Otra Virtualización

Esta operación parece muy exótica y que rara vez se necesitará. Pero por las constantes vulnerabilidades descubiertas en los sistemas operativos, es muy común tener la última versión estable del sistema operativo para obtener el mejor desempeño posible del Hardware y la máxima seguridad posible en el sistema anfitrión y dentro de el, ejecutar una o más versiones de sistemas operativos huésped -no necesariamente actualizados- para dentro de ellos correr otras versiones de sistemas operativos obsoletos o vulnerables, permitiendo la estabilidad en entornos de producción, así como migraciones en vivo entre servidores. Esto se logra por ejemplo, para un procesador AMD al usar:

```
$ kvm -m 2048 -hda Linux.img -cpu phenom,+svm
```

Para un procesador Intel al usar:

```
$ kvm -m 2048 -hda Linux.img -cpu kvm64,+vmx
```

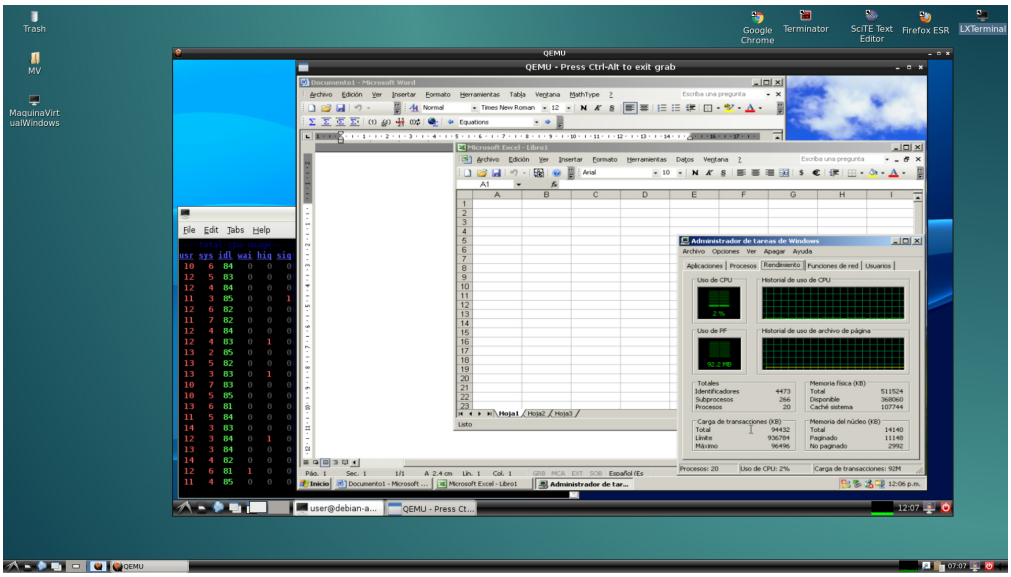

Figura 20: Sobre un equipo AMD de gama baja y 4 GB de RAM, se muestra el uso de una virtualización sobre otra virtualización y corriendo dentro de ella, una máquina virtual con Windows XP en la cual se muestra el uso de RAM y CPU dentro de la misma.

De esta forma, los sistemas virtualizados huésped heredan la capacidad de virtualizar por Hardware del anfitrión, acelerando los procesos anidados de las virtualizaciones, aumentando las posibilidades de uso de la virtualización al permitir explotar de forma eficiente el gran poder computacional que día a día se desarrolla[222].

11.7 Manipulación del Estado de la Máquina Virtual

La virtualización permite manipular el estado de una máquina en ejecución, por ejemplo, una vez que se está corriendo una máquina virtual:

```
$ kvm -boot c -hda Windows.img -m 400
```

es posible manipular el estado de la misma en algún punto de su ejecución al usar la combinación de teclas:

[222] Es común que un sólo equipo de cómputo cuente con hasta 1 TB de RAM y varios procesadores por tarjeta madre, donde cada procesador tiene decenas de Cores.

[Ctrl] + [Alt] + [2]

ya en ella, podemos detener y grabar el estado de la máquina virtual:

```
(qemu) savevm test.vm
(qemu) quit
```

para que en otro momento, podamos restaurar la máquina virtual tal como estaba cuando esta se detuvo mediante:

```
$ kvm -boot c -hda Windows.img -m 400 \
-loadvm test.vm
```

11.8 Optimización de Imágenes

Las imágenes de disco de KVM/QEMU después de ser generadas -al instalar algún sistema operativo o paquetes-, tienen muchos archivos internos dispersos, para optimizar su rendimiento es recomendable convertir la imagen dispersa en una que no tenga esta propiedad, mediante:

```
$ qemu-img convert disk-sparse.img -O qcow2 disk.img
```

o puede ser compactada y optimizada mediante:

```
$ qemu-img convert -c disk-sparse.img -O qcow2 disk.img
```

Para descompactar una imagen se hace mediante:

```
$ qemu-img convert disk-compact.img -O qcow2 disk.img
```

11.8.1 Trabajar con una Imagen Virtual sin que se Altere

Supongamos que tenemos creada una máquina virtual *debianStable.img* y la usamos de mediante:

```
$ kvm -hda debianStable.img -m 800
```

y ahora queremos hacer algunas instalaciones y pruebas pero no queremos dañar la máquina virtual original, entonces podemos hacer un instantánea temporal (Snapshot) en la que se guarden las modificaciones y al cerrar la máquina virtual se pierdan estas, entonces usamos:

```
$ kvm -hda debianStable.img –snapshot m 800
```

En algunos casos, es deseable que al trabajar con una máquina virtual, dejar la información de la máquina virtual base intacta y guardar los cambios que se requieran en otro archivo hasta que ya no sea requerida. Una forma de hacer esto, es hacer una copia y trabajar con la copia de esta o crear un archivo que almacene por separado sólo los cambios a la imagen, para esto último usamos[223]:

```
$ qemu-img create -b debianStable.img -f qcow2 debian.img
```

y así trabajamos con la imagen resultante (para este ejemplo *debian.img*) como con cualquier otra imagen:

```
$ kvm -hda debian.img -m 800
```

de esta forma, todos los cambios al trabajar serán almacenados en la imagen *debian.img* dejando intacta la imagen base *debianStable.img* y cuando ya no sea requerida puede ser borrada.

[223] Se pueden crear tantas instantáneas de una máquina virtual como sean requeridas, mientras no se cambie la imagen base.

11.8.2 Aumento de Desempeño

La virtualización normalmente es rápida, pero en algunas circunstancias se hace lenta, esto es ajeno a KVM/QEMU y generalmente es por la constante grabación de pequeños paquetes de datos al disco por parte de la máquina virtual.

Para optimizar el desempeño de la máquina virtual es posible pedirle a KVM/QEMU que trate de usar un Cache y baje lo menos posible a disco la información, esto aumentará notablemente el desempeño de la máquina virtual.

Para aumentar el desempeño, en lugar de usar:

```
$ kvm -boot c -hda Win.img -m 400
```

Usar:

```
$ kvm -drive file=Win.img,Cache=writeback,media=disk \
-m 400
```

En el caso de usar un archivo ISO, usar:

```
$ kvm -drive file=fedora.iso,Cache=writeback,media=cdrom
\
-m 512
```

Ejemplo 9 *Instalación y uso de una máquina virtual (por ejemplo para UBUNTU 11.10) usando el Cache, en este caso necesitamos:*

Generar un disco virtual, por ejemplo de 10 GB mediante:

$ qemu-img create -f qcow2 disco.img 10G

Instalar la imagen de UBUNTU en un disco virtual:

*$ kvm -no-reboot -boot d -drive file=ubuntu-11.10-desktop- *
*i386.iso,Cache=writeback,media=cdrom -drive file=disco.img, *
Cache=writeback,media=disk -m 500

Después de la instalación, es conveniente compactar y desfragmentar la imagen usando:

$ qemu-img convert -c disco.img -O qcow2 Ubuntu.img

Ahora ya podemos utilizar la imagen y hacer uso del Cache para acelerar el desempeño mediante:

$ kvm -drive file=Ubuntu.img,Cache=writeback,media=disk -m 500

Mejorando el Desempeño del Vídeo de la Máquina Virtual Por omisión se tiene un tarjeta gráfica de pobre desempeño en la máquina virtual, si se necesita mayor resolución en la salida gráfica, una opción es usar la opción *-vga*, donde algunas de sus posibilidades son: *std, cirrus, vmware, qxl, xenfb, tcx, cg3, virtio, none*, etc. Y podemos usarlas mediante:

```
$ kvm -drive file=fedora.iso,Cache=writeback,media=cdrom
\
-m 512 -vga std
```

o

```
$ kvm -drive file=fedora.iso,Cache=writeback,media=cdrom
\
-m 512 -vga vmware
```

hay otras opciones que permiten inclusive el uso de GPUs reales o virtuales.

Uso de Tarjeta de Sonido Dentro de KVM/QEMU Por omisión el uso de la tarjeta de audio no está habilitada, para habilitarla usar *-soundhw*, algunas posibles opciones son: *sb16, es1370, adlib, gus, cs4231a, hda, pcspk, all*, etc. Y podemos usarlas mediante:

```
$ kvm -boot c -hda Windows.img -m 400 \
-soundhw hda
```

hay otras opciones que permiten inclusive el uso de la tarjeta de sonido del equipo anfitrión.

Algunas Otras Opciones Lanzar KVM con dos procesadores, 1536 MB de RAM, dispositivo de red e1000, MAC addres[224] 52:54:00:12:34:50, iniciando el DHCP[225] en la dirección 10.0.2.40 y reenviando la salida del puerto 22 de la máquina virtual al 5555 del equipo huésped, mediante:

[224] En las redes de computadoras, la dirección MAC (siglas en inglés de Media Access Control) es un identificador de 48 bits (6 bloques de dos caracteres hexadecimales (4 bits)) que corresponde de forma única a una tarjeta o dispositivo de red. Se la conoce también como dirección física, y es única para cada dispositivo.

[225] Protocolo de configuración dinámica de Host (en inglés: Dynamic Host Configuration Protocol, también conocido por sus siglas de DHCP) es un protocolo de red de tipo

```
$ kvm -smp 2 -drive file=debianStableTmp.img,Cache=writeback,
media=disk -m 1536 -device e1000,netdev=user.0,mac=
52:54:00:12:34:50 -netdev user,id=user.0,dhcpstart=10.0.2.40,
Hostfwd=tcp::5555-:22 &
```

o lanzar kvm con dos procesadores, 1536 MB de RAM, dispositivo de red e1000 y reenviando la salida del puerto 22 de la máquina virtual al 5555 del equipo huésped de la siguiente forma:

```
$ kvm -smp 2 -drive file=debianStableTmp.img,Cache=writeback,
media=disk -m 1536 -device e1000,netdev=user.0 -netdev user,id=
user.0,Hostfwd=tcp::5555-:22 &
```

el redireccionamiento de puerto puede ser hecho también con:

```
$ kvm -m 512 -cpu phenom,+svm -hda b.qcow2 -redir tcp:5555:
10.0.2.15:22 &
```

Si se desea usar ssh y scp en la máquina virtual usar:

```
# apt install openssh-server
```

acceder usando:

```
$ ssh -p 5555 root@localHost
```

hacer copia del equipo huésped a la máquina virtual mediante:

```
$ scp -P 5555 file.txt usr@localHost:/tmp
```

cliente/servidor mediante el cual un servidor DHCP asigna dinámicamente una dirección IP y otros parámetros de configuración de red a cada dispositivo en una red para que puedan comunicarse con otras redes IP.

Algunos Problemas Comunes con la Red Por lo general las máquinas virtuales detectan correctamente la red, pero en el caso de Windows esto no siempre pasa, por ello es común emular una tarjeta de red lo más genérica posible, esta puede ser RTl8139, para ello es necesario que al lanzar la máquina virtual que se indique:

$$-net\ nic, model = rtl8139\ -net\ user$$

por ejemplo mediante:

```
$ kvm -boot c -hda WindowsXP.img -m 400 \
-net nic,model=rtl8139 -net user
```

algunas de las otras opciones para la red son: NE2000 PCI, RTL8139, PCNET y NE2000 ISA.

Direcciones de Red Usadas en QEMU/KVM

Gateway/DHCP/TFTP Server:	10.0.2.2
DNS Server:	10.0.2.3
Samba Server:	10.0.2.4
Netmask:	255.255.255.0
Guest IP:	cualquier dirección superior a 10.0.2.15

11.9 Uso de Máquinas Virtuales de VirtualBox en KVM/QEMU

Virtualbox es un programa desarrollado por Oracle ampliamente usado sobre todo para la plataforma Windows. Se trata de un Software multiplataforma capaz de virtualizar prácticamente la totalidad de sistemas operativos con arquitectura x86/amd64. La base de este Software dispone de una licencia GPL2 (véase 14.4), mientras que el Pack de extensiones que añaden funcionalidades están bajo licencia privativa, Virtualbox es gratuito para un uso no comercial.

VirtualBox (https://www.virtualbox.org/) dispone de diversas imágenes funcionales listas para descargar y usar varias decenas de distribuciones de Linux (https://virtualboxes.org/images/ y https://www.osboxes.org).

Interacción de VirtualBox en KVM/QEMU Ya que VirtualBox es ampliamente usado, KVM/QEMU ha desarrollado formas de usar, convertir y migrar máquinas de VirtualBox y otros manejadores de máquinas virtuales con un mínimo esfuerzo, ejemplo de ello es que se puede descargar cualquier imagen *VDI* de VirtualBox y usarla directamente en KVM usando la misma sintaxis que con sus propias máquinas virtuales.

Para mostrar esto, descargar de:

$$https://virtualboxes.org/images/lubuntu/$$

la imagen de LUBUNTU 12.10:

$$http : //sourceforge.net/projects/virtualboximage/files/Lubuntu/12.10/lubuntu1210.7z/download$$

y descomprimir el archivo lubuntu1210.7z, esto dejará una imagen de VirtualBox de LUBUNTU cuyo nombre es lubuntu1210.vdi. Entonces esta imagen la usaremos directamente en KVM/QEMU, mediante:

```
$ kvm -m 2000 -hda lubuntu1210.vdi
```

o

```
$ qemu-system-x86_64 -enable-kvm -m 2000 -hda lubuntu1210.vdi
```

Nota: esta imagen usa como usuario y clave de acceso: lubuntu/lubuntu

Algunas veces es necesario montar y extraer el contenido de un disco virtual, supongamos que tenemos una máquina virtual de VirtualBox y queremos ver su contenido, para ello usamos:

```
$ qemu-img convert diskname.vmdk -O qcow2 diskname.qcow2
```

o para el formato RAW:

```
$ qemu-img convert diskname.vmdk -O raw diskname.raw
```

Instalar nbd-client:

```
# apt install nbd-client
```

después:

```
# qemu-nbd –connect=/dev/nbd0 /mnt/kvm/diskname.qcow2
# fdisk /dev/nbd0 -l
# sudo mount /dev/nbd0p1 /mnt/somepoint/
# umount /mnt/somepoint/
```

11.10 Conversión de Máquinas Virtuales a KVM/QEMU

Es posible convertir máquinas virtuales de los proyectos de virtualización:

- VMware ESXi
- OVA exported from VMware
- VMX from VMware
- RHEL 5 Xen
- SUSE Xen
- Citrix Xen
- Hyper-V

a KVM/QEMU, mediante el comando *virt-v2v*, este convierte un Hipervisor de estos proyectos para ser ejecutado en *KVM/QEMU*. Puede leer máquinas virtuales de dichos proyectos de ambientes Linux y Windows que se ejecutan en *VMware, Xen, Hyper-V* y algunos otros Hipervisores, y convertirlos a *KVM* administrado por *libvirt, OpenStack, oVirt, Red Hat Virtualization (RHV)* u otros objetivos.

También hay un *Front-End*[226] complementario llamado *virt-p2v* que se presenta como una imagen *ISO, CD o PXE*[227] que se puede iniciar en máquinas físicas para virtualizar esas máquinas (de físico a virtual o *p2v*).

Ejemplos:

[226] En diseño de Software el Front-End es la parte del Software que interactúa con los usuarios

[227] Preboot eXecution Environment (PXE) (Entorno de ejecución de prearranque), es un entorno para arrancar e instalar el sistema operativo en computadoras a través de una red, de manera independiente de los dispositivos de almacenamiento de datos disponibles (como discos duros) o de los sistemas operativos instalados.

Convertir de VMware vCenter a un servidor libvirt local si se tiene una imagen de *VMware vCenter server* llamada *vcenter.example.com*, en un centro de datos llamado *Datacenter*, y un *ESXi*[228] hipervisor llamado *esxi*. Entonces podemos convertir el invitado llamado *vmware_guest* a una máquina virtual para *libvirt* de la siguiente manera:

```
$ virt-v2v -ic vpx://vcenter.example.com/Datacenter/esxi \
vmware_guest
```

en este caso es necesario ejecutar el comando en modo root, ya que necesita comunicación con el demonio[229] *libvirt* y copiar localmente en:
/var/lib/libvirt/images.

Convertir de VMware a RHV[230]/oVirt[231] este ejemplo es similar al anterior, excepto que se quiere enviar el huésped a *RHV Data Domain* usando *RHV REST API*. La interfaz de red del huésped debe ser conectada con la red del objetivo llamada *ovirtmgmt*, entonces:

```
$ virt-v2v -ic vpx://vcenter.example.com/Datacenter/esxi \
vmware_guest -o rhv-upload -oc https://ovirt-engine.exam\
ple.com/ovirt-engine/api -os ovirt-data -op /tmp/ovirt-ad\
min-password -of raw -oo rhv-cafile=/tmp/ca.pem -oo \
rhv-direct –bridge ovirtmgmt
```

[228] VMware ESXi (anteriormente VMware ESX) es una plataforma de virtualización a nivel de centro de datos producido por VMware, Inc.. Es el componente de su producto VMware Infraestructure que se encuentra al nivel inferior de la capa de virtualización, el hipervisor, aunque posee herramientas y servicios de gestión autónomos e independientes.

[229] En sistemas UNIX/LINUX se conoce como demonio o Daemon (Disk And Execution Monitor) a un proceso que se ejecuta en segundo plano del sistema operativo, se ejecuta en todo momento y no posee interacción directa con el usuario, también se le conoce genéricamente como servicio o proceso, del cual no percibimos su ejecución. Un demonio realiza una operación específica en tiempos predefinidos o en respuesta a ciertos eventos del sistema.

[230] Red Hat Virtualization REST Application

[231] oVirt is an open-source distributed Virtualization solution

en este caso el *Host*[232] ejecutando *virt-v2v* actúa como un servidor de conversión.

Convertir de ESXi hipervisor sobre SSH a libvirt local Si se tiene un hipervisor *ESXi* llamado *esxi.example.com* con acceso habilitado con *SSH*, entonces se puede convertir de *VMFS*[233] almacenamiento *VMFS* sobre el servidor a un archivo local de la siguiente forma:

```
$ virt-v2v \
-i vmx -it ssh \
"ssh://root@esxi.example.com/vmfs/volumes/datastore1\
/guest/guest.vmx" -o local -os /var/tmp
```

El huésped no debe estar corriendo y *virt-v2v* no necesita ser ejecutado por *root*.

Convertir imagen de disco a OpenStack Glance[234] dada una imagen en disco se puede convertir a otro hipervisor ejecutándose sobre *OpenStack* (sólo imágenes basadas en *OpenStack* sobre *KVM* son soportadas), para ello hacemos:

```
$ virt-v2v -i disk disk.img -o glance
```

Convertir imagen de disco a imagen de disco dada una imagen de disco de otro hipervisor que se quiera convertir a *KVM/QEMU* tenemos dos opciones:

```
$ virt-v2v -i disk disk.img -o local -os /var/tmp
```

[232] El término Host o anfitrión se usa en informática para referirse a las computadoras u otros dispositivos conectados a una red que proveen y utilizan servicios de ella. Los usuarios deben utilizar anfitriones para tener acceso a la red. En general, los anfitriones son máquinas monousuario o multiusuario que ofrecen servicios de transferencia de archivos, conexión remota, servidores de base de datos, servidores Web, etc.

[233] VMware VMFS es el sistema de archivos en clúster de VMware, Inc. utilizado por el paquete de virtualización de servidores insignia de la compañía, vSphere. Fue desarrollado para almacenar imágenes de disco de la máquina virtual, incluidas instantáneas.

[234] OpenStack es un proyecto de computación en la nube para proporcionar una infraestructura como servicio (IaaS).

el otro método más complejo es escribir un *libvirt XML* que describa el invitado a convertir (si se puede usar el hipervisor de origen para que proporcione el *libvirt XML* es mejor), entonces hacemos:

```
$ virt-v2v -i libvirtxml guest-domain.xml -o local -os /var/tmp
```

dado que *guest-domain.xml* contiene la(s) ruta(s) de las imágen(es) del disco invitado, no es necesario especificar el nombre de la imagen del disco en la línea de comandos.

Para convertir una imagen de disco local e inmediatamente iniciarla en *QEMU* local, hacemos lo siguiente:

```
$ virt-v2v -i disk disk.img -o qemu -os /var/tmp –qemu-boot
```

11.11 Comunicación de las Máquinas Virtuales con el Sistema Anfitrión e Internet

Para tener comunicación de las máquinas virtuales y el sistema anfitrión e internet, existen varias maneras de hacer esto, a saber:

a) Mediante el uso de algún navegador Web, se puede acceder a su cuenta de correo electrónico y al almacenamiento en la nube como *Google Drive, Dropbox, HubiC, pCloud, MediaFire, FlipDrive, Mega*, entre otros.

b) En el sistema operativo Linux, se puede acceder a cualquier servidor de internet mediante los protocolos SSH, SAMBA[235] o montar un sistema de archivos mediante SSHFS[236], NFS[237], entre otros.

1) Por ejemplo con *PCManFM, Dolphin, Nautilus, Thunar, Konqueror*, entre otros, podemos acceder a una máquina que tenga un servidor:

[235] Samba es una implementación libre del protocolo de archivos compartidos de Microsoft Windows (antiguamente llamado SMB, renombrado recientemente a CIFS) para sistemas de tipo UNIX. De esta forma, es posible que computadoras con GNU/Linux, Mac OS X o Unix en general se vean como servidores o actúen como clientes en redes de Windows.

[236] Secure SHell FileSystem (SSHFS) es un sistema de archivos para Linux (y otros sistemas operativos con una implementación FUSE, tal como en Mac OS X), que opera sobre archivos en una computadora remota usando un entorno seguro de acceso. En la computadora local donde se monta SSHFS, la implementación hace uso del módulo del kernel FUSE.

[237] El sistema de archivos de red (Network File System) es una aplicación cliente/servidor que permite al usuario ver y opcionalmente almacenar y actualizar archivos en un equipo remoto como si estuvieran en el propio equipo del usuario.

A) Acceder a un servidor SAMBA, escribir la ruta de archivos en el manejador de archivos:

smb://estud@192.168.13.230/estud/

B) Acceder a un servidor SSH, escribir la ruta de archivos en el manejador de archivos:

sftp://usuario@192.168.13.230/home/usuario/

2) En línea de comandos, podemos:

A) Montar con *SSHFS* un directorio de otra máquina con servidor *SSH*:

$ sshfs usuario@192.168.13.230:/home/usuario/ /home/algun/lugar

B) Montar con *mount* un directorio de otra máquina con servidor *NFS*:

mount 10.0.2.2:/directorio /home/algun/lugar

C) Usar SCP y SFTP de SSH para transferir archivos:

para copiar un archivo, usamos:

$ scp archivo.dat usuario@192.168.13.230:~/Datos/

para copiar un subdirectorio, usamos:

$ scp -r Directorio usuario@192.168.13.230:.

para copiar un archivo de una máquina remota a nuestra máquina, usamos:

$ scp usuario@192.168.13.230:/home/usuario/archivo .

c) En cualquier sistema operativo podemos usar algún navegador gráfico de *FTP, FTPS o SFTP* como *FileZilla, WinSCP, PSCP, PSFTP, FireFTP, CoreFTP*, entre muchos otros, para transportar archivos y carpetas.

d) Se puede usar *FSDEV* de KVM/QEMU que monta un recurso local mediante las siguientes indicaciones:

```
$kvm [...]
-fsdev local,id=fs1,path=$HOME/code/Linux,security_model=none
-device virtio-9p-pci,fsdev=fs1,mount_tag=Host-code
```

Donde *$HOME/code/Linux* es la ruta a compartir, y *Host-code* es el identificador para el montaje, en la MV se puede usar:

```
$ mkdir -p /mnt/Host
```

Donde */mnt/Host* es el directorio de montaje (checar que se den los permisos pertinentes), para ahora hacer:

```
# mount Host-code -t 9p /mnt/Host
```

Para desmontar usar:

```
# umount /mnt/Host
```

e) Mediante el uso de la línea de comandos (véase 3.4) usando el comando *scp* o *rsync* y sí así se requiere, comprimiendo con *tar* los directorios y archivos para su fácil traslado ó bien mediante programas que posean una interfaz gráfica de usuario para *SSH* o *SCP*.

f) Leer un dispositivo *USB* montado en el sistema anfitrión desde la máquina virtual, para ello el dispositivo *USB* deberá estar conectado en la máquina anfitrión y deberá ser accedido directamente en la máquina virtual. KVM/QEMU necesita parámetros adicionales, el parámetro *-usb* activa el soporte en la máquina virtual de dispositivos *USB*. La emulación de Intel SB82371 UHCI-Controller tiene 8-puertos en el USB hub. Si se busca tener acceso a uno de los dispositivos físicos, se requiere encontrar los parámetros *Vendor-ID* y *Product-ID*. Esta información se obtiene examinando la salida del comando:

```
# /sbin/lsusb
```

o

```
$ cat /proc/bus/usb/devices
```

Entonces es posible decirle a KVM/QEMU los datos de *VendorID* y *ProductID* a través de la línea de comandos (véase 3.4):

```
$ qemu -usb -usbdevice Host:<VendorID>:<ProductID> ...
```

o iniciar KVM/QEMU con soporte para dispositivos USB activados mediante:

```
$ qemu -usb ...
```

después de iniciar la máquina virtual, cambiar al sistema de monitoreo de la máquina virtual presionando:

```
[Ctrl]+[Alt]+[2] e introducir el siguiente comando:
usb_add Host:<VendorID>:<ProductID>
```

cuando se retorne al ambiente gráfico al teclear [Ctrl]+[Alt]+[1] se verá el mensaje de reconocimiento del dispositivo *USB*. Por ejemplo si se tiene una impresora HP Scanjet 3300C conectada en el puerto USB de la computadora, la salida del comando lsub es:

```
# lsusb
Bus 003 Device 002: ID 03f0:0205 ScanJet 3300C
```

así, el comando en KVM/QEMU para dejar accesible el dispositivo es:

```
$ qemu -usb -usbdevice Host:03f0:0205 ...
```

g) Usar la impresora conectada en el puerto paralelo, para ello al invocar la ejecución de la máquina virtual usar:

```
$ qemu -parallel /dev/parport0 ...
```

h) Montar el contenido de un disco virtual y poder intercambiar información entre la máquina virtual y la huésped, primero convertir el disco a formato accesible a Linux:

```
$ qemu-img convert disco.img -O raw tmp.img
```

montar la imagen en Linux como root:

```
# mkdir disk
# mount -o loop,offset=32256 tmp.img disk
```

trabajar con la imagen montada y al terminar desmontar esta:

```
# umount ./disk
```

y puede ser regresada al formato original mediante:

```
$ qemu-img convert -c tmp.img -O qcow2 disco.img
```

11.12 Montar el Disco Virtual en el Sistema Anfitrión

Algunas veces es necesario extraer información del disco de la máquina virtual. Si la máquina virtual está corriendo, podemos usar algunos métodos de conexión, como por ejemplo SSH, SFTP, etc. para transferir la información que necesitemos. Si la máquina no está corriendo, entonces del disco virtual es posible extraer la información montándolo en el sistema anfitrión, usando:

Para poder hacer el montaje, necesitamos tener instalado *qemu-utils*, si no se tiene instalado lo hacemos mediante:

```
# apt install qemu-utils
```

Cargamos el modulo del Kernel *NBD* usando:

```
# modprobe nbd max_part=8
```

Montamos la imagen del disco *Imagen.qcow2* como un dispositivo *NBD*:

```
# qemu-nbd -c /dev/nbd0 –read-only Imagen.qcow2
```

Si queremos podemos listar las particiones sobre *NBD*:

```
$ fdisk -l /dev/nbd0
```

Montamos la partición del sistema de archivos *NBD* en cualquier directorio, por ejemplo en */mnt/*:

```
# mount -o ro /dev/nbd0p1 /mnt
```

Ahora podemos ver todo el disco virtual en el directorio */mnt/* y extraer la información necesaria. Al terminar de extraer la información hay que desmontar el disco, usando:

```
# umount /mnt
```

11.13 Copiar la Máquina Virtual a un Disco

Si la máquina virtual generada la deseamos pasar a un disco físico (SSD, HD o USB) podemos hacerlo, mediante el procedimiento que a continuación indicaremos, pero debemos tener en cuenta que cuando se creó en disco se indico un tamaño particular, ese tamaño se necesitará en el disco en el que se monte la imagen de la máquina virtual, si el disco físico es mayor tamaño, se desperdiciará el espacio sobrante en el disco físico.

Si suponemos que tenemos un disco en */dev/sdd1* en el que volcaremos la imagen del disco virtual *Imagen.qcow2*, entonces la primera forma es:

```
# qemu-img dd -f qcow2 -O raw bs=4M if=Imagen.qcow2
of=/dev/sdd1
```

Segunda forma:

```
$ qemu-img convert Imagen.qcow2 -O raw Imagen.img
# dd if=Imagen.qcow2 of=/dev/sdd1
```

11.14 Significado de las Banderas de /proc/cpuinfo

Recordemos que para revisar si hay soporte en Hardware para la virtualización, usamos:

```
$ egrep "vmx|svm" /proc/cpuinfo
```

si soporta la virtualización por Hardware, aparecerá la bandera:

```
Procesadores INTEL: vmx
Procesadores AMD: svm
```

Hay una gran variedad de banderas que informan sobre el Hardware del que se dispone y las opciones que pueden usarse en *KVM/QEMU* que son soportadas por Hardware -como la virtualización dentro de una virtualización-, en esta sección veremos parte de ellas para poder usarlas si son necesarias para un proyecto en particular.

Intel Advanced Vector Extensions Programming Reference
fpu: Onboard FPU (floating point support)
vme: Virtual Mode Extensions (8086 mode)
de: Debugging Extensions (CR4.DE)
pse: Page Size Extensions (4MB memory pages)
tsc: Time Stamp Counter (RDTSC)
msr: Model-Specific Registers (RDMSR, WRMSR)
pae: Physical Address Extensions (support for more than 4GB of RAM)
mce: Machine Check Exception
cx8: CMPXCHG8 instruction (64-bit compare-and-swap)
apic: Onboard APIC
sep: SYSENTER/SYSEXIT
mtrr: Memory Type Range Registers
pge: Page Global Enable (global bit in PDEs and PTEs)
mca: Machine Check Architecture
cmov: CMOV instructions (conditional move) (also FCMOV)
pat: Page Attribute Table
pse36: 36-bit PSEs (huge pages)
pn: Processor serial number
clflush: Cache Line Flush instruction
dts: Debug Store (buffer for debugging and profiling instructions)
acpi: ACPI via MSR (temperature monitoring and clock speed modulation)
mmx: Multimedia Extensions
fxsr: FXSAVE/FXRSTOR, CR4.OSFXSR
sse: Intel SSE vector instructions
sse2: SSE2
ss: CPU self snoop
ht: Hyper-Threading
tm: Automatic clock control (Thermal Monitor)
ia64: Intel Itanium Architecture 64-bit (not to be confused with Intel's 64-bit x86 architecture with flag x86-64 or AMD64 bit indicated by flag lm)
pbe: Pending Break Enable (PBE# pin) wakeup support

AMD-defined CPU features, CPUID level 0x80000001
syscall: SYSCALL (Fast System Call) and SYSRET (Return From Fast System Call)
mp: Multiprocessing Capable.
nx: Execute Disable
mmxext: AMD MMX extensions
fxsr_opt: FXSAVE/FXRSTOR optimizations
pdpe1gb: One GB pages (allows hugepagesz=1G)
rdtscp: Read Time-Stamp Counter and Processor ID
lm: Long Mode (x86-64: amd64, also known as Intel 64, i.e. 64-bit capable)
3dnowext: AMD 3DNow! extensions
3dnow: 3DNow! (AMD vector instructions, competing with Intel's SSE1)

Transmeta-defined CPU features, CPUID level 0x80860001
recovery: CPU in recovery mode
longrun: Longrun power control
lrti: LongRun table interface

Other features, Linux-defined mapping
cxmmx: Cyrix MMX extensions
k6_mtrr: AMD K6 nonstandard MTRRs
cyrix_arr: Cyrix ARRs (= MTRRs)
centaur_mcr: Centaur MCRs (= MTRRs)
constant_tsc: TSC ticks at a constant rate
up: smp Kernel running on up
arch_perfmon: Intel Architectural PerfMon
pebs: Precise-Event Based Sampling
bts: Branch Trace Store
rep_good: rep microcode works well
nopl: The NOPL (0F 1F) instructions
xtopology: cpu topology enum extensions
tsc_reliable: TSC is known to be reliable
nonstop_tsc: TSC does not stop in C states
extd_apicid: has extended APICID (8 bits)

amd_dcm: multi-node processor
aperfmperf: APERFMPERF
eagerfpu: Non lazy FPU restore
nonstop_tsc_s3: TSC doesn't stop in S3 state

Intel-defined CPU features, CPUID level 0x00000001 (ecx)
pni: SSE-3 (Prescott New Instructions)
pclmulqdq: Perform a Carry-Less Multiplication of Quadword instruction – accelerator for GCM)
dtes64: 64-bit Debug Store
monitor: Monitor/Mwait support (Intel SSE3 supplements)
ds_cpl: CPL Qual. Debug Store
vmx: Hardware virtualization: Intel VMX
smx: Safer mode: TXT (TPM support)
est: Enhanced SpeedStep
tm2: Thermal Monitor 2
ssse3: Supplemental SSE-3
cid: Context ID
fma: Fused multiply-add
cx16: CMPXCHG16B
xtpr: Send Task Priority Messages
pdcm: Performance Capabilities
pcid: Process Context Identifiers
dca: Direct Cache Access
sse4_1: SSE-4.1
sse4_2: SSE-4.2
x2apic: x2APIC
movbe: Move Data After Swapping Bytes instruction
popcnt: Return the Count of Number of Bits Set to 1 instruction (Hamming weight, i.e. bit count)
tsc_deadline_timer: Tsc deadline timer
aes/aes-ni: Advanced Encryption Standard (New Instructions)
xsave: Save Processor Extended States: also provides XGETBY, XRSTOR,XSETBY
avx: Advanced Vector Extensions
f16c: 16-bit fp conversions (CVT16)
rdrand: Read Random Number from Hardware random number

generator instruction
hypervisor: Running on a hypervisor

VIA/Cyrix/Centaur-defined CPU features, CPUID level 0xC0000001
rng: Random Number Generator present (xstore)
rng_en: Random Number Generator enabled
ace: on-CPU crypto (xcrypt)
ace_en: on-CPU crypto enabled
ace2: Advanced Cryptography Engine v2
ace2_en: ACE v2 enabled
phe: PadLock Hash Engine
phe_en: PHE enabled
pmm: PadLock Montgomery Multiplier
pmm_en: PMM enabled

More extended AMD flags: CPUID level 0x80000001, ecx
lahf_lm: Load AH from Flags (LAHF) and Store AH into Flags (SAHF) in long mode
cmp_legacy: If yes HyperThreading not valid
svm: Secure virtual machine: AMD-V
extapic: Extended APIC space
cr8_legacy: CR8 in 32-bit mode
abm: Advanced Bit Manipulation
sse4a: SSE-4A
misalignsse: Misaligned SSE mode
3dnowprefetch: 3DNow prefetch instructions
osvw: OS Visible Workaround
ibs: Instruction Based Sampling
xop: extended AVX instructions
skinit: SKINIT/STGI instructions
wdt: Watchdog timer
lwp: Light Weight Profiling
fma4: 4 operands MAC instructions
tce: translation Cache extension
nodeid_msr: NodeId MSR
tbm: Trailing Bit Manipulation

topoext: Topology Extensions CPUID leafs
perfctr_Core: Core Performance Counter Extensions
perfctr_nb: NB Performance Counter Extensions
perfctr_l2: L2 Performance Counter Extensions

Auxiliary flags: Linux defined - For features scattered in various CPUID levels
ida: Intel Dynamic Acceleration
arat: Always Running APIC Timer
cpb: AMD Core Performance Boost
epb: IA32_ENERGY_PERF_BIAS support
xsaveopt: Optimized Xsave
pln: Intel Power Limit Notification
pts: Intel Package Thermal Status
dts: Digital Thermal Sensor
hw_pstate: AMD HW-PState
proc_feedback: AMD ProcFeedbackInterface
intel_pt: Intel Processor Tracing

Virtualization flags: Linux defined
tpr_shadow: Intel TPR Shadow
vnmi: Intel Virtual NMI
flexpriority: Intel FlexPriority
ept: Intel Extended Page Table
vpid: Intel Virtual Processor ID
npt: AMD Nested Page Table support
lbrv: AMD LBR Virtualization support
svm_lock: AMD SVM locking MSR
nrip_save: AMD SVM next_rip save
tsc_scale: AMD TSC scaling support
vmcb_clean: AMD VMCB clean bits support
flushbyasid: AMD flush-by-ASID support
decodeassists: AMD Decode Assists support
pausefilter: AMD filtered pause intercept
pfthreshold: AMD pause filter threshold

Intel-defined CPU features, CPUID level 0x00000007:0 (ebx)
fsgsbase: {RD/WR}{FS/GS}BASE instructions
bmi1: 1st group bit manipulation extensions
hle: Hardware Lock Elision
avx2: AVX2 instructions
smep: Supervisor Mode Execution Protection
bmi2: 2nd group bit manipulation extensions
erms: Enhanced REP MOVSB/STOSB
invpcid: Invalidate Processor Context ID
rtm: Restricted Transactional Memory
mpx: Memory Protection Extension
rdseed: The RDSEED instruction
adx: The ADCX and ADOX instructions
smap: Supervisor Mode Access Prevention

12 Escritorios Remotos y Virtuales

Con el propósito de que cualquier usuario que cuente con un dispositivo de cómputo[238] con red[239] puedan usar, configurar o instalar aplicaciones en los ambientes computacionales que se tienen instalados en otros equipos de forma remota, se crearon los escritorios remotos y los escritorios virtuales. Estos permiten visualizar la salida gráfica -de un sistema operativo en múltiples equipos o diversos sistemas operativos en un mismo equipo- por medio de internet (aún si la velocidad de conexión es baja).

Los casos de uso son muchos y se centran muy especialmente en los ámbitos de la asistencia remota[240] y del teletrabajo.

Escritorio Remoto Esta es una de las muchas aplicaciones que permiten acceder a un equipo de cómputo remoto mediante internet y controlarlo como si estuviéramos delante de él -más o menos-. Con estas aplicaciones, nos ahorramos tener que desplazarnos hasta donde está el equipo de cómputo al que queremos conectarnos, y así podemos por ejemplo ofrecer asistencia remota desde nuestro equipo de cómputo o usar los programas que se tienen instalados en un equipo remoto.

Algunas opciones de escritorios remotos[241] son:

- Chrome Escritorio Remoto (de descarga y uso gratuito), donde instalamos el servidor de escritorio remoto a través del navegador Chrome (o Chromium) en el equipo de cómputo a controlar y mediante el navegador Chrome en el otro equipo, se puede acceder remotamente cuando se necesita desde cualquier lugar con internet.

- "Asistencia rápida" (o Quick Assist) de Windows 10 es una utilidad del sistema operativo que permite a dos personas compartir un equipo

[238] Puede ser computadora personal , tableta, teléfono inteligente, Chromebook corriendo algún sistema operativo como Windows, Linux, MacOS, Android, Raspberry PI, IOS, Chrome, Solaris, HP-UX, AIX, etc.

[239] ¡Claro desde casa!, sin dirección IP pública fija homologada.

[240] Si algo no le funciona a alguien, estos servicios remotos nos permiten "meternos" en su equipo de cómputo y solucionarlo, incluso explicando mientras se está haciendo, porque se toma el control del teclado, ratón y pantalla, pero el usuario sigue teniendo control si quiere retomarlo y puede ver todo lo que hacemos en el escritorio remoto.

[241] Otras opciones son: Apple Remote Desktop, TeamViewer, SupRemo, Ammyy Admin, Iperius Remote, AnyDesk, VNC Connect, etc.

mediante conexión remota, pero sin necesidad de descargar ni instalar nada adicional.

Escritorio Virtual Es el acceso a un equipo de cómputo virtual en la nube, cuyo poder de procesamiento no se encuentra en un equipo de cómputo físico, sino en servidores ubicados en un centro de datos.

El usuario inicia sesión con sus credenciales y accede a un escritorio con las aplicaciones y programas instalados como si estuviera sentado frente a ese equipo de cómputo virtual.

¿Cuáles son las ventajas? los escritorios virtuales[242] tienen muchas ventajas para las organizaciones que necesitan entregar acceso a un equipo de cómputo con un conjunto establecido de aplicaciones. Se reducen los costos administrativos y mantenimiento de licencias. También facilita la solución de problemas de usuario y reduce problemas de seguridad.

Otras ventajas de esta tecnología:

1.- Administración centralizada de aplicaciones

2.- Se puede acceder desde dispositivos móviles, como teléfonos o tabletas.

3.- Facilita la aplicación de políticas organizacionales.

4.- Acelera la habilitación de nuevos puntos de trabajo.

Desde hace años existen ejemplos de estas tecnologías: "Windows Virtual Desktop" el escritorio virtual de Windows 10 sobre Microsoft Azure y recientemente con: "Cloud PC" para ofrecer "Desktop as a Service" de la división "Cloud Managed Desktops" de Microsoft.

Tipos de Servicios en la Nube Hay tres tipos principales de servicios en la nube: Software como servicio (SaaS) , Plataforma como servicio (PaaS) e Infraestructura como servicio (IaaS). No existe un enfoque único para la nube. Lo ideal es encontrar la solución adecuada que respalde los requisitos de un grupo de usuarios o empresa.

SaaS y sus beneficios el Software como servicio (SaaS) es un modelo de entrega de Software en el que las aplicaciones del cliente están alojadas en las instalaciones del proveedor de la nube. El cliente accede a sus aplicaciones

[242] Ejemplo de estos productos son: VMware Horizon, Citrix Virtual Apps and Desktops, Oracle Secure Global Desktop, HP Workspace, Amazon WorkSpaces, Parrallels Remote Application Server, Microsoft Windows Virtual Desktop, etc.

a través de internet. En lugar de pagar y mantener su propia infraestructura informática, el cliente aprovecha la suscripción al servicio de pago por uso.

Muchas empresas consideran que SaaS es la solución ideal porque les permite ponerse en marcha rápidamente con la tecnología más innovadora disponible. Las actualizaciones automáticas reducen la carga sobre los recursos internos. Los clientes pueden ampliar la escala de sus servicios para afrontar las cargas de trabajo fluctuantes y agregar más servicios o funciones a medida que van creciendo. Una suite de nube moderna proporciona un Software completo para cada necesidad empresarial, como la experiencia del cliente, la adquisición de ERP, la administración de la cartera de proyectos ERP, la cadena de suministro y la planificación empresarial.

PaaS y sus beneficios la plataforma como servicio (PaaS) brinda a los clientes la ventaja de acceder a las herramientas de desarrollador que necesitan para crear y administrar aplicaciones móviles y Web sin necesidad de invertir ni mantener la infraestructura subyacente. El proveedor aloja la infraestructura y los componentes de Middleware y el cliente accede a esos servicios a través de un navegador Web.

Para ayudar a la productividad, el proveedor de servicio PaaS ofrece componentes de programación listos para usar que permiten a los desarrolladores agregar nuevas capacidades a sus aplicaciones, incluidas tecnologías innovadoras tales como inteligencia artificial (IA), Chatbots, Blockchain e internet de las cosas (IoT). También incluye soluciones para analistas, usuarios finales y administradores profesionales de técnologías de la información (TI), incluidos análisis de Big Data, administración de contenido, administración de base de datos, administración de sistemas y seguridad.

IaaS y sus beneficios la infraestructura como servicio (IaaS) permite a los clientes acceder a servicios de infraestructura a pedido a través de internet. La ventaja clave es que el proveedor de la nube aloja los componentes de la infraestructura que proporcionan capacidad de cómputo, almacenamiento y red para que los suscriptores puedan ejecutar sus cargas de trabajo en la nube. El suscriptor de la nube generalmente es responsable de instalar, configurar, asegurar y mantener cualquier Software que se encuentre dentro de la infraestructura basada en la nube, como la base de datos, el Middleware y el Software de aplicación.

Beneficios del cómputo en la nube hay varias tendencias que impulsan a los usuarios y las empresas de todas las industrias a migrar hacia la nube. Para la mayoría de las organizaciones, la forma actual de hacer negocios podría no ofrecer la agilidad para crecer ni proporcionar la plataforma o la flexibilidad necesarias para competir. La explosión de datos creada por un número cada vez mayor de empresas digitales está llevando el costo y la complejidad del almacenamiento del centro de datos a nuevos niveles. Eso requiere nuevas habilidades y herramientas de análisis por parte del departamento de TI.

Las soluciones que ofrece la nube moderna ayudan a los usuarios y las empresas a enfrentar los desafíos de la era digital. En lugar de administrar su propia TI, la nube permite que las organizaciones respondan rápidamente a un panorama comercial más acelerado y complejo.

De qué manera la nube puede fomentar la innovación los clientes de la nube tienen automáticamente las últimas innovaciones y tecnologías emergentes integradas en sus sistemas de TI. El proveedor de la nube asume la responsabilidad de desarrollar nuevas capacidades y características. Los clientes que están en la nube obtienen esa ventaja estratégica.

Lo importante es la velocidad de la innovación. Los clientes de un proveedor pueden aprovechar una arquitectura de cómputo de nube moderna para innovar más rápido, aumentar la productividad y reducir los costos. Las capacidades integradas en la nube de múltiples proveedores (SaaS, PaaS e IaaS) proporcionan a las empresas la capacidad de pasar de las operaciones a la innovación. Las empresas pueden ofrecer nuevas aplicaciones y servicios, incluido el uso de tecnologías innovadoras como la inteligencia artificial (IA), Chatbots, Blockchain e internet de las cosas (IoT). Las empresas pueden aprovechar la abundancia de datos para obtener información predictiva de sus negocios y, en última instancia, obtener mejores resultados para sus clientes.

Confianza y seguridad Mudarse a la nube elimina los dolores de cabeza y los costos de mantener la seguridad de TI. Un proveedor de nube experimentado invierte continuamente en la última tecnología de seguridad, no solo para responder a posibles amenazas, sino también para permitir que los clientes cumplan mejor los requisitos de las reglamentaciones aplicables. Destacando que, la gran mayoría de estos servicios corren bajo alguna vari-

ante del sistema operativo Linux y dentro de él, se corren máquinas virtuales de diversos sistemas operativos según lo requiera el usuario.

12.1 Escritorio Remoto

Por un lado, si eres una de esas personas a los que colegas, amigos o familiares constantemente le piden ayuda para "arreglar su equipo de cómputo" porque creen que eres el que más sabe de tecnología, o por el otro estas desesperado porque necesitas ayuda en tu equipo de cómputo, entonces necesitas conocer la maravillosa herramienta de escritorio remoto.

Los programas "Chrome Escritorio Remoto" y "Asistencia Rápida de Windows" son dos de las muchas aplicaciones[243] que permiten acceder a un escritorio remoto y controlarlo remotamente como si estuviéramos delante de él -más o menos- cuando se necesita, desde cualquier lugar con internet.

En el caso de "Chrome Escritorio Remoto" es un servicio multiplataforma que solo requiere tener instalado el navegador Chrome -o sus derivados- para permitir el acceso sobre internet al equipo que se requiera. En el caso de "Asistencia Rápida" de Windows todo está listo para ser usado, pues es un paquete que viene integrado al sistema operativo Windows.

A continuación describiremos cómo usar dichos escritorios remotos.

12.1.1 Escritorio Remoto de Chrome

Es posible utilizar un equipo de cómputo o un dispositivo móvil para acceder a las aplicaciones y los archivos guardados en otro equipo de cómputo a través de internet gracias al Escritorio Remoto de Chrome (o Chromium). Podemos ver el vídeo de instalación y uso en cualquiera de las siguientes direcciones:

https://www.youtube.com/watch?v=P7xMQNB_9u0

[243] Si requiere de información adicional de algunos clientes de el escritorio remoto (para Windows, Linux, MacOS, Android, Raspberry PI, IOS, Chrome, Solaris, HP-UX, AIX) puede consultarlos en:
https://www.nobbot.com/pantallas/escritorio-remoto-en-chrome-como-instalarlo/
https://www.xataka.com/basics/escritorio-remoto-chrome-como-configurarlo-para-manejar-tu-ordenador-a-distancia
https://business.tutsplus.com/es/articles/best-remote-access-desktop-software–cms-31917
https://computerhoy.com/listas/software/mejores-programas-gratis-controlar-tu-escritorio-remoto-69563
https://www.xataka.com/basics/programas-escritorio-remoto

https://www.youtube.com/watch?v=YPAISPZC20U
https://www.youtube.com/watch?v=EGVhxS1t9yU

Es posible acceder al Escritorio Remoto de Chrome desde un equipo de cómputo conectándose a internet. Para acceder a un equipo de cómputo de forma remota desde un dispositivo móvil, es necesario ingresar a la cuenta personal de Google (@ciencias.unam.mx o @gmail.com) y descargar la aplicación: Escritorio Remoto de Chrome.

Configurar el acceso remoto a tu equipo de cómputo Puedes configurar el acceso remoto a tu equipo de cómputo Mac, Windows o Linux siguiendo estos pasos:

1.- Abre Chrome en tu equipo de cómputo.

2.- Escribe: remotedesktop.google.com/access en la barra de direcciones y pulsa: Enter.

3.- En la opción: "Configurar el acceso remoto" y selecciona: "Descargar".

4.- Sigue las instrucciones que aparecen en pantalla para descargar e instalar el Escritorio Remoto de Chrome.

Puede que tengas que escribir la contraseña de tu equipo de cómputo para que el Escritorio Remoto de Chrome pueda acceder. También es posible que se te pida que cambies la configuración de seguridad en Preferencias.

A continuación veremos cómo compartir tu ordenador con otro usuario para utilizar el Escritorio Remoto de Chrome.

Compartir tu equipo de cómputo con otro usuario Puedes permitir que otros usuarios accedan de forma remota a tu equipo de cómputo. Tendrán acceso completo a tus aplicaciones, archivos, correos electrónicos, documentos e historial.

1.- Abre Chrome en tu equipo de cómputo.

2.- Escribe: remotedesktop.google.com/support en la barra de direcciones y pulsa: Enter.

3.- En "Recibir asistencia" y selecciona: "Descargar".

4.- Sigue las instrucciones que aparecen en pantalla para descargar e instalar el Escritorio Remoto de Chrome.

5.- En la opción: "Recibir asistencia", selecciona: "Generar código".

6.- Copia el código y envíaselo a la persona que quieras que tenga acceso a tu equipo de cómputo -el código tiene una vigencia máxima de 5 minutos-.

7.- Cuando esa persona introduzca tu código de acceso en el sitio Web, se te mostrará un cuadro de diálogo con su dirección de correo electrónico. Selecciona la opción: "Compartir", para permitirle el acceso completo a tu equipo de cómputo.

8.- Para finalizar la sesión compartida, selecciona: "Dejar de compartir".

El código de acceso solo funcionará una vez. Cuando compartas tu equipo de cómputo, se te pedirá que confirmes que quieres seguir compartiéndolo cada 30 minutos.

Acceder a un equipo de cómputo de forma remota

1.- Abre Chrome en un equipo de cómputo.

2.- Escribe: remotedesktop.google.com/access en la barra de direcciones y pulsa: Enter.

3.- Haz clic en la opción: "Acceder" para seleccionar el equipo de cómputo que quieras.

4.- Introduce el PIN necesario para acceder a otro equipo de cómputo.

5.- Selecciona la flecha para conectarte.

Para tu protección, todas las sesiones de escritorio remoto están completamente cifradas.

Detener una sesión remota Cuando hayas terminado, cierra la pestaña para detener la sesión. También puedes seleccionar: "Opciones Desconectar".

Quitar un equipo de cómputo de la lista

1.- Abre Chrome en un equipo de cómputo.

2.- Escribe: remotedesktop.google.com/access en la barra de direcciones y pulsa: Enter.

3.- Junto al equipo de cómputo que quieras quitar, y selecciona: "Inhabilitar conexiones remotas".

Ofrecer asistencia remota

1.- Si alguien ha compartido contigo su código de acceso remoto, puedes ofrecerle asistencia de forma remota.

2.- Abre Chrome en un equipo de cómputo.

3.- Escribe: remotedesktop.google.com/access en la barra de direcciones y pulsa: Enter.

4.- En la opción: "Proporcionar asistencia", introduce el código y selecciona: "Conectar".

12.1.2 Escritorio Remoto de Windows

¿Cómo ayudar a alguien a resolver problemas en Windows accediendo y controlando su equipo de cómputo de forma remota sin instalar nada?

"Asistencia rápida" (o Quick Assist) es una utilidad del propio sistema operativo que viene integrada hace algún tiempo en Windows. Permite a dos personas compartir un equipo mediante conexión remota al estilo de *Team Viewer* y herramientas similares, pero sin necesidad de descargar ni instalar nada adicional[244].

Cómo Usar la Asistencia Rápida de Windows Asistencia rápida es probablemente una de las herramientas más fáciles de usar que te vas a encontrar, está diseñada especialmente para proporcionar apoyo técnico a alguien con pocos conocimientos informáticos y su uso es sencillo.

[244] Es necesario que ambos usuarios cuenten con una cuenta de Microsoft. Si requieres generar una cuenta, lo puedes hacer en:
https://account.microsoft.com/account?lang=es-es

Lo mejor de todo es que ya viene instalada en Windows, así que no tendrás que indicarle demasiados pasos complicados a la persona a la que quieres ayudar y esta no tendrá que descargar nada:

- Lo primero es sentarse frente a la computadora, tanto la persona que va a ofrecer ayuda como la que va a recibirla deben ejecutar la aplicación. Solo es cuestión de presionar la tecla de Windows, escribir: "Asistencia rápida" y presionar: Enter.
- Lo siguiente es seleccionar en cada equipo de cómputo cuál será el rol de cada uno. Primero la persona que va a ayudar debe seleccionar: "Ayudar a otra persona" y seleccionar el botón azul en la parte inferior.
- Después deberá iniciar sesión con su cuenta de Microsoft y una vez hecho esto aparecerá un código de seguridad de seis dígitos mismo que tendrá que compartir con la persona a la que va a ayudar.
- Por razones de seguridad ese código solo será válido durante los próximos 10 minutos. Así que antes de que acabe ese tiempo, deberás compartir el código con la persona que recibe ayuda desde su casa y esta deberá ingresarlo en el cajón de: "Asistencia rápida" justo debajo de donde dice: "Obtener asistencia".
- Una vez hecho esto la persona que ofrece ayuda tendrá que seleccionar entre dos opciones: tomar el control total del equipo de cómputo o solo ver la pantalla. La primera le permitirá usar el equipo de cómputo de la otra persona como si estuviese frente a él, la segunda solo le deja mirar qué está pasando sin poder interactuar.
- Cuando el ayudante haya elegido una opción, la persona que recibe ayuda deberá confirmar que está dando permiso para que se vea su pantalla o se controle su equipo de cómputo.
- En la ventana aparecerá el nombre de usuario que la persona que ayuda tiene en su cuenta de Microsoft, y al final solo hay que hacer Click en el botón: "Permitir".
- Si el que recibe ayuda completa los permisos, de inmediato en el escritorio de Windows de quien asiste aparecerá una ventana con el escritorio de la persona a la que ayuda, con o sin control de este dependiendo de lo que se haya elegido.

- Asistencia rápida también permite abrir una pequeña ventana de Chat para enviar indicaciones a la otra persona. La resolución de pantalla del anfitrión se escala de forma automática cada vez que redimensiones la ventana y en general funciona sumamente bien y es muy responsivo.

- Quien recibe ayuda puede detener la "Asistencia rápida" en cualquier momento, solo tiene que presionar a: "X" en el mensaje que le indica que el uso compartido de pantalla está activado. También es posible pausar la sesión en lugar de terminarla del todo.

Si en estos tiempos de trabajo remoto y confinamiento necesitas ayudar a alguien qué esta lejos, si ambos tienen Windows esta es sin duda una herramienta extremadamente útil y poderosa.

12.2 Escritorio Virtual

Las plataformas de escritorio remoto son como decimos una excelente solución para tareas de administración y asistencia remota, pero estas soluciones pueden quedarse cortas si queremos ir a un objetivo más ambicioso: el de poder trabajar con un escritorio remoto virtual.

Como habíamos visto anteriormente, eso es lo que ofrecen las plataformas de escritorio virtual y variantes como las plataformas DaaS (Desktop as a Service). Estas últimas son simplemente una implementación VDI (Virtual Desktop Infraestructure) sobre la nube.

La diferencia entre ellas dos es que usando un VDI una empresa u organización puede implementar escritorios virtuales desde sus centros de datos locales y son sus técnicos los que deben implementar y gestionar esa infraestructura. Con DaaS todo se basa en la nube y no es necesario adquirir Hardware, porque otra empresa proporciona tanto los servidores como la plataforma, su gestión y su mantenimiento.

En estas plataformas la idea es siempre la misma: ofrecer a los usuarios acceso a un escritorio virtual alojado en la nube. Pueden acceder a ese "equipo de cómputo virtual" desde cualquier otro dispositivo (otro equipo de cómputo más o menos potente, un móvil, una tableta), y trabajar en cualquier dispositivo, por modesto que sea, con el entorno y las aplicaciones que la empresa ha puesto a disposición de sus usuarios en esos equipos de cómputo virtuales.

Las ventajas para las empresas son numerosas: los usuarios o empleados pueden acceder a sus sesiones de trabajo desde cualquier sitio y dispositivo, las empresas ahorran recursos a la hora de actualizar y mantener la infraestructura que es utilizada en los puestos de trabajo.

Además, ese tipo de escritorios virtuales garantizan un acceso seguro a todas las aplicaciones -sin que el usuario tenga que usar equipos propios que también pueda tener para uso personal- y de esta manera los usuarios no tienen que preocuparse de las actualizaciones o de instalar nuevas aplicaciones. La gestión está centralizada, es mucho más sencilla y homogénea, además de ser totalmente escalable y adaptarse dinámicamente a las necesidades de la empresa, también incluye temas muy importantes como el de la realización de copias de seguridad.

12.2.1 Escritorios y Máquinas Virtuales con VNC

Con el propósito de que el usuario que tenga acceso a un equipo de cómputo, tableta, teléfono inteligente, Chromebook o dispositivo conectado a la red, pueda usar los ambientes computacionales que se tienen instalados en un equipo determinado, se ha desarrollado el servidor de computación virtual en red VNC[245] (Virtual Network Computing) que permite visualizar la salida gráfica (de un sistema operativo en múltiples equipos o diversos sistemas operativos en un mismo equipo) por medio de red, aún si la velocidad de conexión es baja.

VNC es un programa de Software libre basado en una estructura cliente-servidor que permite interactuar con el servidor remotamente a través de un dispositivo que disponga de un cliente VNC (Windows, Linux, MacOS, Android, Raspberry PI, iOS, Chrome, Solaris, HP-UX, AIX) o bien usando algún navegador Web (con seguridad WSS) si la salida VNC se manda como

[245] Si requiere de información adicional de algunos clientes de VNC (para Windows, Linux, MacOS, Android, Raspberry PI, IOS, Chrome, Solaris, HP-UX, AIX) puede consultarlos en:
https://www.realvnc.com/es/connect/download/vnc/
https://www.geckoandfly.com/23203/vnc-client-viewer-windows-mac-linux/
https://lifehacker.com/the-best-vnc-client-for-android-5838717
https://www.tecmint.com/best-remote-linux-desktop-sharing-Software/
https://www.lifewire.com/vnc-free-software-downloads-818116
https://thelinuxcode.com/vnc-viewer-client/
https://www.howtogeek.com/142146/how-to-use-google-chrome-to-remotely-access-your-computer/

HTML (se puede usar por ejemplo el paquete noVNC) compartiendo la pantalla, teclado y ratón, sin imponer restricciones del equipo servidor con respecto al del cliente (también conocido como Computación en la Nube).

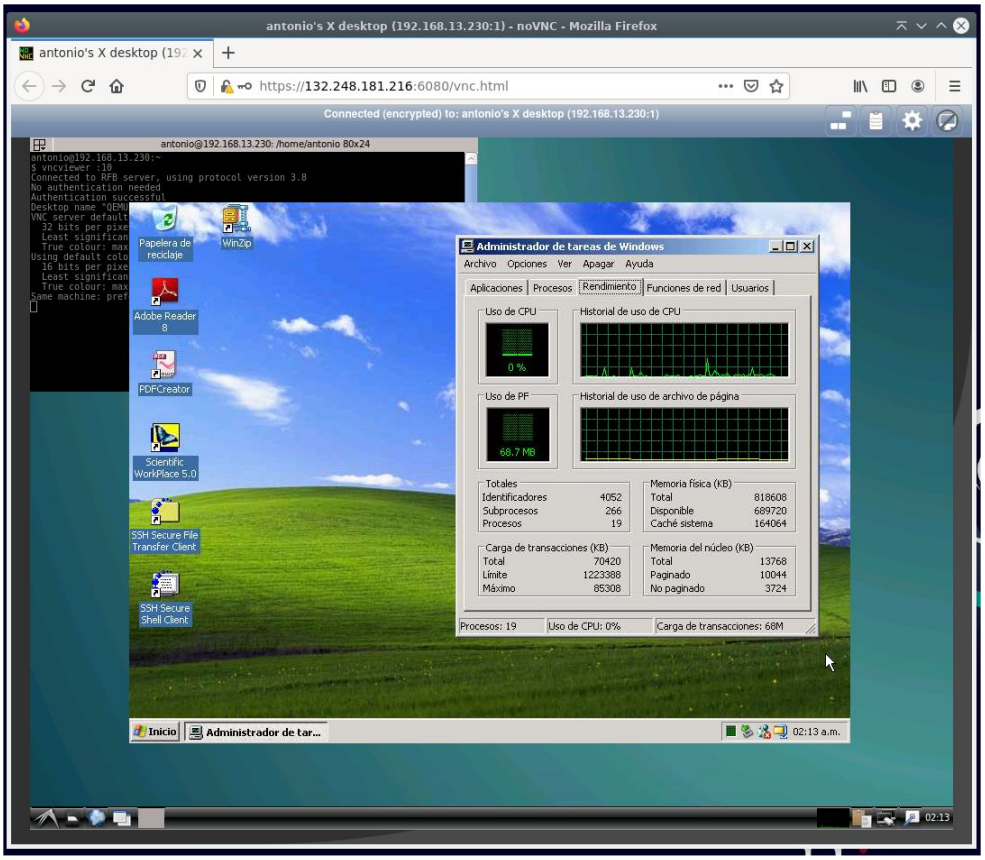

Figura 21: Uso de noVNC desde el navegador Mozilla Firefox sobre un equipo en red que soporta ejecución de máquinas virtuales sobre Debian GNU/Linux.

Cualquier equipo puede convertirse en un servidor de VNC, instalando alguna aplicación como: TigerVNC, RealVNC, Connections, TeamViewer, Remmina, NoMachine, Apache Guacamole, XRDP, FreeNX, X2Go, XPra, AnyDesk, Krdc, Krfb, Vino, vnc4server, x11vnc, tigervnc-viewer, Vinagre, Desktopable, Ammyy Admin, Gnome-rdp, entre otros.

Para mostrar cómo trabajar con uno, usaremos tightvncserver, es ligero

y se instala facilmente en Debian GNU/Linux, mediante:

```
# apt install tightvncserver
```

no requiere ninguna configuración adicional por parte del administrador (para este ejemplo supondremos que la IP del servidor es 192.168.13.230). Para compartir una sesión del usuario mediante VNC (sin cifrado) en el servidor (usando por ejemplo el puerto 30[246]) escribimos:

```
$ vncserver :30
```

la primera vez pedirá la clave de acceso (de 8 caracteres) y su confirmación, además si así lo queremos podemos generar otra clave de acceso para que se vea el escritorio en modo de solo lectura (se pueden usar para que por ejemplo, múltiples alumnos vean en distintos equipos lo que se hace en el escritorio principal sin que interfieran en él). En caso de que el puerto esté ocupado, usamos otro e intentaremos nuevamente levantar el servidor.

Cada usuario puede lanzar tantos escritorios remotos como sea necesario usando distintos puertos en el servidor de VNC, en cada uno de ellos verá un escritorio propio, pero compartirán el directorio de trabajo así como la clave de acceso.

Después de seguir estos pasos, ya es posible conectarse desde cualquier equipo con algún cliente de VNC, usando el puerto seleccionado. Por ejemplo en Debian GNU/Linux[247], si se instala como cliente de VNC a xtightvncviewer mediante:

```
# apt install xtightvncviewer
```

entonces podemos ver el escritorio compartido, usando:

```
$ vncviewer 192.168.13.230:30
```

cuando ya no se necesite el servidor de VNC, se debe finalizar el servidor de VNC del puerto levantado, mediante:

```
$ vncserver -kill :30
```

[246] El puerto a usar está dentro del rango 0 en adelante -0 equivale al puerto físico 5900-.

[247] Para poder interactuar con el escritorio remoto o máquina virtual mediante VNC en Windows es necesario bajar e instalar algún cliente, por ejemplo alguno de estos paquetes:

https://www.realvnc.com/es/connect/download/vnc/windows/
https://www.tightvnc.com/

¿Qué se puede o no se puede hacer en VNC? Desde un equipo remoto se puede hacer casi cualquier cosa -salvo escuchar el audio aunque hay proyectos trabajando en ello- que sea posible hacer sentado delante de un equipo, así como utilizar el teclado y el ratón. No se pueden controlar de forma remota dispositivos Apple iOS y Android desde un equipo de escritorio, aunque sí lo contrario.

En el equipo GNU/Linux se pueden crear cuentas individuales, compartidas y para grupos de trabajo. Además de emular diversas arquitecturas (x86_64, PowerPC, Sparc32 y 64, MIPS, ARM, ColdFire, Cris, MicroBlaze, SH4, Xtensa, entre otros) y sus respectivos procesadores; permitiendo la ejecución de máquinas virtuales para ser usadas en forma monousuario o multiusuario.

Para proporcionar el servicio de VNC usamos TigerVNC (*tightvncserver*), es una implementación de VNC neutra, independiente, de alto rendimiento y de código abierto. Es una aplicación cliente-servidor que permite a los usuarios iniciar e interactuar con aplicaciones gráficas en máquinas remotas y/o máquinas virtuales basadas en QEMU/KVM.

A diferencia de otros servidores VNC como *VNC X*, *Vino* o *Connections* que se conectan directamente con el escritorio en tiempo de ejecución, *tigervnc-vncserver* utiliza un mecanismo diferente que configura un escritorio virtual independiente para cada usuario. Es capaz de ejecutar aplicaciones de vídeo y 3D, tratando de mantener una interfaz de usuario coherente y reutilizar componentes, donde sea posible, a través de las diversas plataformas que admite. Además, ofrece seguridad a través de una serie de extensiones que implementan métodos avanzados de autenticación y cifrado TLS.

Usando VNC en GNU/Linux Por ejemplo, el usuario que disponga de una cuenta en algún servidor o que instale GNU/Linux, puede usar máquinas virtuales y compartir su escritorio de forma remota[248] sin cifrado o con él. Por ejemplo, usaremos tightvncserver como servidor de VNC, lo instalamos usando:

```
# apt install tightvncserver
```

y a xtightvncviewer como cliente de VNC, lo instalamos usando:

[248] Si no contamos con una IP pública fija homologada y deseamos dar acceso a nuestro equipo de cómputo fuera de nuestra red, será necesario instalar y configurar una Virtual Private Network VPN (Red Privada Virtual) como OpenVPN.

```
# apt install xtightvncviewer
```

Uso de VNC sin Cifrado para hacer uso de VNC sin cifrado (y mayor velocidad en la transmisión), hay que acceder al servidor remoto usando SSH[249] o MOSH[250] (suponiendo la dirección 192.168.13.230), usamos:

```
$ ssh usuario@192.168.13.230
```

Una vez iniciada la sesión, es necesario levantar el servidor de VNC, usando por ejemplo el puerto[251] 30 (equivalente al puerto físico 5930), escribimos:

```
$ vncserver :30[252]
```

la primera vez pedirá la clave de acceso (de 8 caracteres) y su confirmación, además si así lo queremos podemos generar otra clave de acceso para que se vea el escritorio en modo de solo lectura (se pueden usar para que por ejemplo, múltiples alumnos vean en distintos equipos lo que se hace en el escritorio principal sin que interfieran en él). En caso de que el puerto esté ocupado, usamos otro e intentaremos nuevamente levantar el servidor.

[249] SSH (o Secure SHell) es un protocolo que facilita las comunicaciones seguras entre dos sistemas usando una arquitectura cliente/servidor y que permite a los usuarios conectarse a un "anfitrión" (Host) remotamente. A diferencia de otros protocolos de comunicación remota tales como FTP o Telnet, SSH encripta la sesión de conexión, haciendo imposible que alguien pueda obtener acceso a nuestros datos.

[250] MOSH (Mobile Shell) como medio de conexión (no corta la comunicación por inactividad como SSH), lo podemos instalar usando:

```
# apt install mosh
```

y su uso es similar al de SSH:

```
$ mosh usuario@192.168.13.230
```

[251] El puerto a usar está dentro del rango 0 en adelante -0 equivale al puerto físico 5900-.

[252] Podemos cambiar el tamaño de la ventana del cliente de VNC, además por omisión se usan 32 bits por cada Pixel, esto puede resultar muy pesado para conexiones de internet lentas, por ello se sugiere usar 24, 16 u 8 bits por Pixel, indicándolo de la siguiente forma:

```
$ vncserver -geometry 1280x768 -depth 16 :30
```

Cada usuario puede lanzar tantos escritorios remotos como sea necesario usando distintos puertos en el servidor de VNC, en cada uno de ellos verá un escritorio propio, pero compartirán el directorio de trabajo así como la clave de acceso.

Después de seguir estos pasos, ya es posible conectarse desde cualquier equipo con algún cliente de VNC, usando el puerto 30:

```
$ vncviewer 192.168.13.230:30
```

Cuando ya no se necesite el servidor de VNC, hay que conectarse de nuevo al servidor remoto y finalizar el servidor de VNC del puerto o puertos levantados, usando:

```
$ vncserver -kill :30
```

Uso de VNC con Cifrado en la máquina en la que se usará el cliente de VNC, lanzamos la tunelización del cliente usando SSH (suponiendo la dirección 192.168.13.230), mediante:

```
$ ssh -L 5930:localhost:5930 –C N -f -l usuario 192.168.13.230
```

y accedemos al servidor usando SSH, mediante:

```
$ ssh usuario@192.168.13.230
```

ahora debemos lanzar el servidor VNC, pero este debe ser sólo local al servidor, si usamos el puerto[253] 30, entonces usamos:

```
$ vncserver -localhost :30[254]
```

[253] El puerto a usar está dentro del rango 0 en adelante -0 equivale al puerto físico 5900-.

[254] Podemos cambiar el tamaño de la ventana del cliente de VNC, además por omisión se usan 32 bits por cada Pixel, esto puede resultar muy pesado para conexiones de internet lentas, por ello se sugiere usar 24, 16 u 8 bits por Pixel, indicándolo de la siguiente forma:

```
$ vncserver -geometry 1280x768 -depth 16 localhost :30
```

la primera vez pedirá la clave de acceso (de 8 caracteres) y su confirmación, además si así lo queremos podemos generar otra clave de acceso para que se vea el escritorio en modo de solo lectura. En caso de que el puerto este ocupado, usamos otro e intentaremos nuevamente levantar el servidor.

Cada usuario puede lanzar tantos escritorios remotos como sea necesario usando distintos puertos en el servidor de VNC, en cada uno de ellos verá un escritorio propio, pero compartirán el directorio de trabajo así como la clave de acceso.

Ahora ya podemos lanzar el cliente de VNC, mediante:

```
$ vncviewer localhost:30
```

Cuando ya no se necesite el servidor de VNC, hay que conectarse de nuevo al servidor remoto y finalizar el servidor de VNC del puerto o puertos levantados, usando:

```
$ vncserver -kill :30
```

12.3 Desde la Nube

Existen diferentes servicios Web[255] usando VNC que permiten instalar, configurar y usar cientos de sistemas operativos Linux y Unix -máquinas virtuales usando servicios Web en Debian GNU/Linux y QEMU- desde el navegador, esto en aras de que los usuarios que cuenten con algún sistema de acceso a red y un navegador puedan usar, configurar e instalar algún sistema operativo y su respectiva paquetería sin hacer instalación alguna en su equipo de cómputo, tableta o teléfono celular[256].

Una muestra de estos proyectos son: Distrotest (https://distrotest.net) y JSLinux (https://bellard.org/jslinux).

Algunas versiones listas para usar son:

[255] Cuando se trabaja desde la Web es recomendable usar el modo Privado o Incógnito para no guardar el historial de navegación, información introducida en los formularios y borrar al cerrar el navegador los datos de los sitios visitados. Pero recuerda que los sitios Web que visitamos sí guardan información de nuestra visita, nuestro proveedor de internet también guarda constancia de nuestra visita y si descargamos algo, esto no se borra al igual que el historial de descargas, además de las marcas de páginas o favoritos se conservarán al cerrar el navegador.

[256] Estos servicios son conocidos como computación en la nube (Cloud Computing).

4mLinux, AbsoluteLinux, Academix, AlpineLinux, Antergos, antiX Linux, Aptosid, ArchBang, ArchLabs, ArchLinux, Archman, ArchStrike, ArcoLinux, ArtixLinux, AryaLinux, AV Linux, BackBoxLinux, BigLinux, Bio-Linux, BlackArch, BlackLab, BlackPantherOS, BlackSlash, blag, BlankOn, Bluestar, Bodhi, BunsenLabs, ByzantineOS, Caine, Calculate Linux Desktop, CentOS, Chakra, ChaletOS, ClearOS, Clonezilla, ConnochaetOS, Cucumber, Damn Small Linux, Damn Small Linux Not, Debian, DebianEdu, deepin, DEFT, Devil-Linux, Devuan, DragonFly BSD, Dragora, DuZeru, Dyne:bolic, Edubuntu, elementaryOS, Elive Linux, Emmabuntüs, Emmi OS, Endless OS, EnsoOS, Exe GNU/Linux, ExTiX, Fatdog64, Fedora Atomic, Fedora Server, Fedora Workstation, FerenOS, FreeBSD, FreeDOS, Frugalware, G4L, GeckoLinux, Gentoo, GNewSense, GoboLinux, Gparted, GreenieLinux, GRML, GuixSD, Haiku, Heads, Kali Linux, Kanotix, KaOS, Knoppix, Kodachi, KolibriOS, Korora, Kubuntu, Kwort, Linux Lite, Linux Mint, LiveRaizo, LMDE, Lubuntu, LXLE OS, Macpup, Mageia, MakuluLinux, Manjaro, Matriux, MauiLinux, MenuetOS, MinerOS, MiniNo, Modicia, Musix, MX Linux, Nas4Free, Neptune, NetBSD, Netrunner, NixOs, NST, NuTyX, OpenIndiana, OpenMandriva, openSUSE, OracleLinux, OSGeo live, OviOS, Parabola CLI, Parabola LXDE, Pardus, Parrot Home, Parrot Security, Parrot Studio, Parsix, PCLinuxOS, PeachOSI, Pentoo, Peppermint, PeppermintOS, Pinguy, PinguyOS, plopLinux, PointLinux, Pop!_OS, PORTEUS, Puppy Linux, PureOS, Q4OS, QubesOS, Quirky, Raspberry Pi Desktop, ReactOS, Redcore, Rescatux, RevengeOS, RoboLinux, Rockstor, ROSA FRESH, Runtu, Sabayon, SalentOS, Salix, ScientificLinux, Siduction, Slackware, Slax, SliTaz, Solus, SolydK, SolydX, SparkyLinux, Springdale, StressLinux, SubgraphOS, SwagArch, Tails, Tanglu, Tiny Core, Trisquel, TrueOS, TurnKey Linux, Ubuntu, Ubuntu Budgie, Ubuntu Studio, UbuntuKylin, Uruk, VectorLinux, VineLinux, VoidLinux, Voyager, VyOS, WattOs, Xubuntu, Zentyal, Zenwalk, Zevenet, Zorin OS

Terminales de Linux en la Web

- https://www.tutorialspoint.com/execute_bash_online.php
- http://www.webminal.org/
- https://bellard.org/jsLinux/
- https://codeanywhere.com/
- https://copy.sh/v86/
- https://www.masswerk.at/jsuix/
- https://linuxcontainers.org/lxd/try-it/
- http://cb.vu/

Editores BAHS en la Web

- https://www.shellcheck.net/
- https://www.learnshell.org/
- https://www.tutorialspoint.com/execute_bash_online.php
- https://paiza.io/en/projects/new?language=bash
- https://www.jdoodle.com/test-bash-shell-script-online
- http://rextester.com/l/bash_online_compiler

Windows 365 Microsof presentó en julio del 2021 la nueva versión de Windows en la nube que llevará el nombre de Windows 365. Este nuevo servicio de suscripción, está especialmente dirigido a empresas, que permitirá acceder a nuestra sesión de usuario desde cualquier equipo (el PC, el Mac, la tableta o teléfono Android, etc.), pues el Software y el sistema de archivos estarán alojados en una máquina virtual remota, por lo que la confi-guración, documentos y herramientas disponibles serán idénticos desde donde accedamos.

Así, desde cualquier navegador (o bien usando la aplicación de Escritorio Remoto de Windows) podremos acceder a un Windows 10/11 disfrutando de una experiencia de arranque casi instantáneo, pero esa no será la única ventaja de esta plataforma. Aún más importante será la posibilidad de contar con varios 'ordenadores' en una misma cuenta, cada uno con distinta potencia (RAM, núcleos de procesador...) y capacidad de almacenamiento contratada, según el trabajo que necesitemos llevar a cabo en cada momento.

Microsoft ya ha confirmado que ofrecerá 12 configuraciones de Hardware distintas para sus equipos virtualizados (iniciando en $130 pesos mexicanos por mes). Así, las empresas podrán 'crear PCs' en cuestión de minutos y asignar cada uno a un empleado, eliminando los inconvenientes que conlleva el hecho de manejar Hardware físico.

13 Herramientas Administrativas y de Seguridad

Existen diversos sitios Web que están enfocados a explorar detalladamente cada distribución actual o antigua, a un nivel técnico acompañado de grandes y útiles análisis técnicos sobre los mismos, lo que facilita el aprendizaje puntual sobre qué distribución usar o empezar a usar sin tanta incertidumbre, algunos de estos lugares son:

- ArchiveOS https://archiveos.org
- Distro Chooser https://distrochooser.de/es/
- Distro Watch https://distrowatch.com
- Linux Distribution List https://lwn.net/Distributions/

¿Qué otros sabores de GNU/Linux hay?

https://upload.wikimedia.org/wikipedia/commons/1/1b/Linux_Distribution_Timeline.svg

Existen distintas distribuciones de GNU/Linux[257] para instalar, puedes conocer y descargar las diferentes distribuciones desde:

https://es.wikipedia.org/wiki/Anexo:Distribuciones_Linux
https://en.wikipedia.org/wiki/List_of_Linux_distributions

y ver cual es la que más te conviene:

https://en.wikipedia.org/wiki/Comparison_of_Linux_distributions

o probar alguna versión Live[258]:

[257] Una lista de las distribuciones de Linux y su árbol de vida puede verse en la página Web http://futurist.se/gldt/

[258] Linux es uno de los sistemas operativos pioneros en ejecutar de forma autónoma o sin instalar en la computadora, existen diferentes distribuciones Live -descargables para formato CD, DVD, USB- de sistemas operativos y múltiples aplicaciones almacenados en un medio extraíble, que pueden ejecutarse directamente en una computadora, estos se descargan de la Web generalmente en formato ISO.

https://livecdlist.com/

también las puedes correr como máquina virtual para VirtualBox:

https://www.osboxes.org/

o máquina virtual para QEMU/KVM:

https://docs.openstack.org/image-guide/obtain-images.html
https://github.com/palmercluff/qemu-images
https://bierbaumer.net/qemu/

por otro lado, existen diferentes servicios Web que permiten instalar, configurar y usar cientos de sistemas operativos Linux y Unix desde el navegador, una muestra de estos proyectos son:
Distrotest https://distrotest.net
JSLinux https://bellard.org/jslinux
OnWorks https://www.onworks.net

Entre las distintas distribuciones de GNU/Linux para instalar, una de las más ampliamente usadas es Debian GNU/Linux[259] y sus derivados como Ubuntu. La comunidad de Debian GNU/Linux te apoya para que lo obtengas, instales y de una vez por todas puedas usar GNU/Linux en tu computadora.

Tenemos una réplica en México de Debian GNU/Linux en:

http://ftp.mx.debian.org/debian/

de aquí puedes descargar las imágenes de instalación:

http://ftp.mx.debian.org/Replicas/debianInstall/

además de manuales de instalación y administración:

http://ftp.mx.debian.org/Replicas/debianInstall/DebianAdministracion/

con distintas versiones para todos los gustos:

[259] Algunas de las razones para instalar GNU/Linux Debian están detalladas en su página Web https://www.debian.org/intro/why_debian.es.html

https://www.debian.org/releases/index.es.html

también lo puedes correr como máquina virtual (VirtualBox):

https://www.osboxes.org/debian/

o en versiones Live:

https://www.debian.org/CD/live/

¿Por que usar Debian GNU/Linux?

https://www.debian.org/intro/why_debian.es.html

Arquitecturas soportadas (y sí, 32 bits seguirá soportada):

https://www.debian.org/releases/lenny/ia64/ch02s01.html.es

Entornos de escritorio soportados:

https://wiki.debian.org/es/DesktopEnvironment

Derivados de Debian:

https://upload.wikimedia.org/wikipedia/commons/6/69/DebianFamilyTree1210.svg

Administrador del Sistema Las labores del administrador de sistemas pueden incluir:

- Agregar, remover o actualizar información de cuentas, reinicio de contraseñas, etc.
- Análisis de registros de sistema e identificar potenciales inconvenientes con los equipos de cómputo de la red local o remota.
- Instalar actualizaciones de sistemas operativos, correcciones y cambios de configuración.
- Instalar y configurar nuevo Hardware y/o Software.
- Responder consultas técnicas y asistencia a usuarios.

- Es responsable de la seguridad (esta es inherente al cargo).
- Es responsable de documentar la configuración del sistema, ya sea para beneficio propio cuando tenga que tomar vacaciones, o para sus sucesores en el cargo.
- Solución de los problemas que reporten los usuarios.
- Afinación del desempeño del sistema.
- Asegurarse de que la infraestructura de red esté iniciada y funcionando (monitorización de servidores y redes).
- Configuración, adición y borrado de archivos del sistema.
- Supervisar de manera directa que los ambientes de desarrollo, pruebas y producción se sincronicen y funcionen sin inconveniente alguno.

En esta sección exploraremos algunas de estas actividades y cómo lograrlo usando Debian GNU/Linux. Además ponemos tu disposición una amplia bibliografía para aprender a usar, administrar y optimizar cada uno de los distintos aspectos de Linux en:

Sistemas Operativos

13.1 Cuentas de Usuario

El sistema operativo Linux, es un sistema multiusuario y multiplataforma, especialmente creado para trabajar estilo red. Es por esta razón, que cada usuario debe tener una cuenta para poder trabajar bajo GNU/Linux.

La tarea de crear cuentas de usuario es hecha por el administrador (Root ó Súper usuario), el cual tiene la capacidad de incluir nuevos usuarios al sistema y asignarle los permisos que crea convenientes.

Gestionar Usuarios en GNU/Linux Un usuario o cuenta de un sistema se identifica de forma única mediante un número denominado UID (número de identificación único). En GNU/Linux podremos encontrar básicamente tres tipos de usuarios:

Usuario root también llamado usuario administrador o superusuario, con UID 0 y que cuenta con privilegios de acceso y modificación sobre todo el sistema. Será el encargado de realizar configuraciones importantes, instalar programas, actualizaciones y todo lo relativo a la gestión de cuentas de usuarios y acceso total a todos los archivos y directorios con independencia de propietarios y permisos.

Usuarios especiales más que usuarios, son cuentas del sistema que se instalan con determinado Software o que ya vienen por defecto en el sistema. Ejemplo de ello son bin, mail, apache, sshd, etc. No son usuarios normales, pero tampoco llegarán a tener todos los permisos de root. Solamente tendrán activadas ciertas funciones de root en función del propósito para el que estén creados, lo anterior para proteger al sistema de posibles formas de vulnerar la seguridad. No tienen contraseñas pues son cuentas que no están diseñadas para iniciar sesiones con ellas -también se les conoce como cuentas de "no inicio de sesión" (nologin)-. Se crean (generalmente) automáticamente al momento de la instalación de GNU/Linux o de la aplicación y generalmente se les asigna un UID entre 1 y 100 (definido en /etc/login.defs).

Usuarios normales estos serán los que utilizamos en nuestro sistema para realizar las típicas tareas diarias. Contaremos con acceso a la terminal de comandos para tareas básicas, contaremos con un directorio de trabajo para cada uno en /home, así como una UID. Tienen solo privilegios completos en su directorio de trabajo o HOME. Por seguridad, es siempre mejor trabajar como un usuario normal en vez del usuario root, y cuando se requiera hacer uso de comandos solo de root, utilizar el comando su. En las distribuciones actuales de Linux se les asigna generalmente un UID superior a 1000.

13.1.1 El Usuario Administrador del Sistema

El usuario root en GNU/Linux es el usuario que tiene acceso administrativo al sistema. Los usuarios normales no tienen este acceso por razones de seguridad.

Cambiar de Usuario en Linux el comando su (Switch User) se utiliza para cambiar de usuario cuando estamos dentro de la consola de Linux, ejemplo:

```
$ su antonio
```

si delante del usuario ponemos - nos abrirá una nuevo Shell con las preferencias del usuario al que cambiemos, por ejemplo:

```
$ su - administracion
```

Por otro lado, si usamos el comando *su* sin usuario, nos permitirá ingresar como el usuario administrador del sistema *root* (por defecto), pidiendo el clave o *Password* de dicho usuario, ejemplo:

```
$ su
```

Uso de Sudo en múltiples sistemas derivados de Debian GNU/Linux no incluye el usuario root (por ejemplo en todos los derivados de Ubuntu). En su lugar, se da acceso administrativo a usuarios individuales, que pueden utilizar la aplicación *sudo* para realizar tareas administrativas. La primera cuenta de usuario que creó en su sistema durante la instalación tendrá, de forma predeterminada, acceso a *sudo*.

Cuando se necesite ejecutar una aplicación que requiere privilegios de administrador, *sudo* le pedirá que escriba su contraseña de usuario normal. Esto asegura que aplicaciones incontroladas no puedan dañar su sistema, y sirve como recordatorio de que está a punto de realizar acciones administrativas que requieren que tenga cuidado.

Para usar *sudo* en la línea de comandos, simplemente escriba *sudo* antes del comando que desea ejecutar, sudo le pedirá su contraseña[260]:

```
$ sudo apt update
```

[260] *Sudo* recordará su contraseña durante un periodo de tiempo (predeterminado a 15 minutos). Esta característica se diseñó para permitir a los usuarios realizar múltiples tareas administrativas sin tener que escribir su contraseña cada vez.

El Shell y el Prompt los programas, tanto los escritos por el usuario como los de sistemas, normalmente se ejecutan con un intérprete de órdenes. El intérprete de órdenes en Linux es un proceso de usuario como cualquier otro y recibe el nombre de Shell (concha o cáscara) porque rodea al núcleo del sistema operativo.

El Shell indica que está listo para aceptar otra orden exhibiendo una señal de espera (*Prompt*) y el usuario teclea una orden en una sola línea. En el Bourne Shell y sus derivados como Bash el *Prompt* que nos permite escribir los diferentes comandos, generalmente termina con el carácter:

- $ para usuario sin privilegios
- # para el administrador, conocido como *root*

Formas de Encontrar Información de Cuentas de Usuario y Detalles de Inicio de Sesión Hay varias formas de mostrar información acerca de los usuarios en un equipo con GNU/Linux, algunas de ellas son:

- id - Nos proporciona información del usuario, uso:

  $ id antonio

- groups - Indica a los grupos a los que pertenece el usuario, uso:

  $ groups antonio

- finger - Si está instalado, proporciona información sobre los usuarios del sistema, uso:

  $ finger antonio

- getent - Permite obtener entradas de la biblioteca NSS de una base de datos específica del sistema, uso:

  $ getent passwd antonio

- /etc/passwd - Nos permite buscar usuarios en el archivo, uso:

  $ grep -i antonio /etc/passwd

- lslogins - Muestra información de todas las cuentas del sistema, la bandera -u despliega la información de usuarios, uso:

  $ lslogins -u

- users - Despliega los nombre de usuario del sistema, uso:

  $ users

- who - Visualiza a los usuarios ingresados en el sistema, uso:

  $ who -u

- w - Visualiza a los usuarios ingresados en el sistema y desde donde lo hicieron, uso:

  $ w

- last - Muestra a todos los usuarios que han ingresados al sistema, uso:

  $ last

si queremos saber los usuario activos en este momento, podemos usar:

  $ last -ap now

- lastlog - Muestra los detalles de los recientes ingresos de todos los usuarios al sistema, uso:

  $ lastlog

- lastb - Muestra los intentos de inicio de sesión incorrectos, uso:

  # lastb

13.1.2 Creación, Modificación y Eliminación de Cuentas

El proceso de creación de cuentas de usuarios en GNU/Linux es sumamente sencillo, pero debe realizarse de manera metódica, para evitar algún error que nos obligue a crear la cuenta nuevamente. Existen dos formas de realizar esta tarea, una de ellas es completamente gráfica y muy intuitiva. El inconveniente de esta es que si el equipo es servidor, por lo general no tendremos entorno gráfico.

La otra forma, es mediante la terminal en línea de comandos. Es un proceso sencillo y muy eficiente, que se puede realizar desde cualquier equipo de la red, como *root* o mediante el comando *su*.

Se puede hacer a través de tres comandos:

- *adduser*: Crear un nuevo usuario con todos los parámetros predeterminados o actualizar la información del nuevo usuario predeterminado.
- *useradd*: Crear un nuevo usuario o actualizar la información predeterminada del nuevo usuario.
- *newusers*: Actualizar y crear nuevos usuarios en batch.

Y mientras se crea un nuevo usuario, se modificarán los cuatro archivos siguientes:

- */etc/passwd*: Los detalles del usuario se actualizarán en este archivo.
- */etc/shadow*: La información de la contraseña del usuario se actualizará en este archivo.
- */etc/group*: Los detalles del grupo del nuevo usuario se actualizarán en este archivo.
- */etc/gshadow*: La información de la contraseña del grupo del nuevo usuario se actualizará en este archivo.

Archivos de inicialización generales:

- /etc/profile Contiene la configuración del perfil de arranque del *login*. Se ejecuta cada vez que un usuario ingresa al sistema.

- /etc/bashrc Contiene funciones de configuración como el *umask*, *PS1* del Prompt. Se ejecuta cada vez que se invoca una Shell.
- /etc/motd Mensaje del día para todos los usuarios, que será mostrado al inicio de la sesión.
- /etc/default/useradd Configuración de los valores predeterminados al crear un usuario, como ser el directorio personal, el grupo principal.
- /etc/login.defs Configuración de los valores predeterminados al crear un usuario,como el número de usuario y valores de la contraseña.
- /etc/issue Contiene el banner que se mostrará en el momento del *login* local.
- /etc/issue.net Contiene el banner que se mostrará en el momento del *login* remoto, ejemplo por ssh.

Contraseñas las contraseñas o Passwords protegen la información que contienen los dispositivos y cuentas de los usuarios. No obstante, ante la cantidad de claves y combinaciones que cotidianamente se deben utilizar, la mayoría de las personas opta por contraseñas fáciles de recordar por la comodidad que esto implica, o bien, por la falta de conocimiento de lo fácil que puede ser para un ciberdelincuente obtenerlas. Para asegurar la efectividad de las contraseñas[261] y evitar el robo de éstas, es recomendable poner en práctica las siguientes acciones:

- Al generar las contraseñas de los dispositivos y cuentas se deben utilizar claves largas (mínimo 13 caracteres[262]) y únicas para cada caso, evitando utilizar la misma contraseña para diferentes dispositivos o cuentas.

[261] Para conocer la seguridad de una clave, podemos checarla en: https://howsecureismypassword.net/

[262] Solo para tener una idea del tiempo necesario para encontrar la clave de 13 caracteres usando fuerza bruta en un solo procesador: Sólo números (1 hr), mezclar letras mayúsculas y minúsculas (600 años), mezclar números, letras mayúsculas y minúsculas (6 mil años), mezclar números, símbolos, letras mayúsculas y minúsculas (5 mil años).

Si se aumentan más procesadores o múltiples máquinas los tiempos se acortarán de forma drástica.

- Se deben evitar las combinaciones sencillas como fechas de nacimiento, secuencias consecutivas, repeticiones de un mismo dígito o palabras simples como "password" o "contraseña".
- La mayor longitud de la contraseña, así como la incorporación de mayúsculas, minúsculas, números y símbolos (#$,.-_%&*!?), contribuyen a que ésta sea más segura y difícil de vulnerar.
- Se debe evitar escribir contraseñas en papeles o tener archivos con esa información que sean fácilmente accesibles para otros.
- Es importante no facilitar a nadie, aunque así lo solicite, por ningún medio, contraseñas y/o códigos para el inicio de sesión.
- Es recomendable cambiar con frecuencia las contraseñas a efecto de evitar accesos no autorizados.

Creando, Modificando y Borrando una Cuenta de Usuario Lo primero que debemos hacer, es ingresar como root desde la consola de GNU/Linux. Para hacer esto, una vez abierta la consola, tecleamos el siguiente comando:

```
$ su
```

seguidamente, la consola nos pedirá la contraseña, la ingresamos y activaremos los privilegios de root en el terminal. Notemos que el Prompt del sistema es el símbolo $, lo que nos indica que estamos ingresando como usuario estándar. Cuando activemos los privilegios de *su*, el Prompt cambiará al símbolo #.

Crear Cuenta de Usuario una vez activados los privilegios de root, procedemos a crear una nueva cuenta de usuario, para ello usamos:

```
# adduser antonio
```

nos pedirá la contraseña y su confirmación además de información sobre el usuario. De esta forma crea un nuevo usuario con el nombre de antonio, con todas las opciones por defecto, tales como ubicación de su directorio, duración de la cuenta de usuario, Shell a utilizar o grupos en los que va a ser incluido (el identificador de usuario UID y el identificador de grupo GID

serán iguales). Este comando acepta las mismas opciones que el comando *usermod*.

Para conocer la información de un usuario del sistema, usamos:

```
$ id antonio
```

Cambiar Contraseña Usuario en cualquier momento después de creada la cuenta, podemos cambiar la contraseña del usuario, usando:

```
# passwd antonio
```

luego, el sistema nos pedirá que ingresemos la contraseña dos veces para poder verificar si el procedimiento se ha realizado correctamente. Para conocer el estado del Password de mi cuenta puedo usar:

```
$ passwd -S antonio
```

si deseamos conocer el estado del Password de todas las cuentas del sistema usamos:

```
# passwd -Sa
```

El usuario root es el único que puede indicar el cambio o asignación de contraseñas de cualquier usuario. Usuarios normales pueden cambiar su contraseña en cualquier momento con tan solo invocar passwd sin argumentos, y podrá de esta manera cambiar la contraseña cuantas veces lo requiera.

Podemos forzar el cambio de clave de acceso de un usuario en su próximo ingreso al sistema, mediante:

```
# passwd -r usuario
```

Bloquear y Desbloquear un Usuario otra tarea que se puede realizar al gestionar usuarios en GNU/Linux es bloquear y desbloquear una cuenta de usuario. Para bloquear una cuenta, necesitamos invocar passwd con la opción *-l*.

```
# passwd -l antonio
```

la opción -u con passwd desbloquea una cuenta:

```
# passwd -u antonio
```

Borrar Cuenta de un Usuario también, tendremos la capacidad de eliminar cuentas de usuario con el comando:

```
# userdel antonio
```

sin opciones elimina la cuenta del usuario de */etc/passwd* y de */etc/shadow*, pero no elimina su directorio de trabajo ni archivos contenidos en el mismo, esta es la mejor opción, ya que elimina la cuenta pero no la información de la misma.

```
# userdel -r antonio
```

al igual que lo anterior elimina la cuenta totalmente, pero con la opción *-r* además elimina su directorio de trabajo y archivos y directorios contenidos en el mismo, así como su buzón de correo, si es que estuvieran configuradas las opciones de correo. La cuenta no se podrá eliminar si el usuario está en el sistema al momento de ejecutar el comando.

```
# userdel -f antonio
```

la opción -f es igual que la opción *-r*, elimina todo lo del usuario, cuenta, directorios y archivos del usuario, pero además lo hace sin importar si el usuario está actualmente en el sistema trabajando. Es una opción muy radical, además de que podría causar inestabilidad en el sistema, así que hay que usarla solo en casos muy extremos.

Crear Grupos de Trabajo[263] podemos crear grupos completos de usuario, para ello hacemos:

```
# groupadd nombre_grupo
```

y podemos cambiar el nombre de un grupo mediante:

```
# groupmod -n grupo_nuevo grupo_antiguo
```

si necesitamos crear un grupo del sistema, usamos:

```
# groupadd -r nombre_grupo
```

[263] Es posible crear, manipular y borrar grupos mediante el comando *usermod*.

Agregar y Remover usuarios a un Grupo para conocer a los grupos que pertenece un usuario, usamos:

```
$ groups usuario
```

podemos agregar un usuario a un grupo, mediante:

```
# gpasswd -a usuario nombre_grupo
```

y podemos borrar un usuario de un grupo, usando:

```
# gpasswd -d usuario nombre_grupo
```

podemos conocer los grupos a los que pertenece un usuario, usando:

```
$ groups usuario
```

si queremos conocer la lista de todos los grupos existentes, usamos:

```
$ getent group
```

Borrar Grupos de Trabajo podemos borrar grupos completos de usuarios, mediante:

```
# groudel nombre_grupo
```

Access Control List también están disponibles los comandos *getfacl* y *setfacl* pertenecientes a *Access Control List* (ACLs) que es un método más flexible para establecer permisos a usuarios y grupos sobre archivos y directorios. Por ejemplo podemos ver los permisos y grupos a los que pertenece un archivo o directorio mediante:

```
$ getfacl archivo
```

y cambiarlos usando:

```
$ setfacl -m g:invitado:7 archivo
```

Archivos de configuración Los usuarios normales y root en sus directorios de inicio tienen varios archivos que comienzan con "." es decir están ocultos. Varían mucho dependiendo de la distribución de Linux que se tenga, pero seguramente se encontrarán los siguientes o similares:

```
# ls -la

drwx– 2 alumno alumno 4096 jul 9 09:54 .
drwxr-xr-x 7 root root 4096 jul 9 09:54 ..
-rw-r–r– 1 alumno alumno 24 jul 9 09:54 .bash_logout
-rw-r–r– 1 alumno alumno 191 jul 9 09:54 .bash_profile
-rw-r–r– 1 alumno alumno 124 jul 9 09:54 .bashrc
```

.bash_profile aquí podremos indicar alias, variables, configuración del entorno, etc. que deseamos iniciar al principio de la sesión.

.bash_logout aquí podremos indicar acciones, programas, Scripts, etc., que deseemos ejecutar al salirnos de la sesión.

.bashrc es igual que .bash_profile, se ejecuta al principio de la sesión, tradicionalmente en este archivo se indican los programas o Scripts a ejecutar, a diferencia de *.bash_profile* que configura el entorno.

Si deseamos configurar archivos de inicio o de salida de la sesión gráfica entonces, en este caso, hay que buscar en el menú del ambiente gráfico algún programa gráfico que permita manipular que programas se deben arrancar al iniciar la sesión en modo gráfico. En la mayoría de las distribuciones existe un programa llamado "sesiones" o "sessions", generalmente está ubicado dentro del menú de preferencias. En este programa es posible establecer programas o Scripts que arranquen junto con el ambiente gráfico, sería equivalente a manipular 'bashrc'.

Además GNU/Linux permite que el usuario decida que tipo de entorno Xwindow a utilizar, ya sea algún entorno de escritorio como *KDE* o *Gnome* o algún manejador de ventanas como *Xfce* o *Twm*. Dentro del Home del usuario, se creará un directorio o archivo escondido "." , por ejemplo '.gnome'

o '.kde' donde vendrá la configuración personalizada del usuario para ese entorno. Dentro de este directorio suele haber varios directorios y archivos de configuración. Estos son sumamente variados dependiendo de la distribución y del entorno. No es recomendable modificar manualmente (aunque es perfectamente posible) estos archivos, es mucho más sencillo modificar vía la interfaz gráfica, que permiten cambiar el fondo, protector de pantalla, estilos de ventanas, tamaños de letras, etc.

Crear Cuenta de Usuario con useradd Para hacerlo, escribimos el siguiente comando desde el terminal de GNU/Linux:

```
# useradd nombre_usurio
```

donde: nombre_usuario es el nombre del usuario al cual le estamos creando la cuenta. Una vez creado el nombre de usuario, su cuenta estará activa.

Las opciones más comunes o importantes del comando *useradd* son las siguientes:

- -c añade un comentario al momento de crear al usuario, campo 5 de /etc/passwd
- -d directorio de trabajo o home del usuario, campo 6 de /etc/passwd
- -e fecha de expiración de la cuenta, formato AAAA-MM-DD, campo 8 de /etc/shadow
- -g número de grupo principal del usuario (GID), campo 4 de /etc/passwd
- -G otros grupos a los que puede pertenecer el usuario, separados por comas.
- -r crea una cuenta del sistema o especial, su UID será menor al definido en /etc/login.defs en la variable UID_MIN, además no se crea el directorio de inicio.
- -s Shell por defecto del usuario cuando ingrese al sistema. Si no se especifica, Bash, es el que queda establecido.
- -u UID del usuario, si no se indica esta opción, automáticamente se establece el siguiente número disponible a partir del último usuario creado.

El siguiente paso es crearle una contraseña, para hacerlo sólo debemos teclear el comando:

```
# passwd nombre_usuario
```

Luego, el sistema nos pedirá que ingresemos la contraseña dos veces para poder verificar si el procedimiento se ha realizado correctamente.

Si por ejemplo queremos agregar un usuario a un grupo primario y un grupo suplementario usaremos la opción -g y -G , respectivamente, ejemplo:

```
# useradd -g "programadores" -G "frontend" antonio
```

Podemos ver las características de la cuenta usando:

```
# chage -l antonio
```

y

```
$ id antonio
```

Modificar Usuarios usermod permite modificar o actualizar un usuario o cuenta ya existente. Sus opciones más comunes son las siguientes:

- -c añade o modifica el comentario, campo 5 de /etc/passwd
- -d modifica el directorio de trabajo o home del usuario, campo 6 de /etc/passwd
- -e cambia o establece la fecha de expiración de la cuenta, formato AAAA-MM-DD, campo 8 de /etc/shadow
- -g cambia el número de grupo principal del usuario (GID), campo 4 de /etc/passwd
- -G establece otros grupos a los que puede pertenecer el usuario, separados por comas.
- -l cambia el login o nombre del usuario, campo 1 de /etc/passwd y de /etc/shadow

- -L bloque la cuenta del usuario, no permitiéndole que ingrese al sistema. No borra ni cambia nada del usuario, solo lo deshabilita.
- -s cambia el Shell por defecto del usuario cuando ingrese al sistema.
- -u cambia el UID del usuario.
- -U desbloquea una cuenta previamente bloqueada con la opción -L.

Si quiseramos cambiar el nombre de usuario de 'tony' a 'antonio':

```
# usermod -l tony antonio
```

casi seguro también cambiará el nombre del directorio de inicio o HOME en /home, pero si no fuera así, entonces:

```
# usermod -d /home/tony antonio
```

otros cambios o modificaciones en la misma cuenta:

```
# usermod -c "supervisor de area" -s /bin/ksh -g 1505 antonio
```

lo anterior modifica el comentario de la cuenta, su Shell por defecto que ahora será Korn Shell y su grupo principal de usuario quedó establecido al GID 1505 y todo esto se aplicó al usuario 'antonio' que como se observa debe ser el último argumento del comando.

El usuario 'antonio' salió de vacaciones y nos aseguramos de que nadie use su cuenta:

```
# usermod -L antonio
```

Podemos ver las características de la cuenta usando:

```
# chage -l antonio
```

y

```
$ id antonio
```

/etc/passwd Cualquiera que sea el tipo de usuario, todas las cuentas se encuentran definidas en el archivo de configuración 'passwd', ubicado dentro del directorio /etc. Este archivo es de texto tipo ASCII, se crea al momento de la instalación con el usuario root y las cuentas especiales, más las cuentas de usuarios normales que se hayan indicado al momento de la instalación.

El archivo */etc/passwd* contiene una línea para cada usuario del sistema, y puede ser visualizada usando:

```
$ cat /etc/passwd
```

o podemos usar el comando *column* para mostrarla algo más legible:

```
$ column -s ":" -t /etc/passwd
```

Cada línea contiene algo similar a:

```
root:x:0:0:root:/root:/bin/Bash
antonio:x:501:500:Antonio Carrillo:/home/antonio:/bin/Bash
```

La información de cada usuario está dividida en 7 campos delimitados cada uno por ':' dos puntos.

/etc/passwd

- Campo 1 Es el nombre del usuario, identificador de inicio de sesión (login). Tiene que ser único.
- Campo 2 La 'x' indica la contraseña cifrada del usuario, además también indica que se está haciendo uso del archivo /etc/shadow, si no se hace uso de este archivo, este campo se vería algo así como: 'ghy675gjuXCc12r5gt78uuu6R'.
- Campo 3 Número de identificación del usuario (UID). Tiene que ser único. 0 para root, generalmente las cuentas o usuarios especiales se numeran del 1 al 100 y las de usuario normal del 1000 en adelante.
- Campo 4 Numeración de identificación del grupo (GID). El que aparece es el número de grupo principal del usuario, pero puede pertenecer a otros, esto se configura en /etc/groups.

- Campo 5 Comentarios o el nombre completo del usuario.
- Campo 6 Directorio de trabajo (Home) donde se sitúa al usuario después del inicio de sesión.
- Campo 7 Shell que va a utilizar el usuario de forma predeterminada.

Podemos ver las características de la cuenta usando:

```
# chage -l antonio
```

y

```
$ id antonio
```

/etc/shadow Anteriormente (en sistemas Unix) las contraseñas cifradas se almacenaban en el mismo */etc/passwd*. El problema es que 'passwd' es un archivo que puede ser leído por cualquier usuario del sistema, aunque solo puede ser modificado por root. Con cualquier computadora potente de hoy en día, un buen programa de descifrado de contraseñas y paciencia es posible "crackear" contraseñas débiles (por eso la conveniencia de cambiar periódicamente la contraseña de root y de otras cuentas importantes). El archivo 'shadow', resuelve el problema ya que solo puede ser leído por *root*. Considérese a 'shadow' como una extensión de 'passwd' ya que no solo almacena la contraseña cifrada, sino que tiene otros campos de control de contraseñas.

El archivo /etc/shadow contiene una línea para cada usuario, similar a las siguientes:

```
root:ghy675gjuXCc12r5gt78uuu6R:10568:0:99999:7:7:-1::
antonio:rfgf886DG778sDFFDRRu78asd:10568:0:-1:9:-1:-1::
```

La información de cada usuario está dividida en 9 campos delimitados cada uno por ':' dos puntos.

/etc/shadow

- Campo 1 Nombre de la cuenta del usuario.
- Campo 2 Contraseña cifrada, un '*' indica cuenta de 'nologin'.
- Campo 3 Días transcurridos desde el 1/ene/1970 hasta la fecha en que la contraseña fue cambiada por última vez.
- Campo 4 Número de días que deben transcurrir hasta que la contraseña se pueda volver a cambiar.
- Campo 5 Número de días tras los cuales hay que cambiar la contraseña. (-1 significa nunca). A partir de este dato se obtiene la fecha de expiración de la contraseña.
- Campo 6 Número de días antes de la expiración de la contraseña en que se le avisará al usuario al inicio de la sesión.
- Campo 7 Días después de la expiración en que la contraseña se inhabilitará, si es que no se cambió.
- Campo 8 Fecha de caducidad de la cuenta. Se expresa en días transcurridos desde el 1/Enero/1970 (epoch).
- Campo 9 Reservado.

/etc/group Este archivo guarda la relación de los grupos a los que pertenecen los usuarios del sistema, contiene una línea para cada usuario con tres o cuatro campos por usuario:

```
root:x:0:root
test:x:501:
antonio:x:502:ventas,supervisores,produccion
alumno:x:503:ventas,pedro
```

- Campo 1 indica el usuario.
- Campo 2 'x' indica la contraseña del grupo, que no existe, si hubiera se mostraría un 'hash' cifrado.

- Campo 3 es el Group ID (GID) o identificación del grupo.
- Campo 4 es opcional e indica la lista de grupos a los que pertenece el usuario

Actualmente al crear al usuario con useradd se crea también automáticamente su grupo principal de trabajo GID, con el mismo nombre del usuario. Es decir, si se añade el usuario 'antonio' también se crea el /etc/group el grupo 'antonio'.

Podemos ver las características de la cuenta usando:

```
# chage -l antonio
```

y

```
$ id antonio
```

/etc/login.defs En el archivo de configuración */etc/login.defs* están definidas las variables que controlan los aspectos de la creación de usuarios y de los campos de shadow usadas por defecto. Algunos de los aspectos que controlan estas variables son:

- Número máximo de días que una contraseña es válida PASS_MAX_DAYS
- El número mínimo de caracteres en la contraseña PASS_MIN_LEN
- Valor mínimo para usuarios normales cuando se usa useradd UID_MIN
- El valor umask por defecto UMASK
- Si el comando useradd debe crear el directorio home por defecto CREATE_HOME

Basta con leer este archivo para conocer el resto de las variables que son autodeScriptivas y ajustarlas al gusto.

Comandos de administración y control de usuarios Existen varios comandos más que se usan muy poco en la administración de usuarios, que sin embargo permiten administrar aún más a detalle a tus usuarios de Linux. Algunos de estos comandos permiten hacer lo mismo que los comandos previamente vistos, solo que de otra manera 'chage', 'id', ' gpasswd', 'groupmod', 'groups', 'shadow', 'chfn', 'groupmod', 'grpck', 'pwck', 'vigr', 'vipw'; y otros como 'chpasswd' y 'newusers' resultan muy útiles y prácticos cuando de dar de alta a múltiples usuarios se trata y si se necesita trabajar con control de accesos los comando 'setfacl' y 'getfacl' son una mejor opción.

13.1.3 Configurar Usuarios con Restricciones

Si por algún motivo debemos darle acceso a nuestro equipo a un usuario (ya sea físico o mediante SSH), siempre debemos velar por la seguridad y estabilidad del equipo. Entonces, ¿cómo crear un usuario con privilegios extremadamente limitados, tanto así que no pueda ni siquiera salir de su jaula (home)?.

Una primera opción es el uso del Shell restringido, es un Shell (en Bash: *rbash* o *bash -r*) que restringe algunas de las capacidades disponibles para una sesión de usuario interactiva, o para un Script de Shell, que se ejecuta dentro de él. Su objetivo es proporcionar una capa adicional de seguridad, pero es insuficiente para permitir la ejecución de Software completamente no confiable. Una operación en modo restringido se encuentra en el Shell Bourne original y su posterior contraparte Bash, y en el Shell Korn. En algunos casos, se utiliza un Shell restringido junto con una cárcel *chroot*, en un intento adicional de limitar el acceso al sistema en su conjunto (como al usar el paquete *jailkit*).

Restringir al Usuario con Rbash Un modo restringido se usa para configurar un entorno más controlado que el estándar, se usa el Shell instalado de Bash mediante el uso de (*rbash* o *bash -r*). Su comportamiento y forma de operar es idéntico a Bash, con la excepción de lo que no está permitido:

- Salir del directorio Home con el comando cd.
- No se permite modificar los valores de SHELL, PATH, ENV o BASH_ENV.
- Comandos, secuencias o nombres que contengan «/», como carácter o argumento.

- Archivos que contengan una barra como argumento para la opción -p del propio *hash*.
- Importar definiciones de funciones desde el entorno de inicio de Shell.
- Analizar el valor de SHELLOPTS desde el entorno de inicio Shell.
- Redirigir salidas usando los operadores de redirección >, > |, <>, > &, &> y >>.
- Usar exec para reemplazar el shell con otro comando.
- Agregar o eliminar comandos integrados con las opciones -f y -d, del comando *enable*.
- Uso del comando Enable Builtin para habilitar funciones Shell, que están deshabilitadas.
- Desactivar cualquier modo de salir del modo restringido.
- Desactivar las restricciones de 'set +r' o 'set +o'
- Alguna más de menor importancia.

Uso siempre podemos en la consola lanzar un Bash restringido y en el correr los programas que necesitamos, mediante:

```
$ bash -r
```

y si lo que deseamos es que el usuario siempre tenga estas restricciones, podemos editar el archivo: */etc/passwd* y cambiar en el usuario correspondiente el Shell */bin/bash* por */bin/rbash*, por ejemplo:

```
acl:x:1006:1001:acl,001„:/home/acl:/bin/bash
```

a:

```
acl:x:1006:1001:acl,001„:/home/acl:/bin/rbash
```

Si bien, esta forma de restringir un usuario funciona, hay mecanismos más robustos para restringir a los usuarios y evitar daños a nuestro equipo, para ello se han desarrollado diversos programas llamados Sanbox.

¿Qué es un Sanbox? Un Sandbox es un mecanismo de seguridad para disponer de un entorno aislado del resto del sistema operativo. Todos los programas que se ejecutan dentro de un Sandbox lo hacen de forma controlada mediante los siguientes aspectos:

- Se les asigna un espacio en disco. Estos programas no podrán acceder a ningún espacio del disco que no les haya sido asignado previamente.
- Podemos hacer que nuestros programas se ejecuten en un sistema de archivos temporal (tmpfs) para aislarlos del resto del sistema operativo.
- También se les asigna un espacio en memoria. Los programas no podrán acceder a otras partes de la memoria que no les hayan sido asignadas.
- Les podemos dar o restringir la capacidad para acceder y consultar dispositivos de almacenamiento externos.
- Les restringimos la capacidad para que puedan inspeccionar la máquina anfitrión.
- Podemos restringir el acceso de los programas a la red, al servidor de las X, al servidor de sonido, etc.
- Podemos limitar el ancho de banda que usa un determinado programa.

Consecuentemente podemos afirmar que los programas se ejecutan en un entorno controlado y separado del sistema operativo. Por lo tanto el hecho de ejecutar un programa en un Sandbox es un ejemplo específico de virtualización.

¿Qué utilidad tiene un Sanbox? Nos podemos hacer la idea que su principal utilidad es ejecutar programas de forma segura y sin peligro de comprometer el resto del sistema operativo. Algunos ejemplos de los casos en que podemos utilizar un Sandbox son los siguientes:

- Instalar y usar de forma segura programas que no son fiables y/o pueden contener virus y/o código malicioso.
- Usar programas que tienen el potencial de comprometer la seguridad de nuestro sistema operativo como por ejemplo un navegador Web, Skype, un lector de pdf, un Keygen descargado de Internet, etc.

- Crear un entorno de pruebas para ejecutar programas o un código que acabamos de crear.
- Para descargar Torrents.
- Etc.

Por lo tanto podemos ver que un Sandbox no es de uso exclusivo para programadores o expertos informáticos. También es útil para todo tipo de usuarios. Algunos de los navegadores Web que usamos disponen de su propio Sandbox. No obstante no es mala idea usar dos capas de protección porque los navegadores Web son altamente susceptibles de sufrir problemas de seguridad.

Restringir al Usuario con Firejail Es un programa que utiliza SUID (Set user ID), los espacios de nombres del Kernel Linux (Linux Namespaces) y Seccomp-bpf para crear un Sandbox. De esta forma podremos ejecutar cualquier tipo de programa de forma segura en un entorno controlado.

Podemos usar Firejail en multitud de programas y aplicaciones. Algunos ejemplos son los siguientes:

- En aplicaciones gráficas como por ejemplo un navegador Web, un programa para descargar Torrent, en Skype, en un reproductor de vídeo como VLC, etc.
- En programas que se ejecutan en un servidor y a través de la terminal como por ejemplo el servidor Web Apache.

Por experiencia puedo afirmar lo siguiente:

- Firejail es ligero. La velocidad de ejecución de los programas dentro del Sandbox es la misma que fuera del Sandbox.
- Al arrancar el ordenador no se cargan procesos adicionales en memoria. Por lo tanto podemos instalar Firejail con total tranquilidad porque únicamente consumirá recursos en el momento de generar y cargar el programa en el Sandbox.
- El programa prácticamente no tiene dependencias. Por lo tanto la cantidad de paquetes que instalaremos será mínima.
- Su uso es extremadamente sencillo. Incluso un usuario básico puede usar este Sandbox.

Instalar el Sanbox Firejail tenemos que ejecutar el siguiente comando en la terminal:

```
# apt install firejail
```

Si además queremos disponer de una interfaz gráfica para gestionar nuestro Sandbox ejecutaremos el siguiente comando en la terminal:

```
# apt install firetools
```

El uso básico para correr cualquier aplicación (por ejemplo el navegador Chrome) dentro Sanbox Firejail es:

```
$ firejail chrome
```

y si lo que deseamos es que el usuario siempre corra dentro del Sanbox Firejail, podemos editar el archivo: */etc/passwd* y cambiar en el usuario correspondiente el Shell */bin/bash* por */usr/bin/firejail*, por ejemplo:

```
acl:x:1006:1001:acl,001„:/home/acl:/bin/bash
```

a:

```
acl:x:1006:1001:acl,001„:/home/acl:/usr/bin/firejail
```

Por poner algunos ejemplos (en creciente nivel de privacidad) para Firefox, podemos usar:

```
$ firejail –alluser firefox &
$ firejail firefox &
$ firejail –dns=8.8.8.8 –dns=8.8.4.4 firefox &
$ firejail –net=eth0 –ip=10.10.20.30 firefox &
$ firejail –net=eth0 –ip=10.10.20.30 –dns=8.8.8.8 firefox &
$ firejail –net=eth0 –mac=00:11:22:33:44:55 firefox &
$ firejail –disable-mnt firefox &
$ firejail –timeout=01:30:00 firefox &
$ firejail –private=~/Temp –noprofile firefox &
$ firejail –read-only=~/.mozilla firefox &
$ firejail –private firefox &
$ firefox –appamor firefox &
```

otro ejemplo para correr una sesión Bash, podemos usar:

```
$ firejail bash
$ firejail –private bash
$ firejail –private-bin=sh,bash,python*
$ firejail –blacklist=~/dir[1234]
$ firejail –read-only=~/dir[1-4]
```

Existen decenas de opciones que podemos usar en Firejail, para conocerlas todas es recomendable consultar:

```
$ man firejail
```

Firejail incluye perfiles de configuración predeterminados para multitud de programas en GNU/Linux. En estos perfiles de configuración es donde se definen las propiedades para aislar el programa del resto del sistema operativo. En estos ficheros de configuración podemos definir parámetros como por ejemplo:

- Las carpetas de nuestro sistema de archivos a las que tendrá acceso el programa que se ejecuta en el Sandbox.
- Los permisos que tendrá el programa sobre ciertos directorios de nuestro sistema operativo.
- Si los programas tienen acceso al servidor de sonido.
- Deshabilitar la aceleración 3D de un programa en concreto.
- Usar unos DNS específicos para un programa usado dentro del Sandbox.
- Determinar si el programa tendrá acceso a internet.
- Etc.

Para ver la totalidad de perfiles disponibles podemos ejecutar el siguiente comando en la terminal:

```
$ ls /etc/firejail/
```

Modificar los Perfiles de Configuración Estándar si queremos modificar alguna característica de un perfil estándar tenemos que acceder a su código y modificarla. A modo de ejemplo, para modificar el código del perfil de Firefox deberíamos ejecutar el comando:

```
# nano /etc/firejail/firefox.profile
```

Una vez abierto el editor de texto deberemos modificar los parámetros pertinentes. Para conocer la totalidad de parámetros existentes y su funcionalidad les recomiendo que abran un terminal y ejecuten el siguiente comando:

```
$ man 5 firejail-profile
```

Después de ejecutar el comando obtendrán una explicación detallada de todos los parámetros de configuración que podemos usar con Firejail.

Crear Perfiles de Programas que no Están Disponibles se dispone de perfiles de configuración para más de 300 programas. No obstante, puede darse el caso que tengamos que ejecutar un programa que no no disponga de ningún perfil. En tal caso se usará un perfil estándar. Si el perfil estándar no es de nuestro agrado podemos crear/modificar un perfil de configuración para el programa en cuestión.

Ejecutar un Programa Dentro del Sandbox Firejail nos facilita la ejecución de una gran variedad de programas gracias a los perfiles de configuración que hemos mencionado anteriormente. Para abrir un programa en el Sandbox tan solo tenemos que abrir una terminal y teclear *firejail* seguido del comando que usaríamos para ejecutar el programa desde la terminal. Por lo tanto, para ejecutar Chrome en el Sandbox ejecutaríamos el siguiente comando en la terminal:

```
$ firejail chrome
```

al ejecutar el comando arrancará Chrome dentro del Sandbox. Al no indicar ningún tipo de parámetro se iniciará con la configuración predeterminada de Chrome.

Es posible iniciar un programa añadiendo/modificando algunas de las configuraciones que figuran en el perfil de configuración. Para ello ejecutaremos el comando firejail seguido las opciones para modificar provisionalmente el perfil del programa más el comando para ejecutar el programa en la terminal. Por lo tanto, si queremos ejecutar VLC sin que tenga acceso a internet ejecutaremos el siguiente comando:

```
$ firejail –net=none vlc
```

Si queremos que VLC siempre arranque de este modo podríamos realizar las siguientes modificaciones. Primero creamos la carpeta que alojará los ficheros que modificaran la configuración estándar de un programa determinado. Para ello ejecutamos el siguiente comando en la terminal:

```
$ mkdir ~/.config/firejail/
```

seguidamente creamos el fichero que modificará la configuración predeterminada de VLC ejecutando el siguiente comando en la terminal:

```
$ nano ~/.config/firejail/vlc.profile
```

una vez se abra el editor de textos pegamos el siguiente código:

```
include /etc/firejail/vlc.profile
net none
```

La primera línea contiene la palabra include seguida de la ruta que contiene el perfil estándar que queremos modificar. La segunda línea contiene los nuevos parámetros de configuración que queremos aplicar. Una vez realizados los cambios los guardamos y cerramos el fichero.

La próxima vez que arranquemos VLC en el Sandbox, no tendrá acceso a internet. Por lo tanto, VLC únicamente será útil para reproducir ficheros de vídeo y sonido locales.

Ejecutar un Programa en Modo Privado Firejail nos permite ejecutar programas en modo privado. Este modo tiene las siguientes particularidades:

- Es la opción que ofrece más seguridad para ejecutar un programa o para acceder a nuestras cuentas bancarias.
- El modo privado utiliza la configuración estándar de un programa. Por lo tanto, si abrimos Google Chrome en modo privado dispondremos de la configuración estándar de Chrome sin ninguna extensión instalada. Esto puede resultar útil porque cuantas más extensiones usemos y más modifiquemos la configuración, más inseguro estamos haciendo nuestro navegador.
- El programa se ejecuta en un sistema de ficheros temporal (tmpfs). Por lo tanto, absolutamente todos los ficheros que se almacenen en el sistema de archivos temporal serán borrados al cerrar nuestro Sandbox.
- Este modo permite ejecutar programas sin necesidad de modificar ningún archivo en nuestro disco duro.

Para ejecutar el navegador Google Chrome en modo privado y forzando que se utilicen por defecto los dns de Google ejecutaremos el siguiente comando en la terminal:

$ firejail –private –dns=8.8.8.8 –dns=8.8.4.4 google-chrome

Este ejemplo que acabamos de ver es una buena solución en el caso que queramos acceder a nuestras cuentas bancarias de forma segura a través de internet.

Hacer que los Programas Dentro del Sandbox Tengan su Propia Red Mediante Macvlan existe la posibilidad que los programas ejecutados en el sandbox tengan una IP interna distinta a la de nuestro ordenador, un Firewall propio, su propia tabla ARP, etc.

Para ello tenemos que ejecutar un comando del siguiente estilo:

$ firejail –net=nombre_interfaz_de_red nombre_del_programa

Por lo tanto, si nuestra interfaz de red es eth0 y queremos ejecutar Firefox con una IP interna distinta a la de nuestro ordenador ejecutaremos el siguiente comando:

```
$ firejail –net=eth0 firefox
```

si además queremos definir una IP concreta podemos ejecutar el siguiente comando:

```
$ firejail –net=eth0 –ip=192.168.1.80 firefox
```

Nota: Esta opción únicamente funciona para redes cableadas. Actualmente no está soportado en redes inalámbricas.

Si es necesario, también podemos limitar el ancho de banda que consumen los programas que se ejecutan dentro del Sandbox. Para ello deberemos seguir las siguientes instrucciones:

Primero crearemos una Sandbox con nombre navegador y una interfaz de red propia para arrancar Firefox. Para ello ejecutaremos el siguiente comando en la terminal:

```
$ firejail –name=navegador –net=eth0 firefox
```

a continuación abriremos otra terminal y limitaremos el ancho de banda del Sandbox navegador ejecutando el siguiente comando en la terminal:

```
$ firejail –bandwidth=navegador set eth0 80 20
```

De este modo estamos aplicando la siguiente limitación:

- La velocidad máxima de descarga de Firefox será de 80 kilobytes.
- La velocidad de subida máxima será de 20 kilobytes por segundo.

Si en algún momentos queremos eliminar los límites de velocidad impuestos, tan solo tenemos que ejecutar el siguiente comando:

```
$ firejail –bandwidth=navegador clear eth0
```

Comprobar qué Programas se están ejecutando dentro del Sandbox tan solo tenemos que ejecutar el siguiente comando:

```
$ firejail –list
```

también podemos usar la opción:

```
$ firejail –tree
```

Ver los Recursos que está Consumiendo el Sandbox para monitorizar el uso de recursos de los programas ejecutándose en el Sanbox podemos ejecutar el siguiente comando:

```
$ firejail –top
```

Acceder al Sandbox para Administrarlo si tenemos necesidad de acceder dentro de un Sandbox para modificar ciertos parámetros, como por ejemplo comprobar una configuración de red. Tan solo tenemos que ejecutar el comando *firejail -join=* seguido del número de PID del programa que se ejecuta en el Sandbox (supongamos PID 5394).

```
$ sudo firejail –join=5394
```

una vez dentro del sandbox podremos realizar las tareas de administración que consideremos oportunas.

Usar el Sandbox de Forma Predeterminada para usar Firejail de forma predeterminada en todas las aplicaciones que dispongan de un perfil tenemos que ejecutar el siguiente comando en la terminal:

```
# firecfg
```

este comando creará enlaces simbólicos de todas las aplicaciones que dispongan de un perfil de Firejail en la ubicación */usr/local/bin* y apuntarán a */usr/bin/firejai*l. De este modo, cada vez que arranquemos un programa que disponga de un perfil arrancará dentro del sandbox

Para hacer que los programas vuelvan arrancar fuera del Sandbox se deberá acceder en la ubicación */usr/local/bin* y borrar la totalidad de enlaces simbólicos que se crearon con el comando *firecfg*.

Podemos usar Firejail de forma predeterminada en una sola aplicación creando un enlace simbólico de forma manual. Para ello ejecutamos un comando del siguiente tipo:

```
# ln -s /usr/bin/firejail /usr/local/bin/nombre_programa
```

por lo tanto si queremos que Firefox arranque de forma predeterminada en el Sandbox deberemos ejecutar el siguiente comando en la terminal:

```
# ln -s /usr/bin/firejail /usr/local/bin/firefox
```

Si algún día queremos deshacer esta configuración, tan solo tenemos que acceder en la ubicación */usr/local/bin/* y borrar el enlace simbólico de Firefox.

13.2 Información del Equipo y Sistema Operativo

Antes de proceder a realizar actividades administrativas es bueno saber lo más posible de nuestro sistema operativo, para ello existen múltiples comando que nos proporcionan esta información, a saber:

Algunos comando usados para conocer y manipular el Hardware:

- lscpu - Información de procesador, uso:

  ```
  $ lscpu
  ```

- lshw - Lista de Hardware en Linux, uso:

  ```
  $ lshw
  ```

- hwinfo - Información del Hardware en Linux, uso:

  ```
  # hwinfo
  ```

- dmidecode - Lista de Hardware de Linux, uso:

  ```
  # dmidecode
  ```

- hardinfo - Información del Hardware en Linux, uso:

  ```
  # hardinfo
  ```

- cpuid - Información del CPU, uso:

  ```
  $ cpuinfo
  ```

- cpufreq-info - Información sobre las frecuencias soportadas por el equipo, uso:

  ```
  # cpufreq-info
  ```

- lspci - Lista todos los dispositivos PCI, uso:

  ```
  $ lscpi
  ```

  por ejemplo, podemos obtener la cantidad de RAM de la tarjeta de video:

  ```
  lspci | grep -i "Controlador VGA" | cut -d' ' -f1 | xargs lspci -v -s | grep ' prefetchable'
  ```

- lspcmcia - Información de PCMCIA, uso:

  ```
  # lspcmcia
  ```

- lsscsi - Listar dispositivos SCSI, uso:

  ```
  $ lsscsi
  ```

- lsusb - Lista de los buses USB y detalles del dispositivo, uso:

  ```
  $ lsusb
  ```

- lsof - Información de todos los archivos abiertos y sus procesos, uso:

  ```
  $ lsof
  ```

- swapon - Nos muestra la el tamaño de la partición de intercambio (Swap), uso:

  ```
  $ swapon
  ```

- inxi - Script completo para Bash con múltiple información, uso:

  ```
  $ inxi -F
  ```

- i7z - Información en tiempo real sobre cada core, uso:

```
$ i7z
```

- stress - Permite generar carga para medir rendimiento del equipo –cpu (cpu), –vm (memoria), –io (entrada/salida), uso:

```
$ stress –cpu 2
```

- sysbench - Permite conocer el rendimiento de un sistema con Linux

```
$ sysbench cpu –cpu-max-prime=20000 run
```

- nproc - Imprime el número de cores del equipo, uso:

```
$ nproc
```

- free - Da información de la memoria RAM y Swap, uso:

```
$ free
```

- lsipc - Muestra la información de los inter-procesos de comunicación, uso:

```
$ lsipc
```

- lslocks - Muestra la información de todos los bloqueos de archivos activos, uso:

```
$ lslocks
```

- lsmem - Enumera los rangos de memoria disponibles, uso:

```
$ lsmem
```

- lsmod - Lista de los módulos cargados en el Kernel, uso:

```
$ lsmode
```

- tuptime - Reporte histórico de uso del sistema en tiempo real, uso:

```
$ tuptime
```

- modprobe - Adiciona o remueve módulos del Kernel, uso:

```
# modprobe vmhgfs
# modprobe -r vmhgfs
```

- sensors - Lee información del CPU como la temperatura, voltaje y velocidad de los ventiladores[264], uso:

[264] Para instalarlo usamos:

```
$ sensors
```

Algunos comando usados para conocer y manipular la red:

- ip - Información sobre los dispositivos de red y su configuración, uso:

```
$ ip address
```

- macchanger - Permite cambiar la MAC Addres de nuestra tarjeta de red, uso:

```
# macchange -r wlp3s0
```

- nmcli - Información sobre los dispositivos de red y su configuración, uso:

```
$ nmcli
```

- ping - Envia ICMP ECHO_REQUEST a servidores en red, uso:

```
$ ping 192.168.1.1
```

- whois - Proporciona información de un dominio, uso:

```
$ whois www.google.com
```

- nslookup - Nos da información de DNS de una dirección IP, uso:

```
$ nslookup www.google.com
```

- iperf - Permite hacer mediciones de rendimiento en red en tiempo real, uso:

```
$ iperf -s
```

```
# apt install lm_sensors
```

lo configuramos mediante:

```
# sensors-detect
```

- iperf3 - Permite hacer mediciones de rendimiento en red en tiempo real, uso:

  $ iperf3 -s -f K

- ifstat - Imprime las estadísticas de las interfaces de red, uso:

  $ ifstat

- nc - Herramienta versátil y potente de red, uso:

  $ nc -v -n 127.0.0.1 1-100

- ss - Nos muestra las estadísticas de los Sockets TCP/UDP entre otros, uso:

  $ ss

Algunos comando usados para conocer discos y sus particiones:

- df - espacio en disco de los sistemas de archivos, uso:

  $ df -h

- lsblk - lista de dispositivos de bloque, uso:

  $ lsblk

- fdisk - permite manipular la tabla de particiones de un disco, uso:

  # fdisk /dev/sda

- fsck[265] - permite detectar y corregir errores de los discos, uso:

  # fsck -ay /dev/sda1
  # fsck.ext4 -py /dev/sda1

[265] Existen comandos específicos para diferentes tipos de sistemas de archivos, por ejemplo: fsck.cramfs, fsck,exfat, fsck.ext2, fsck.ext3, fsck.ext4, fsck.fat, fsck.minix, fsck.msdos, fsck.vfat.

- badblocks - permite buscar errores en un dispositivo, uso:

  # badblocks -w -s -o error.log /dev/sda

- parted - permite manipular la tabla de particiones de un disco, uso:

  # parted /dev/sda1

- gparted - permite manipular la tabla de particiones de un disco, uso:

  # gparted

- fio - Permite hacer pruebas de dispositivos de entrada/salida, uso:

  $ fio –name=global –rw=randread –size=128m –name=job1 –name=job2

- mount - Permite montar y desmontar y ver sistema de archivo montados, uso:

  $ mount

- hdparm - Información de disco duro, uso:

  # hdparam -I /dev/sda2

Algunos comando para conocer datos del sistema operativo:

- uname, Imprime detalles de la máquina y del sistema operativo que se esta ejecutando, uso:

  $ uname -a

- lsb_release, Imprime información del sistema operativo, uso:

  $ lsb_release -a

- hostnamectl, Imprime información del equipo y del sistema operativo, uso:

```
$ hostnamectl
```

- /etc/os-release, Información del sistema operativo, uso:

```
$ cat /etc/os-release
```

- /etc/issue, Información del sistema operativo, uso:

```
$ cat /etc/issue
```

Archivos del directorio */proc/*, contienen información accesible usando el comando *cat* -muchos de los comando de Linux toman su información de este lugar-*:*

- */proc/* - Información de los procesos corriendo en el sistema
- */proc/cpuinfo* - Información de CPU
- */proc/meminfo* - Información de la memoria
- */proc/modules* - Información de los módulos cargados al Kernel
- */proc/filesystems* - Información de los sistemas de archivos soportados
- */proc/crypto* - Información sobre los sistemas de cifrado disponibles
- otras opciones son: brcm_monitor0, keys, loadavg, mtrr, softirqs, version, buddyinfo, devices, interrupts, key-users, locks, pagetypeinfo, stat, vmallocinfo, cgroups, diskstats, iomem, kmsg, partitions, swaps, vmstat, cmdline, dma, ioports, kpagecgroups, misc, sched_debug, sysrq-trigger, zoneinfo, consoles, execdomains, kallsyms, kpagecount, schedstat, timer_list, fb, kcore, kpageflags, mounts, slabinfo, uptime, etc.

Comandos que nos proporcionan informaciòn sobre dependencias y bibliotecas son:

- ldd - Nos muestra las dependencias de objetos compartidos de un comando, uso:

```
$ ldd /bin/ls
```

- ltrace - Un rastreador de llamadas a biblioteca, uso:

  $ ltrace ls

- hexdump - Despliega el contenido de un archivo en ASCII, decimal, hexadecimal o octal, uso:

  $ hexdump -c /bin/ls

- strings - Imprime las cadenas de caracteres imprimibles en el archivo, uso:

  $ strings /bin/ls

- readelf - Muestra la información del formato de archivo ejecutable y enlazable (ELF), uso:

  $ readelf -h /bin/ls

- objdump - Muestra información de un archivo objeto, uso:

  $ objdump -d /bin/ls

- strace - Nos muestra el sistema de rastreo de llamadas y señales, uso:

  $ strace -f /bin/ls

- nm - Lista los símbolos de los archivos de objeto, usó:

  $ nm archivoobjeto

- gdb - El depurador de GNU, uso:

  $ gdb -q ./ejecutable

Comandos que nos permiten apagar o reinicializar el sistema:

- shutdown - permite apagar o reinicializar el equipo, usó:

```
# shutdown -h now
# shutdown -r now
# shudown +5 "Guarda tus archivos, el equipo se apagará en
5 minutos"
```

- halt - Detiene todas las funciones del equipo, usó:

  ```
  #halt
  ```

- poweroff - apaga el equipo, usó:

  ```
  # poweroff
  ```

- reboot - reinicializa el equipo, usó:

  ```
  # reboot
  ```

Las Pruebas de Rendimiento o Benchmarks son importantes en muchos casos donde se necesita saber el rendimiento de un equipo de cómputo o de una máquina virtual. Poner a prueba nuestro sistema y Hardware es necesario para saber hasta dónde puede llegar o simplemente conocer los puntos más débiles para poder ampliarlo en un futuro o mejorar su rendimiento. Especialmente importante es para aquellos que son especialmente exigentes con el rendimiento como los diseñadores, científicos, etc.

En Linux, al igual que en otras plataformas, existen gran variedad de programas para realizar este tipo de pruebas de rendimiento. Una aplicación con interfaz gráfica es *Hardinfo*. Puedes instalarla fácilmente y ver información del Hardware y sistema que posees además de algunos datos de rendimiento, es decir, es parecido a Everest o AIDA64 que existe para Microsoft Windows. Pero no es el único programa para ello.

Otra suite de herramientas del famoso portal Phoronix - Phoronix Test Suite- un completo paquete de herramientas para medir el rendimiento a fondo gracias a las herramientas para Benchmark que probarán la CPU, RAM, disco duro, etc., todas ellas basadas en scripts PHP que irás ejecutando desde el terminal. Además, también existen otras alternativas como GeekBench, aunque prefiero la de Phoronix que además admite instalar algunos complementos muy interesantes.

Prueba de Disco con dd Una prueba rápida para medir el rendimiento de la velocidad de escritura del disco de nuestro equipo, la podemos realizar con el comando *dd*, mediante:

```
$ dd if=/dev/zero of=archivoTemporal bs=5G count=1 oflag=dsync
```

en mi máquina genera la siguiente salida:

```
1+0 records in
1+0 records out
1073741824 Bytes (1.1 GB, 1.0 GiB) copied, 2.04589 s, 568 MB/s
```

y para medir la latencia usamos:

```
$ dd if=/dev/zero of=archivoTemporal bs=512 count=1000 oflag=dsync
```

en mi máquina genera la siguiente salida:

```
1000+0 records in
1000+0 records out
512000 Bytes (512 kb, 500 KiB) copied, 8.25034 s, 62.1 kB/s
```

El comando dd es muy útil para averiguar el rendimiento de E/S secuencial simple.

Prueba de Disco con hdparam Podemos hacer la prueba de velocidad, usando el comando *hdparam*, primero debemos conocer los discos duros que tenemos en nuestro equipo, mediante:

```
$ df
```

ya sabiendo la estructura de nuestro disco, podemos pedir que haga la prueba de velocidad, por ejemplo a */dev/sda2/*, mediante:

```
# hdparm -Tt /dev/sda2
```

en mi máquina genera la siguiente salida:

/dev/sda2:

Timing Cached reads: 10348 MB in 1.99 seconds = 5189.20 MB/sec

Timing buffered disk reads: 1268 MB in 3.00 seconds = 422.49 MB/sec

Si necesitamos toda la información del disco, usamos:

#hdparm -I /dev/sda1

Nótese que ambas pruebas tiene valores distintos en cuanto a la velocidad de grabación -525 MB/s de *dd* vs 422.49 MB/sec de *hdparam*-, esta discrepancia es normal y está en función de el resto de los procesos que acceden al disco y que están activos en un cierto momento, para obtener valores más exactos, debemos haber cargado el sistema operativo en modo monousuario texto y realizar pruebas de forma exhaustiva.

Prueba Rendimiento con sysbench Con ella se puede realizar Benchmarks para hacer pruebas y conocer el rendimiento y características de algunos componentes y sistemas como la CPU, RAM, I/O, etc., todo basado en sencillas herramientas para la línea de comandos. Podemos instalarla desde consola usando:

apt install sysbench

Test de CPU con este Benchmark podremos medir el rendimiento de la CPU de nuestro equipo, obligándole a realizar algunos cálculos que se consideran costosos y mostrándonos el tiempo final. Esto es especialmente útil sobre todo en máquinas virtuales o compartidas, en las que no sabemos el Chip que tenemos debajo ni cuanta prioridad nos dio el administrador. Para comprobar el rendimiento de la CPU basta con el siguiente comando:

$ sysbench cpu –cpu-max-prime=20000 run

en este comando le indicamos que calcule **20,000** números primos. Podemos indicarle un número mayor o menor para hacer más o menos precisa la prueba. El resultado del test nos muestra el tiempo que emplea en calcularlos.

Test de Memoria con este Benchmark podremos medir el rendimiento de la memoria de nuestro equipo, en este caso usamos dos hilos:

```
$ sysbench memory –threads=2 run
```

Test de ficheros este es el otro test importante que se puede evaluar y efectuar con *sysbench*. Permite evaluar la potencia de I/O de los discos, algo que suele definir el rendimiento final de un equipo, la entrada/salida. Aquí existen muchas opciones que se le pueden pasar para realizar las comprobaciones, pero lo principal es preparar los ficheros con los que se operará. Se recomienda emplear ficheros que dupliquen por lo menos la cantidad de RAM disponible en el sistema. Si esto no es posible, habrá que indicarle que no emplee memoria Caché, para evitar obtener resultados no reales.

En este test se comprueba el rendimiento de nuestro sistema de ficheros. Este test depende de 3 etapas. La primera preparamos los ficheros con los que haremos el test. Después haremos el test propiamente dicho. Y finalmente borraremos los ficheros de la prueba.

Creamos los ficheros con:

```
$ sysbench fileio –file-total-size=100G prepare
```

Ejecutamos el test:

```
$ sysbench fileio –file-total-size=100G –file-test-mode=rndrw
\
–init-rng=on –max-time=300 –max-requests=0 run
```

Y finalmente borramos los ficheros:

```
$ sysbench fileio –file-total-size=100G cleanup
```

En esta prueba el valor relevante es la velocidad de acceso indicado en KB/sec. Señalar que se usa un valor alto de 100Gb del disco para evitar el uso de la Cache del sistema. Pero podemos usar otros valores que consideremos oportunos. El tiempo de ejecución de la prueba sería de 300 segundos aunque podemos cambiarlo por lo que consideremos.

Test de Bases de Datos esta prueba me parece muy valiosa ya que pone a prueba las bases de datos PostgreSQL y MySQL. Al igual que en la prueba del sistema de ficheros aquí requerimos de tres pasos. Primero la creación de la base de datos en por ejemplo MySQL. Luego la prueba propiamente dicha. Y finalmente el borrado de la base de datos usada. Aquí deberemos aportar los datos de acceso a la base de datos, como usuario (en el ejemplo root) y su contraseña.

Creamos la Bases de Datos:

```
$ sysbench oltp –oltp-table-size=1000 –mysql-db=test \
–mysql-user=root –mysql-password=contrasena prepare
```

Después nos disponemos a hacer la prueba con el comando:

```
$ sysbench oltp –oltp-table-size=1000 –mysql-db=test \
–mysql-user=root –mysql-password=contrasena \
–max-time=60 –oltp-read-only=on –max-requests=0 \
–num-threads=8 run
```

Y borramos la base de datos con:

```
$ sysbench oltp –mysql-db=test –mysql-user=root \
–mysql-password=contrasena cleanup
```

En esta prueba el valor relevante es el número de transacciones totales y cuántas realiza el servidor en un segundo.

13.2.1 Monitoreo y Diagnóstico de Dispositivos NVMe

NVMe significa Non-Volatile Memory Express y se refiere a cómo el Software y el almacenamiento se comunica a través de PCIe y otros protocolos, incluido TCP. Es una especificación abierta dirigida por una organización sin fines de lucro y define varias formas de almacenamiento de estado sólido. Cada vez es más común que los dispositivos de cómputo cuenten con unidades NVMe, en el caso de dispositivos de almacenamiento, los dispositivos de estado sólido NVMe son con mucho, más veloces que los discos duros equivalentes.

Para hacer el monitoreo y diagnóstico de dispositivos NVMe, podemos instalar el paquete *nvme-cli*, para ello hacemos:

```
# apt install nvme-cli
```

Después de instalar el paquete, podemos ver las opciones del paquete, mediante:

```
$ nvme help
```

Para listar todas las unidades NVMe usamos:

```
# nvme list
```

esto nos enumerará los dispositivos, el número de serie, la marca, el tamaño, la revisión del Firmware, etc. Para obtener aún más información sobre la unidad (por ejemplo /dev/nvme0n1) y las funciones que admite usamos:

```
# nvme id-ctrl /dev/nvme0n1
```

Podemos leer sobre el estado general de la unidad con el *smart-log* subcomando:

```
# nvme smart-log /dev/nvme0n1
```

esto nos proporcionará la temperatura de la unidad, las horas de uso que ha tenido hasta ahora, cuántas veces se apagó de manera insegura, etc.

Podemos también formatear la unidad (esto borrará todos los datos), usando:

```
# nvme format /dev/nvme0n1
```

además podemos borrar de manera segura (además del formateo) usando:

```
# nvme sanitize /dev/nvme0n1
```

esta última opción es ideal para cuando deseamos evitar que alguien recupere los datos de nuestras unidades NVMe.

13.2.2 Monitoreo y Diagnóstico de la GPU

Es posible obtener información básica de la tarjeta gráfica y en su caso de la GPU de nuestro equipo desde la línea de comandos, usando:

```
# lshw -C display -short
```

o en su defecto, podemos usar:

```
$ lspci -v | more
```

Podemos hacer pruebas de estrés a nuestra GPU de varias maneras, por ejemplo:

glmark2 es una prueba de rendimiento usando OpenGL 2.0 y ES 2.0, para instalarla descargamos el .deb de su página Web (https://pkgs.org/download/glmark2) y hacemos:

```
# dpkg -i glmark2*
```

y la usamos mediante:

```
$ glmark2
```

glxgears esta prueba nos muestra unos engranajes rotando, para instalarla usamos:

```
# apt install mesa-utils mesa-utils-extra
```

y la usamos mediante:

```
$ vblank_mode=0 glxgears
$ glxgears
```

gpustat esta prueba está escrita en Python, para instalarla usamos:

```
$ pip3 install gpustat
```

y la usamos mediante:

```
$ gpustat
$ gpustat -cp
```

intel_gpu_top nos despliega un sumario de Intel GPU, para instalarla usamos:

```
# apt install intel-gpu-tools
```

y la usamos mediante:

```
# intel_gpu_top
```

nvidia-smi la compañía NVIDIA provee un sistema de monitoreo para sus principales familia de arquitecturas de GPUs, para instalarla usamos:

```
# apt install nvidia_smi
```

y la usamos mediante:

```
# nvidia-smi
```

nvtop otra forma de probar el rendimiento de las GPU NVIDIA es mediante *nvtop*, para instalarlo usamos:

```
# apt install nvtop
```

y la usamos mediante:

```
$ nvtop
```

radeontop para los equipos con GPU de AMD podemos usar radeontop, para instalarla usamos:

```
# apt install radeontop
```

y la usamos mediante:

```
# radeontop
```

13.3 Instalar, Actualizar y Borrar Paquetes

Un paquete de Software en un Sistema Operativo GNU/Linux es generalmente un archivo comprimido que posee una estructura interna predefinida que facilita y permite que el mismo sea manipulado por herramientas de gestión de Software (*Synaptic, Pacman, Yum, Entropy, ZYpp,* etc.) para lograr su compilación y/o instalación, actualización y/o eliminación sobre el sistema operativo, de forma cómoda, segura, estable y centralizada.

Un paquete es compilable si su instalación se basa en su código fuente directamente (Ejm. **.tar.gz*) o instalable si lo hace en binarios compilados ya para una determinada arquitectura o plataforma (Ejm. **.deb*). La mayoría de los paquetes vienen con su documentación incluida, sus Scripts de pre y post instalación, sus archivos de configuración inicial, sus archivos de recursos, y sus binarios o el código fuente con todo lo necesario si está destinado a ser compilado.

La mayoría de los formatos de paquete vienen con sus correspondientes herramientas de gestión de Software, los más conocidos son los *.deb* creado para la distribución Debian GNU/Linux y todas sus distribuciones derivadas, y los *.rpm* creados por *Red Hat* para su propia distribución y derivadas como *Fedora* y *Open SUSE*. También existen los paquetes compilables *.ebuilds* de *Gentoo*.

El que un paquete haya sido creado para una distribución en particular no implica que sólo se pueda usar en esa distribución o derivadas, ya que basta con tener herramientas especializadas en cualquier otra distribución para la gestión de estos formatos para poder usarlos. Entre esas Herramientas tenemos: *dpkg*, *apt-get*, *aptitude*, *rpm*, *emerge*, *alíen*, entre otros.

Cada distribución mantiene almacenada su paquetería en repositorios, tanto en medios como *CDs / DVDs* como en servidores remotos, lo que permite actualizar e instalar por red todo o parte del sistema operativo desde una localización segura y confiable (repositorios oficiales) para no tener que andar buscando servidores desconocidos (e inseguros) a menos que fuese estrictamente necesarios.

Cada distribución suele aportar sus propios paquetes (parches) de seguridad y de mejoras (actualizaciones), para así lograr poner a disposición de sus comunidades de usuarios gran cantidad de Software perfectamente funcional e integrado en el sistema operativo. Y en cuanto a las dependencias entre cada paquete, las mismas suelen gestionarse de forma automática para evitar posibles problemas a los usuarios menos expertos.

¿Compilar o Instalar? lo bueno de compilar frente al instalar se puede decir que lo principal es la posibilidad de especificar opciones de compilación para tu sistema y Software usado que permiten aprovechar mejor los recursos y ajustarse a las preferencias del usuario/administrador, y lo malo lo lento y complicado que puede tornarse este proceso. Ya que por lo general, el instalar un paquete (*.deb* o *.rpm*) es muy rápido y sencillo, pero suele no conseguirse bien actualizado o ajustado a la distribución de nuestro uso o recursos de nuestro equipo de cómputo.

La Retrocompatibilidad es un enorme dolor de cabeza, tomar Software hecho para Linux de hace 10 o 5 años y ejecutarlo en una distribución moderna[266]. Cualquier cosa de mínima complejidad o que use una GUI, simplemente no funciona. Mientras la retrocompatibilidad en Windows es simplemente increíble. En Linux somos dependientes de los repo-sitorios en línea, y cuando una aplicación depende de ciertas librerías que empiezan a desaparecer de esos repositorios, nos encontramos en una pesadilla. Y mientras más viejo el Software, peor.

Si tengo un Software ahora y quiero ejecutarlo dentro de cinco o diez años en el futuro ¿Por qué no debería ser capaz de hacerlo?, Parte de la belleza del Open Source es que el código fuente está disponible, por lo que es más fácil mantener vivo el Software, de modo que no muera cuando alguien deja de mantenerlo. Excepto que mantener el Software en Linux se está convirtiendo en un desafío tan grande que daría igual que fuese privativo. Porque no vamos a ser capaces de hacerlo funcionar en un tiempo razonable, a menos que seas un desarrollador, e incluso entonces te va a costar muchos dolores de cabeza, y vas a dejar algo sin funcionar en el camino.

Instalar, Actualizar y Borrar Paquetes existen distintos comandos para hacer la instalación, actualización y borrado de paquetes en Debian GNU/Linux, el más usado actualmente es *apt* (Advanced Packaging Tool lanzado en 2014, herramienta avanzada de empaquetado que es una interfase del paquete *apt-get lanzado en 1998*), es un programa de gestión de paquetes *.deb* creado por el proyecto Debian, *apt* simplifica en gran medida la insta-

[266] Siempre estamos en posibilidad de usar una Máquina Virtual que nos permite usar un programa desarrollado hace años o décadas en su entorno original, corriendo en un equipo moderno con un sistema operativo de última generación con todas sus actualizaciones de seguridad pertinentes.

lación, actualización y eliminación de programas en GNU/Linux. Existen también programas que proporcionan estos mismos servicios como: *apt-get*, *aptitude*, *tasksel* y *dpkg*.

13.3.1 apt-get

Diariamente se actualizan decenas de paquetes en los servidores de Debian GNU/Linux, por ello es necesario hacer el mantenimiento a los paquetes del sistema cada vez que usemos el equipo de cómputo y tengamos acceso a internet, para ello debemos ser el usuario administrador *root*, mediante el comando[267]:

```
$ su
```

ya siendo *root*, podemos solicitar la descarga de las actualizaciones disponibles, usando[268]:

```
# apt-get update
```

para conocer qué paquetes instalados en el sistema están listos para ser actualizados, usamos:

```
# apt-get check
```

después, se solicita la descarga, actualización y en su caso borrado de los paquetes, mediante la descarga de los *.deb*, usando:

```
# apt-get upgrade
```

al final, se solicita el borrado de los archivos .deb de los paquetes descargados (Cache de descarga), mediante:

[267] En el Bourne Shell y sus derivados como BASH el *Prompt* que nos permite escribir los diferentes comandos, generalmente termina con el carácter:

- $ para usuario sin privilegios
- # para el administrador, conocido como *root*

[268] Si se trabaja en algunas distribuciones derivadas de Debian, es necesario usar sudo antes del comando apt-get, por ejemplo:
sudo apt-get update

```
# apt-get clean
```

Algunas veces, si se interrumpe el proceso de instalación, es posible corregir este proceso mediante:

```
# dpkg –configure -a
```

Si algún paquete instalado automáticamente ya no es requerido porque se borraron sus dependencias podemos solicitar su desinstalación mediante:

```
# apt-get autoremove
```

Si algún paquete ya no es requerido por tener ya instalada una nueva versión del mismo, podemos solicitar su desinstalación mediante:

```
# apt-get autoclean
```

El comando *apt-get* soporta las siguientes opciones:

Actualizar Listas:

```
# apt-get update
```

Revisar actualización de listas:

```
# apt-get check
```

Instalar paquete:

```
# apt-get install paquete
# apt-get install '*nombre*'
# apt-get install paquete=<version>
```

si sólo queremos simular la instalación usamos:

```
# apt-get -s install paquete
```

instala el paquete indicado solo actualizando con una nueva versión, usamos:

```
# apt-get install paquete –only-upgrade
```

instala el paquete indicado sin actualizar la versión instalada, usamos:

```
# apt-get install –no-upgrade
```

para instalar evitando que se agreguen paquetes no necesarios, usamos:

```
# apt-get install –no-install-recommends paquete
```

Reinstalar paquete:

```
# apt-get install –reinstall paquete
```

Actualizar Distro:

```
# apt-get upgrade
# apt-get dist-upgrade
# apt-get full-upgrade
```

Actualizar paquete:

```
# apt-get upgrade paquete
```

Actualizar paquetes usando dselect:

```
# apt-get dselect-upgrade
```

Eliminar paquetes:

```
# apt-get remove
# apt-get autoremove
```

Purgar paquetes:

```
# apt-get purge paquete
```

si sólo queremos simular la operación para conocer sus efectos usamos:

```
# apt-get -s purge paquete
```

Conocer paquete:

```
# apt-cache show paquete
# apt-cache showpkg paquete
# apr-cache policy paquete
# apt-cache depends paquete
# apt-cache rdepends –installed –recurse paquete
```

Listar paquetes:

```
# apt-cache search paquete
# apt-cache pkgnames paquete
# apt-cache search –name-only paquete
```

Listar dependencias de un paquete:

```
# apt-cache depends paquete
```

Listar paquetes instalados:

```
# apt-cache pkgnames –generate
# apt-show-versions
```

Validar dependencias incumplidas de un paquete:

```
# apt-cache unmet paquete
```

Configurar dependencias de un paquete:

```
# apt-get build-dep paquete
```

Descargar paquetes:

```
# apt-get source paquete
```

Corregir problemas post-instalación de paquetes:

```
# apt-get install -f
```

Forzar ejecución de orden de comando:

```
# apt-get comando -y
```

Eliminar descargas de paquetes:

```
# apt-get clean
```

Eliminar paquetes obsoletos y sin usos:

```
# apt-get autoclean
```

Trabajar con las claves GPG de un repositorio

```
# apt-key list
# apt-key del <clave>
```

Otros importantes:

```
# apt-file update
# apt-file search paquete
# apt-file list paquete
```

13.3.2 apt

Este comando es una interfase del paquete *apt-get* que viene ya instalado por omisión en el sistema, cuya sintaxis básica es:

```
# apt [opciones] comando
# apt [opciones] comando paquete
```

para usarlo debemos ser el usuario administrador *root*, mediante el comando[269]:

```
$ su
```

[269] En el Bourne Shell y sus derivados como BASH el *Prompt* que nos permite escribir los diferentes comandos, generalmente termina con el carácter:

- $ para usuario sin privilegios
- # para el administrador, conocido como *root*

ya siendo *root*, podemos solicitar la descarga de las actualizaciones disponibles, usando[270]:

```
# apt update && apt upgrade && apt clean
```

Por ejemplo, si deseamos eliminar el paquete htop e instalar el paquete vim en un solo paso, usamos:

```
# apt purge htop vim+
```

u otra forma de hacer lo mismo:

```
# apt install htop- vim
```

los sufijos + (agregar) y - (eliminar) de los nombres de los paquetes pueden anular y cambiar los interruptores de instalación/eliminación (es una función de ahorro de tiempo y escritura).

Algunas veces, si se interrumpe el proceso de instalación, es posible corregir este proceso mediante:

```
# apt –fix-broken install
# dpkg –configure -a
```

para completar las instalaciones pendientes usar

```
# apt -f upgrade
```

En algunas ocasiones es necesario saber que pasaría si instalamos o borramos un determinado paquete, para estos casos existe la opción –simulate, que se usa:

```
# apt install vlc –simulate
# apt purge vlc –simulate
```

El comando *apt* soporta las siguientes opciones:

Editar la información de /etc/apt/sourcelist

[270] Si se trabaja en algunas distribuciones derivadas de Debian, es necesario usar sudo antes del comando apt, por ejemplo:

```
# sudo apt update
```

```
# apt edit-sources
```

Actualizar:

```
# apt update
```

Conocer los paquetes susceptibles a ser actualizados:

```
# apt list –upgradable
```

Buscar paquetes:

```
$ apt search php
$ apt search mysql-5.?
$ apt search mysql-server-5.?
$ apt search httpd*
$ apt search ^apache
$ apt search ^nginx
$ apt search ^nginx$
```

Instalar paquete:

```
# apt install paquete
# apt -s install paquete
# apt install paquete –no-upgrade
# apt install paquete –only-upgrade
```

Reinstalar paquete:

```
# apt reinstall paquete
# apt install –reinstall paquete
```

Actualizar la distribución:

```
# apt upgrade
# apt full-upgrade
# apt dist-upgrade
```

Actualizar paquete:

```
# apt upgrade paquete
```

Eliminar paquete:

```
# apt remove paquete
# apt autoremove
```

Purgar paquete:

```
# apt purge paquete
# apt -s purge paquete
```

Conocer paquete:

```
# apt show paquete
# apt showsrc paquete
# apt changelog paquete
# apt depends paquete
# apt rdepends paquete
# apt rdepends –installed –recurse paquete
# apt hold paquete
# apt unhold paquete
# apt policy
```

Listar paquetes:

```
# apt list paquete
# apt list –installed
# apt list –upgradeable
# apt list –all-versions paquete
```

Listar paquetes instalados / actualizables:

```
# apt list –installed
# apt list –upgradeable
```

Corregir problemas post-instalación de paquetes:

```
# apt install -f
```

Forzar ejecución de orden de comando:

```
# apt comando -y
```

Eliminar descargas de paquetes:

```
# apt clean
```

Eliminar paquetes obsoletos y sin usos:

```
# apt autoclean
```

Otros importantes:

```
# apt edit-sources
# apt build-dep paquete
# apt source paquete
# apt download paquete
# apt-mark
# apt-mark hold paquete
# apt-mark unhold paquete
```

Cuando se hace una búsqueda, el comando *apt* toma información sobre el estado de la aplicación en su sistema además de lo que arroja la búsqueda de apt-cache, al inicio de la salida hay algunas banderas. Aquí hay una lista de banderas que verá en su terminal y lo que significan:

A- se instala automáticamente, tal vez como parte de un meta-paquete más grande o la instalación del sistema operativo en sí.

B- El paquete está marcado como roto y debe reinstalarse.

H- Medio instalado. El paquete debe terminar de instalarse.

c- El paquete fue eliminado, pero su "fantasma" persiste en forma de archivos de configuración. Puede resolver esto usando *apt-get purge* o *apt purge*, seguido del nombre del paquete con esta bandera.

p- El paquete se purgó o nunca se instaló.

v- El paquete es usado genéricamente por otros para proporcionar una función. Por ejemplo, Firefox proporciona capacidades de

navegación que pueden ser utilizadas por otras aplicaciones, lo que lo convierte en un paquete virtual.

i- Este paquete está instalado en su sistema.

h- Hay una retención en este paquete, lo que evita que se actualicen a versiones más recientes.

13.3.3 aptitude

Este comando[271] es también una interfase del paquete *apt-get* con algunas opciones adicionales (*aptitude* instala también las dependencias sugeridas y *apt-get* instala sólo las recomendadas) no viene instalado por omisión en el sistema, para instalarlo debemos ser el usuario administrador *root*, mediante el comando[272]:

```
$ su
```

ya siendo *root*, lo instalamos usando:

```
# apt install aptitude
```

y soporta las siguientes opciones[273]:

Actualizar Listas:

```
# aptitude update
```

Instalar paquete:

[271] Es también un FrontEnd basado en Ncurses para *apt*, debido a que su interfaz está basado en modo texto, puede ejecutarse desde una terminal, pero también en línea de comandos. Generalmente resuelve las dependencias de forma más inteligente que *apt-get*.

[272] En el Bourne Shell y sus derivados como BASH el *Prompt* que nos permite escribir los diferentes comandos, generalmente termina con el carácter:

- $ para usuario sin privilegios
- # para el administrador, conocido como *root*

[273] Si se trabaja en algunas distribuciones derivadas de Debian, es necesario usar sudo antes del comando aptitude, por ejemplo:
sudo aptitude update

```
# aptitude install paquete
# aptitude -s install paquete
# aptitude install –without-recommends paquete
```

Reinstalar paquete:

```
# aptitude reinstall paquete
```

Actualizar Distro:

```
# aptitude upgrade
# aptitude safe-upgrade
# aptitude full-upgrade
```

Actualizar paquete:

```
# aptitude upgrade paquete
```

Eliminar paquetes:

```
# aptitude remove paquete
```

Purgar paquetes:

```
# aptitude purge paquete
# aptitude -s purge paquete
# aptitude –purge remove paquete
```

Listar paquetes:

```
# aptitude search paquete
```

Listar paquetes instalados / rotos:

```
# apt search [*] | grep "^i"
# apt search [*] | grep "^B"
```

Configurar dependencias de un paquete:

```
# aptitude build-dep paquete
```

Descargar paquetes:

```
# aptitude download paquete
```

Corregir problemas post-instalación de paquetes:

```
# aptitude install -f
# aptitude -f safe-upgrade
```

Forzar ejecución de orden de comando:

```
# aptitude comando -y
```

Eliminar descargas de paquetes:

```
# aptitude clean
```

Eliminar paquetes obsoletos y sin usos:

```
# aptitude autoclean
```

Conocer paquete:

```
# aptitude show paquete
# aptitude show "?installed ?section(fonts)" | egrep '(Paquete|Estado|Versión)'
# aptitude show "?not(?installed) ?section(fonts)" | egrep
'(Paquete|Estado|Versión)'
# aptitude show "?section(fonts)" | egrep '(Paquete|Estado|Versión)'
# aptitude why paquete
```

Otros importantes:

```
# aptitude (un)hold, aptitude (un)markauto, why, why-not
```

Si se han instalado y desinstalado múltiples paquetes a lo largo del tiempo en nuestro equipo es recomendable ver que paquetes dejaron archivos de configuración que al ser desinstalados los paquetes no fueron borrados, para ello usamos:

```
# aptitude purge ~c
```

13.3.4 dpkg

En ocasiones tenemos que instalar un paquete *.deb* que hemos descargado de algún lugar de la Web (de confianza por supuesto). Por lo general, se trata de aplicaciones que no se encuentran en los repositorios pero que están disponibles en forma de paquete *.deb* (navegadores como SRWare Iron, Chrome, Google Earth, Skype, Dropbox, etc).

El programa base para manejar paquetes Debian GNU/Linux en el sistema es *dpkg*. Si tiene paquetes *.deb*, *dpkg* es lo que permite instalar o analizar sus contenidos. Pero este programa sólo tiene una visión parcial del universo Debian: sabe lo que está instalado en el sistema y lo que sea que se le provee en la línea de órdenes, pero no sabe nada más de otros paquetes disponibles. Como tal, fallará si no se satisface una dependencia. Por el contrario, herramientas como apt y aptitude crearán una lista de dependencias para instalar todo tan automáticamente como sea posible.

Se debe ver a *dpkg* como una herramienta de sistema (tras bambalinas), *apt* y *aptitude* como una herramienta más cerca del usuario que evita las limitaciones del primero. Estas herramientas trabajan juntas, cada una con sus particularidades, adecuadas para tareas específicas.

Instalar paquetes para instalar paquetes *.deb* con *dpkg* tenemos que dirigirnos a la carpeta en la que tengamos el paquete *deb*. Puede ser con el explorador de archivos y abrir terminal; en *Dolphin*, por ejemplo, con F4. O con el comando cd (cambiar directorio). Abrimos terminal y operamos como superusuario (su) o usuario con privilegios de root (sudo):

```
# cd /home/usuario/Descargas
```

(suponiendo que el paquete *.deb* está en Descargas de nuestro usuario)

```
# dpkg -i paquete.deb
```

Puede que el paquete se instale sin más problemas porque las posibles dependencias ya las tengamos instaladas. Si nos lanza el error de que hay dependencias no satisfechas, estas se resolverán con:

```
# apt -f install
```

Este comando descargara las dependencias requeridas, las instalará y concluirá la instalación del paquete *.deb* que quedó interrumpida[274].

[274] También podemos usar:

Desinstalar paquetes para remover paquetes *.deb* con *dpkg* tenemos solo que indicar el nombre completo del archivo *.deb* instalado, por ejemplo:

```
# dpkg -r unp
```

Este comando (-r o –remove) eliminará sólo el paquete, pero no los archivos de configuración. Pero si queremos eliminar todo lo relacionado con el paquete en cuestión utilizaremos purge (-P o –purge):

```
# dpkg -P unp
```

en caso de ser necesario forzar la purga usar –force.

Conocer el contenido de un paquete no instalado para conocer el contenido de un paquete no instalado, por ejemplo para *teams*, usamos:

```
$ dpkg -c teams_1.2.00.32451_amd64.deb
```

también podemos usar –info o -L.

Conocer las dependencias de un paquete para conocer las dependencias de un paquete no instalado, por ejemplo para *teams*, usamos:

```
$ dpkg -I teams_1.2.00.32451_amd64.deb
```

Conocer el contenido de un paquete instalado para conocer la ubicación y el contenido de un paquete instalado, por ejemplo para *unp*, usamos:

```
$ dpkg -L unp
```

si necesitamos conocer información del paquete usamos:

```
$ dpkg -s unp
```

en caso de necesitar saber a qu paquete pertenece un determianado comando, usamos:

```
$ dpkg -S /usr/bin/unp
```

apt install ./paquete.deb

Conocer los paquetes instalados para conocer todos los paquetes instalados en el sistema, usamos:

```
# dpkg -l
# dpkg -l patrón_búsqueda
# dpkg –get-selections nombre_paquete
# dpkg –get-selections | grep -v deinstall > lista.txt
```

Problemas algunas veces, si se interrumpe el proceso de instalación, es posible corregir este proceso mediante:

```
# dpkg –configure -a
# dpkg –configure nombre_paquete
# dpkg –configure –pending
```

Búsqueda de paquetes obsoletos con el tiempo algunos paquetes del sistema operativo quedan obsoletos, podemos encontrar estos paquetes usando:

```
# dpkg -l | grep obsolete
```

Otras opciones:

Descomprimir paquete:

```
# dpkg –unpack nombre_paquete
```

Buscar archivos de paquetes instalados:

```
# dpkg -S nombre_archivo
```

13.3.5 tasksel

Múltiples configuraciones de paquetes pueden instalarse desde *tasksel*, este contiene metapaquetes que facilitan la instalación de un sistema Debian, esto es recomendado cuando se tiene poco conocimiento sobre la instalación de paquetes en Debian GNU/Linux, muy usado cuando se hace la instalación de un sistema Debian con instalación mínima, i.e. sin ambiente gráfico, para hacer uso de este recurso, hacer:

```
# tasksel
```

Para conocer los metapaquetes instalados y que se pueden instalar, usar:

```
# tasksel –list-tasks
```

Para conocer los paquetes que contiene un metapaquete

```
# tasksel –task-packages Web-server.
```

13.3.6 deborphan

En algún momento, ciertos paquetes pueden quedar huérfanos y no ser necesarios en el sistema, para conocer estos, es necesario instalar *deborphan*, mediante:

```
# apt install deborphan
```

para luego correrlo, mediante:

```
# orphaner
```

donde es posible conocer los paquetes huérfanos y liberarlos en caso necesario.

13.3.7 debian-goodies

Debian-goodies es un paquete que incluye utilidades estilo caja de herramientas que se utilizan para administrar Debian y sus sistemas derivados, como Ubuntu, Linux Mint, Kali Linux, etc.

Las utilidades del paquete debian-goodies están desarrolladas de tal manera que se combinan con muchas herramientas de Shell reconocidas y otras se incluyen porque no pueden desarrollarse como sus paquetes en distribuciones de Linux basadas en Debian. Las utilidades del paquete debian-goodies incluye: dglob, debget, dpigs, dgrep, debmany, checkrestart, popbugs y whatpkg-broke.

Veamos la descripción de cada utilidad a continuación:

- dglob: produce una lista de nombres de paquetes que coinciden con un patrón

- dgrep: busca una expresión regular en todos los archivos de los paquetes determinados
- dpigs: muestra qué paquetes instalados ocupan más espacio en disco
- deuda: obtenga un .deb para un paquete en la base de datos de APT
- debmany: elija páginas de manual de paquetes instalados o eliminados
- checkrestart: busca y reinicia procesos que utilizan versiones obsoletas de archivos actualizados
- popbugs: muestra un informe personalizado de errores críticos para la versión basado en los paquetes que utilizas
- what-pkg-broke: detecta qué paquete podría haber roto otro

Estas son utilidades muy útiles que pueden facilitar mucho la administración del sistema cuando se usan con otras herramientas de Shell. De hecho, la herramienta Debian-goodies muestra más información sobre paquetes que las herramientas estándar como dpkg y apt tools.

Para instalar el paquete debian-goodies, ejecute:

```
#apt install debian-goodies
```

13.4 Revisión de la Seguridad del Sistema

El uso de Linux como sistema operativo de nuestro equipo o servidor es, ya de por sí, un símbolo de seguridad. El uso de contraseñas Linux es el método más simple y popular para añadir seguridad a cualquier equipo, siempre teniendo presente las medidas básicas de seguridad de las contraseñas como la longitud, caracteres especiales y demás. Cuando se instala un sistema operativo Linux, se crea tanto el usuario administrador, el cual tiene más permisos que un usuario base, como el usuario *root* con el que hay que tener extremo cuidado, ya que posee todos los permisos para realizar acciones en el sistema como crear, editar o eliminar usuarios, aplicaciones etc. Si no tenemos experiencia en el uso y manejo de usuarios root, es recomendable leer al respecto ya que, tomando decisiones equivocadas con los permisos de este usuario, pueden llegarse a dejar inservibles ciertos dispositivos.

Por ello, el trabajo de este usuario debe ser especialmente responsable, si por alguna razón deseamos acceder a Linux con este usuario *root* y hemos olvidado la contraseña, lo que pasa con más frecuencia de lo que imaginamos, debemos seguir los pasos que aquí indicamos junto con los consejos necesarios para recuperar dicha contraseña, teniendo la opción de cambiarla o eliminarla sin ningún problema, ya que, con el uso de root es más simple ejecutar las tareas de soporte y administración en el sistema evitando el uso de «sudo» para dichas operaciones.

¿Qué hacer si olvido/no conozco la clave de root de un equipo? La contraseña de administrador o *root* en un servidor con sistema operativo Linux no debemos olvidarla nunca, sin embargo, en el caso de que la hayamos olvidado o algún compañero de trabajo la haya cambiado y no tengamos la nueva, siempre podremos modificar o eliminar esta contraseña directamente desde el GRUB[275] de arranque en nuestro sistema operativo. La contraseña de *root* siempre se almacena en local en un archivo en concreto, con el objetivo de poder autenticarnos cuando arranca el sistema operativo para acceder a él.

La ventaja de utilizar el GRUB para «recuperar» la contraseña es que es la forma más limpia de poder hacerlo sin necesidad de tener que formatear nuestro servidor y empezar desde cero. Además, también es la forma más rápida de hacerlo y no tendremos ningún tipo de pérdida de información. La desventaja de usar este método, es que cualquier usuario con acceso físico al servidor podría también resetear o borrar la contraseña de *root* de nuestro

[275] El GRUB (GRand Unified Bootloader) es el cargador de arranque desarrollado por el proyecto GNU que nos permitirá elegir qué sistema operativo debe arrancar en el ordenador o en el servidor. Normalmente se utiliza en sistemas operativos basados en GNU/Linux, pero también es capaz de arrancar un sistema operativo Windows sin ningún problema. Otras características interesantes de GRUB es que admite comandos de configuración y puede cargar una configuración predeterminada, además, tenemos una interfaz de menú y también una línea de comandos para ejecutar lo que nosotros queramos, admite todos los sistemas de archivos que se usan actualmente. También es capaz de acceder a datos ubicados en cualquier dispositivo instalado, e incluso admite direccionamiento de bloque lógico (LBA) y arranque en red.

El GRUB es lo primero que se carga en el arranque del sistema, la BIOS o UEFI se encargará de buscar un dispositivo de inicio como el disco duro o SSD, y cargará el MBR con el GRUB en su interior. En la primera fase arrancará el GRUB básico, y se encargará de cargar el GRUB completo ubicada en cualquier parte del disco. Una vez que haya cargado, se presentará al usuario el menú de inicio, y podremos cargar el Kernel o el sistema operativo que nosotros deseemos.

sistema operativo. Para evitar este tipo de problemas, es muy recomendable y necesario por seguridad, que el GRUB también tenga una contraseña de acceso, con el objetivo de evitar posibles problemas de seguridad y que alguien con acceso físico al servidor sea capaz de modificar nuestra contraseña de acceso.

¿Cómo lo hago? si queremos recuperar la contraseña de *root*, o, mejor dicho, resetear esta contraseña, deberemos encender o reiniciar el sistema para acceder a la consola inicial de GRUB, la cual nos va a permitir iniciar el proceso de recuperación de contraseña y que cuando se despliega, pulsamos la tecla «e» para acceder al editor de GRUB. Allí usamos las flechas de desplazamiento de nuestro teclado para ubicar la línea que inicia con «Linux» y al final de ella está el término «ro single». Tenemos que cambiar la cadena «*ro single*» por «*rw init=/bin/bash*» (debes acordarte quitar también el «single»).

Ahora debemos usar las teclas Ctrl + X o F10 para iniciar el modo de usuario simple (single-user). Lo que hará es cargar en modo lectura y escritura ejecutando la terminal Bash, para resetear la contraseña de root. Esto nos permite habilitar permisos de escritura en el gestor de inicio del sistema, y poder realizar modificaciones en el sistema operativo.

Y una vez accedemos a la línea de consola ejecutamos el siguiente comando:

```
# passwd
```

y podremos cambiar la contraseña del superusuario de manera fácil y rápida. Es en este paso es cuando debemos elegir si queremos cambiar o eliminar la contraseña de root, ya que, si escribimos algo y aceptamos el texto, lo que hayamos escrito quedará como contraseña. En caso de querer eliminarla, no ingresaremos ningún texto y aceptaremos igualmente como nueva contraseña para el usuario root, después recibiremos la confirmación del cambio que acabamos de ejecutar, si hemos escrito algo será la nueva contraseña y si no hemos escrito nada, se eliminará la contraseña y no nos la solicitará a la entrada al *root*.

Después de esto podemos iniciar Linux de nuevo y acceder con el usuario *root* usando la contraseña que hemos establecido anteriormente, o accediendo directamente en caso de haberla borrado.

Con estos sencillos pasos lograremos cambiar o eliminar la contraseña de *root*, solo tenemos que fijarnos en lo que elegimos en el par de pasos intermedios, puesto que es lo más complicado de este método, el resto de pasos son sencillos y en pocos minutos podremos tener cambiada o eliminada nuestra contraseña *root*, desde aquí te recomendamos cambiarla, porque eliminarla puede ser altamente peligroso y muy poco recomendable.

¿Es esto un fallo de seguridad de Linux? por lo sencillo del método, parece que se puede acceder a cualquier sistema operativo Linux a través del cambio o eliminación de la contraseña de root. Sin embargo, debemos tener presente que es necesario estar físicamente en el ordenador o servidor, o al menos, tener una interfaz de administración con acceso físico al equipo, de esta forma, podremos acceder al root. No se puede hacer vía remota a través de protocolos como SSH o Telnet, solamente con acceso físico.

Si quieres evitar que alguien pueda realizar este método para cambiar o eliminar la clave de *root*, tendrás que incorporar una contraseña de autenticación en el GRUB, sin esta contraseña no podrás iniciar el GRUB y proceder con el paso. Otra forma es cifrar por completo la partición del sistema operativo con LUKS o similares, de esta forma, no se podrá acceder al cambio del sistema operativo si no conocemos esta contraseña de paso.

Por otro lado, una de las principales actividades de un administrador de sistemas es tener al día las actualizaciones del sistema operativo, para ello es necesario tener actualizadas nuestras aplicaciones como vimos en la sección anterior.

debscan podemos usar la herramienta *debscan* para realizar una revisión de la seguridad del sistema y nos dice si hay reportadas vulnerabilidades asociadas a los paquetes instalados en nuestro equipo. Para instalarlo usamos:

```
# apt install debsecan
```

una vez instalado, debemos definir la versión del sistema operativo instalado, para ello hacemos:

```
# dpkg-reconfigure debsecan
```

y ahora podemos usarlo, por ejemplo:

```
# debsecan –suite buster
```

si deseamos más detalles usamos:

```
# debsecan –suite buster –format detail
```

más opciones del paquete las podemos conocer con:

```
$ man debsecan
```

chkrootkit es una herramienta para interfaces de línea de comandos disponible para sistemas operativos del tipo UNIX y GNU/Linux. Es una herramienta de seguridad que realiza un análisis rápido en busca de signos y cambios en ciertas configuraciones del sistema que indiquen la presencia de *rootkits*[276]. Esta herramienta de seguridad se encuentra disponible para su descarga en la mayoría de las distribuciones GNU/Linux.

Los rootkits se pueden clasificar en dos grupos:

- Rootkit a nivel Kernel: los que actúan desde el Kernel añaden o modifican una parte del código de dicho núcleo para ocultar el Backdoor. Normalmente este procedimiento se complementa añadiendo nuevo código al Kernel, ya sea mediante un controlador (Driver) o un módulo, como los módulos del Kernel de Linux o los dispositivos del sistema de Windows. Estos rootkits suelen parchear las llamadas al sistema con versiones que esconden información sobre el intruso. Son los más peligrosos, ya que su detección puede ser muy complicada.

[276] Un rootkit se usa habitualmente para esconder algunas aplicaciones que podrían actuar en el sistema atacado. Suelen incluir Backdoors (puertas traseras) para ayudar al intruso a acceder fácilmente al sistema una vez que se ha conseguido entrar por primera vez. Por ejemplo, el rootkit puede esconder una aplicación que lance una consola cada vez que el atacante se conecte al sistema a través de un determinado puerto. Los rootkits del Kernel o núcleo pueden contener funcionalidades similares. Un Backdoor puede permitir también que los procesos lanzados por un usuario sin privilegios de administrador ejecuten algunas funcionalidades reservadas únicamente al superusuario. Todo tipo de herramientas útiles para obtener información de forma ilícita pueden ser ocultadas mediante rootkits.

Los rootkits se utilizan también para usar el sistema atacado como «base de operaciones», es decir, usarlo a su vez para lanzar ataques contra otros equipos. De este modo puede parecer que es el sistema infiltrado el que lanza los ataques y no el intruso externo. Este tipo de ataques podrían ser de denegación de servicio (DoS), ataques mediante IRC o mediante correo electrónico (spam).

- Rootkit a nivel aplicación: Los rootkits que actúan como aplicaciones pueden reemplazar los archivos ejecutables originales con versiones crackeadas que contengan algún troyano, o también pueden modificar el comportamiento de las aplicaciones existentes usando Kacks, parches, código inyectado, etc.

En Debian GNU/Linux el comando *chkrootkit* se encuentra disponible en los repositorios oficiales, por lo que para instalar esta herramienta ejecutamos:

```
# apt install chkrootkit
```

El comando chkrootkit es una herramienta para buscar localmente signos de un rootkit. Si bien su ejecución es directa, cuenta con los siguientes Scripts que llevaran adelante las diferentes instancias de análisis cuando ejecutemos el comando:

- chkrootkit: El programa principal que revisa los binarios del sistema operativo en busca de modificaciones hechas por rootkits para saber si los códigos fueron adulterados.
- Ifpromisc.c: Revisa si las interfaces están modo promiscuo, si una interfaz de red está en modo promiscuo puede ser usada por un atacante o por un Software malicioso para captura tráfico de la red y analizarlo después.
- chklastlog.c: Revisa lo que fue eliminado de lastlog. Lastlog es un comando que muestra información sobre los logins en el sistema. Un atacante o rootkit podría modificarlos para evitar detección si el administrador usa el comando para acceder a información sobre los ingresos.
- Chkwtmp.c: Similar al Script anterior, este Script revisa el archivo wtmp que contiene información sobre los accesos en el sistema, chkwtmp.c intenta detectar modificaciones en este archivo.
- chkproc.c: Busca indicios de troyanos en LKM (Loadable Kernel Modules).
- chkdirs.c: Similar al anterior revisa por troyanos dentro de los módulos del Kernel.

- strings.c: Busca por reemplazo de Strings que puedan ocultar rootkits.
- Chkutmp.c: Trabaja sobre el archivo utmp.

El comando chkrootkit ejecutado de forma directa, llevará adelante una búsqueda intensiva por etapas, donde cada etapa se lleva a cabo por medio de los Scripts mencionados anteriormente. Es importante señalar que para su ejecución se necesita conceder permisos de administrador. Veamos:

```
# chkrootkit
```

El comando chkrootkit cuenta con algunas opciones que vamos a conocer a continuación.

- Modo de escaneo experto: El test de rootkit en modo experto realiza las mismas acciones que el test por defecto del comando chkrootkit, solo que cambia la forma en la que nos muestra los resultados por la salida estándar. Si te interesa probar este modo, basta con ejecutar:

  ```
  # chkrootkit -x
  ```

- Modo de escaneo silencioso: Al igual que en el modo experto, el modo silencioso realiza las mismas acciones que si ejecutáramos el comando de forma directa, solo que por la salida estándar no nos imprimirá las instancias del test. Si deseas utilizar el comando chkrootkit de esta forma, ejecuta:

  ```
  # chkrootkit -q
  ```

- Lista de objetivos del test chkrootkit: Para ver la lista de pruebas que va a llevar a delante el comando chkrootkit contamos con la opción -l, veamos.

  ```
  # chkrootkit -l
  ```

- Depurar: Si quisiéramos depurar el comando chkrootkit contamos con el parámetro -d. Se ejecuta de la siguiente manera:

  ```
  # chkrootkit -d
  ```

NOTA: Las herramientas de detección de rootkits no disponen de argumentos que nos permitan eliminarlos. Para ello es necesario una vez identificado el rootkit, buscar como suprimirlos en foros de esta temática ya que no son tan simples de tratar y en el peor de los casos, será necesario reinstalar el sistema operativo completo evitando contaminar la instalación limpia con respaldos contaminados con rootkits.

aide Es un sistema de detección de intrusos *AIDE* (Advanced Intrusion Detection Environment) que permite monitorizar la integridad de archivos, directorios analizando si han cambiado sus características y contenido por medio de varias reglas y directivas. Para instalarlo usamos:

```
# apt install aide
```

para inicializar el paquete, usamos:

```
# aide –init
```

ahora cuando necesitemos revisar la integridad de ficheros y directorios usamos:

```
# aide –check
```

si encuentra modificaciones no esperadas, generará el reporte correspondiente. Para que *aide* tenga en cuenta las últimas modificaciones realizadas a los ficheros, es necesario actualizar, usando:

```
# aide –update
```

de ser requerido es posible poner *aide* en *crontab*.

13.5 Cron y Crontab

Aunque pueda parecer que estamos hablando de lo mismo, no es así, y se podrían considerar dos elementos dependientes uno del otro. Para ser más claros, son los encargados de que la programación de tareas en sistemas Linux sea posible.

¿Qué es cron? El nombre cron viene del griego chronos que significa "tiempo". En el sistema operativo Unix/Linux, cron es un administrador regular de procesos en segundo plano (demonio) que ejecuta procesos o Scrpts a intervalos regulares (por ejemplo, cada minuto, día, semana o mes). Los procesos que deben ejecutarse y la hora en la que deben hacerlo se especifican en el fichero crontab.

Cómo funciona el demonio cron inicia de */etc/rc.d/* o */etc/init.d* dependiendo de la distribución. Cron se ejecuta en el Background, revisa cada minuto la tabla de tareas crontab /etc/crontab o en */var/spool/cron* en búsqueda de tareas que se deban cumplir. Como usuario podemos agregar comandos o Scripts con tareas a cron para automatizar algunos procesos. Esto es útil por ejemplo para automatizar la actualización de un sistema o un buen sistema de respaldos.

¿Qué es Crontab? Crontab es un simple archivo de texto que guarda una lista de comandos a ejecutar en un tiempo especificado por el usuario. Crontab verificará la fecha y hora en que se debe ejecutar el Script o el comando, los permisos de ejecución y lo realizará en el Background. Cada usuario puede tener su propio archivo crontab, de hecho el */etc/crontab* se asume que es el archivo crontab del usuario root, cuando los usuarios normales (e incluso root) desean generar su propio archivo de crontab, entonces utilizaremos el comando *crontab*.

Crontab es la manera más sencilla de administrar tareas de *cron* en sistemas multiusuario, ya sea como simple usuario de sistema o usuario root.

Este archivo */etc/crontab* tiene la siguiente estructura para cada una de sus entradas:

```
min   hor   día   mes   dds   usr      comando
o
.————————- minuto (0 - 59)
|   .————————- hora (0 - 23)
|   |   .————————- día del mes (1 - 31)
|   |   |   .————— mes (1 - 12) O jan,feb,mar, ...
|   |   |   |   .—— día de la semana (0 - 6) (Domingo=0 - 7) o sun,mon,...
|   |   |   |   |   |- usuario
*   *   *   *   * *   *       comando para ser ejecutado
```

Donde:

- m corresponde al minuto en que se va a ejecutar el Script, el valor va de 0 a 59
- h la hora exacta, se maneja el formato de 24 horas, los valores van de 0 a 23, siendo 0 las 12:00 de la medianoche.

- día hace referencia al día del mes, por ejemplo se puede especificar 15 si se quiere ejecutar cada dia 15
- mes hace referencia al mes, 1 - 12 o jan,feb,mar, ...
- dds significa el día de la semana, puede ser numérico (0 a 7, donde 0 y 7 son domingo) o las 3 primeras letras del día en inglés: mon, tue, wed, thu, fri, sat, sun.
- usr define el usuario que va a ejecutar el comando, puede ser root, u otro usuario diferente siempre y cuando tenga permisos de ejecución del Script.
- comando refiere al comando o a la ruta absoluta del Script a ejecutar, ejemplo: /home/usuario/Scripts/actualizar.sh (si se llama a un Script, este debe ser ejecutable).

Notemos: Una expresión CRON es una cadena que comprende 5 o 6 campos separados por espacios en blanco que representan un conjunto de horas, normalmente como un cronograma para ejecutar alguna rutina.

Formato:

- Minutes, obligatorio, valores: 0-59, caracteres: * / , -
- Hours, obligatorio, valores: 0-23, caracteres: * / , -
- Day of month, obligatorio, valores: 1-3, caracteres: * / , - ? L W
- Month, obligatorio, valores: 1-12 o JAN-DEC, caracteres: * / , -
- Day of week , obligatorio, valores: 0-6 o SUN-SAT, caracteres: * / , - ? L #
- Year, No obligatorio, valores: 1970-2099, caracteres: * / , -

Por ejemplo: Se pone a cero el log de error de Apache un minuto después de medianoche (00:01 de cada día del mes, de cada día de la semana)

```
1 0 * * * usr echo -n "" > /www/apache/logs/error_log
```

Se ejecuta el Script: /home/user/test.pl cada 5 minutos

```
*/5 * * * * usr /home/user/test.pl
```

Ejecuta la orden who todos los lunes cada media hora y guarda la salida en el fichero quien.txt

```
0,30 * * * 1 usr /usr/bin/who >> /home/quien.txt
```

Ejecutará el Script actualizar.sh a las 10:15 a.m. todos los días

```
15 10 * * * usuario /home/usuario/Scripts/actualizar.sh
```

Ejecutará el Script actualizar.sh a las 10:15 p.m. todos los días

```
15 22 * * * usuario /home/usuario/Scripts/actualizar.sh
```

Ejecutará una actualización todos los domingos a las 10:00 a.m

```
00 10 * * 0 root apt-get -y update Usuario root
```

Usuario root ejecutará una actualización todos los domingos (sun) a las 10:45 a.m

```
45 10 * * sun root apt -y update
```

El día 20 de noviembre a las 7:30 el usuario correrá el Script

```
30 7 20 11 * usuario /home/usuario/Scripts/actualizar.sh
```

Un molesto recordatorio cada minuto de cada hora todos los días.

```
01 * * * * usuario /home/usuario/Scripts/molestorecordatorio.sh
```

A las 5:30 de la tarde todos los días de lunes a viernes.

```
30 17 * * 1,2,3,4,5 usuario /home/usuario/Scripts/Scripts.sh
```

A las 12 del día todos los días primero, quince y 28 de cada mes (ideal para nóminas)

```
00 12 1,15,28 * * usuario /home/usuario/Scripts/Scripts.sh
```

Por omisión *crontab* envía un correo a la cuenta del usuario que ejecuta el trabajo. Si deseamos deshabilitar el envío de dicho correo, podemos usar > /dev/null 2>&1 al final de la instrucción usada en crontab, por ejemplo:

```
00 12 1,15,28 * * usuario /home/usuario/Scripts/Scripts.sh >
/dev/null 2>&1
```

Hay varios valores predefinidos que se pueden utilizar para sustituir la expresión CRON.

Entrada Descripción

- @yearly, se ejecuta una vez al año Equivale a: 0 0 1 1 *
- @annually, (igual que @yearly) Equivale a: 0 0 1 1 *
- @monthly, se ejecuta una vez al mes Equivale a: 0 0 1 * *
- @weekly, se ejecuta una vez a la semana Equivale a: 0 0 * * 0
- @daily, se ejecuta una vez al dia Equivale a: 0 0 * * *
- @midnight, (igual que @daily) Equivale a: 0 0 * * *
- @hourly, se ejecuta una vez cada hora Equivale a: 0 * * * *

Su uso es muy sencillo.

```
@hourly usuario /home/usuario/Scripts/molestorecordatorio.sh
@monthly usuario /home/usuario/Scripts/respaldo.sh
@daily root apt-get update && apt-get -y upgrade
```

También está disponible @reboot, que permite a un trabajo ejecutarse una vez cada vez que el demonio cron se inicie, que eso típicamente coincidir con el arranque del servidor. Puede ser útil si es necesario levantar un servidor o demonio bajo un usuario en particular o si el usuario no tiene permisos al archivo rc.d/init.d.

Para agregar tareas:

```
# crontab -e
```

Agregar:
@reboot /home/scrip

Actualizar

```
# update-rc.d cron defaults
```

Revisar las tareas del usuario actual:

```
# crontab -l
```

Revisar las tareas de usuario indicado:

```
# crontab -u usuario -l
```

Remover las entradas a Crontab:

```
# crontab -i -r
```

Caracteres especiales

Asterisco (*) indica que la expresión cron coincidirá con todos los valores del campo, por ejemplo, usar un asterisco en el cuarto campo (mes) indicaría cada mes.

Diagonal (/) se utilizan para describir incrementos de rangos. Por ejemplo 3-59 / 15 en el primer campo (minutos) indicaría el 3er minuto de la hora y cada 15 minutos a partir de entonces. los la forma "* / ..." es equivalente a la forma "first-last / ...", es decir, un incremento sobre el mayor alcance posible del campo

Porcentaje (%) los signos de porcentaje (%) en el comando, a menos que se hayan escapado con una barra diagonal inversa (\), se cambiarán a los caracteres de nueva línea y todos los datos después del primer % se enviarán al comando como estándar entrada.

Coma (,) se usan para separar elementos de una lista. Por ejemplo, usando "MON, WED, FRI" en el quinto campo (día de la semana) significaría lunes, miércoles y viernes.

Guión (-) se utilizan para definir rangos. Por ejemplo, 2000-2010 indicaría cada año entre 2000 y 2010 inclusive.

L significa "último". Cuando se usa en el campo del día de la semana, le permite especificar

construcciones como "el último viernes" ("5L") de un mes determinado. (+En el campo del día del mes, específica el último día del mes.

W está permitido para el campo del día del mes. Este personaje se usa para especificar el día de la semana (lunes a viernes) más cercano al día dado. Como ejemplo, si tuviera que especificar "15W" como valor para el campo del día del mes, el significado es: "el día de la semana más cercano al 15 del mes ". Entonces, si el 15 es un sábado, el gatillo se disparará el viernes 14. Si el 15 es domingo, el gatillo se disparará el lunes 16. Si el 15 es un martes, entonces se disparará el martes 15. Sin embargo, si especifica "1W" como valor para día del mes, y el primero es un sábado, el gatillo se disparará el lunes 3, ya que no "saltará" sobre el límite de los días de un mes. Se puede especificar el carácter 'W' solo cuando el día del mes es un solo día, no un rango o una lista de días.

Hash (#) '#' está permitido para el campo del día de la semana y debe ir seguido de un número entre uno y cinco. Le permite especificar construcciones como "el segundo viernes" de un mes determinado.

Interrogación (?) se usa en lugar de '*' para dejar el día del mes o el día de la semana en blanco. Nota: el signo de interrogación es un carácter no estándar y existe sólo en algunas implementaciones cron.

Administración de los jobs del Cron Tal y como hemos podido ver el funcionamiento es muy sencillo. Por último, falta conocer algunos comandos básicos para controlar el cron de nuestro sistema GNU/Linux, Para reemplazar el archivo existente por otro que defina el usuario se debe utilizar el siguiente comando:

```
$ crontab archivo
```

Para editar el archivo existente en la actualidad se utiliza el comando que ya hemos visto a lo largo de esta sección:

```
$ crontab -e
```

Listar todas las tareas existentes en el crontab del usuario:

```
$ crontab -l
```

Borrar el crontab que está configurado:

```
$ crontab -d
```

Definir el directorio en el que se almacenará el archivo de crontab. Para realizar esta operación se deben tener permisos de ejecución en dicho directorio:

```
$ crontab -c dir
```

Instrucción para manejar el crontab de otros usuarios existentes en el sistema:

```
$ crontab -u usuario
$ crontab -l -u root
$ crontab -e usuario2
# crontab -d -u usuario
```

Cómo podemos ver, programar la ejecución de tareas no es nada complicado y se puede realizar de forma rápida si se tiene claro todo lo mencionado aquí.

13.6 Gestión de Servicios

A diferencia de Windows, donde suele haber herramientas gráficas para casi todo, en GNU/Linux muchas de las tareas relativas a la administración del sistema suelen hacerse por consola.

Por si esto no fuera poco, uno de los rasgos característicos de GNU/Linux es que solemos disponer de varios métodos o herramientas para llegar a hacer una misma cosa (como sucede justamente con el caso que nos ocupa). Esto puede crear una cierta sensación de desorden y de dispersión al principio, pero una vez sabido la dinámica no es para nada complicada.

El caso de la gestión de servicios es algo paradigmático, sobre todo con la adopción de *systemd* por parte de la mayoría de distribuciones, que ha supuesto también la llegada de todo un nuevo conjunto de métodos para el control de servicios. Métodos que, por otro lado, siguen conviviendo aún con los que teníamos antes de la implantación masiva de *systemd*[277].

¿Qué es un Servicio en GNU/Linux? Un servicio es un programa que se ejecuta en segundo plano, fuera del control interactivo de los usuarios del sistema, ya que carece de una interfaz. Esto con el fin de proporcionar aún más seguridad, pues algunos de estos servicios son cruciales para el funcionamiento del sistema operativo.

Por otro lado, en sistemas como Unix o GNU/Linux, los servicios también se conocen como demonios (Daemons). A veces el nombre de estos servicios, o demonios, terminan con la letra d. Por ejemplo, sshd es el nombre del servicio que maneja SSH. Los servicios o procesos tienen las siguientes características:

- No acostumbran a tener una interfaz gráfica para que los usuarios puedan interactuar de forma directa con ellos.
- Los servicios o demonios normalmente son inicializados con el arranque del sistema.
- Normalmente están a la espera de que pase un evento para a posteriori poder realizar su función.
- Toda la actividad que generan los servicios queda registrada en los logs del sistema mediante syslog y/o journalctl.

[277] Existen múltiples versiones que no usan *systemd*, una basada en Debian GNU/Linux es Devuan, tiene las mismas características de Debian e incluso usa los mismos repositorios, solo que utiliza como sistema de inicio a SysVinit.

Algunos ejemplos de servicios o demonios que conoce la gran mayoría de gente son los siguientes:

- apache
- httpd
- cron
- dnsmasq
- etc

Controlar Servicios con *systemd* y *systemctl* Lo dicho, empecemos por ver los diferentes comandos que nos permiten controlar el estado de los diferentes servicios a través de *systemd*, que actualmente gestiona el proceso de arranque en gran parte de las distribuciones más comunes, incluyendo Debian, Ubuntu, Fedora, o gran parte de sus derivadas.

Systemd nace como sustituto de *System V* para controlar la secuencia de arranque y la administración de servicios en GNU/Linux, aunque no ha estado exento de controversia, debido a su complejidad e invasión de funciones de otras herramientas para la administración del sistema. Aun así, es lo que tenemos en gran parte de las distribuciones a día de hoy.

Cómo se gestionan los servicios o demonios los servicios o demonios se gestionan mediante los demonios *Init* o *Systemd*. El primer servicio que inicia el Kernel de Linux es *Init* o *Systemd*. Seguidamente, *init* o *systemd* son los encargados de cargar el resto de servicios del sistema operativo. Por lo tanto *Init* o *Systemd* son los padres de todos los demonios o servicios que se inicializan en nuestro sistema operativo.

Init y *Systemd* siempre están activos hasta que el sistema se apaga. Mientras estos servicios estén activos los podremos usar para gestionar los servicios que se inician y paran en nuestro equipo de cómputo o servidor[278].

[278] En el caso que *systemd* o *init* no se inicien, nuestro ordenador nunca podrá llegar a arrancar. Por lo tanto son demonios extremadamente importantes.

Iniciando servicios desde el arranque del sistema en muchos casos es conveniente que un servidor o servicio inicien junto con el arranque del equipo en si, por ejemplo el servidor Web *Apache* o alguna base de datos, esto es para que estén disponibles todo el tiempo y no se requiera de intervención del administrador para iniciarlos.

En GNU/Linux, a diferencia de otros sistemas operativos, es posible configurarlo en base a niveles de ejecución (run levels), cada nivel de ejecución (en la mayoría de las distros son 7), inicia o detiene (*Start* o *Kill*) ciertos servicios. Estos niveles son los siguientes:

- 0 Detener o apagar el sistema
- 1 Modo monousuario, generalmente utilizado para mantenimiento del sistema
- 2 Modo multiusuario, pero sin soporte de red
- 3 Modo multiusuario completo, con servicios de red
- 4 No se usa, puede usarse para un inicio personalizado
- 5 Modo multiusuario completo con inicio gráfico (X Window)
- 6 Modo de reinicio (reset)

Por ejemplo el nivel 0, que apaga el equipo, mata o detiene a todos los procesos del sistema, todos los servicios, lo mismo hace el 6 con la diferencia que después inicia un Script que permite reiniciar el sistema.

En el directorio */etc/* están los directorios de los servicios que se arrancan en cada uno de los diferentes niveles de ejecución:

- rc0.d/
- rc1.d/
- rc2.d/
- rc3.d/
- rc4.d/
- rc5.d/

- rc6.d/
- rcS.d/

y los servicios en: */etc/init.d/*

Cómo verificar el nivel de ejecución

```
$ systemctl get-default
```

Cómo verificar y cambiar el nivel de ejecución

```
# runlevel
```

y para forzar el cambio usamos, por ejemplo:

```
# telinit 3
```

Los diferentes objetos de control de *systemd* se denominan "unidades". Para el control de dichas unidades, *systemd* utiliza una herramienta denominada *systemctl*, que es la que utilizaremos a continuación para controlar el estado de los diferentes servicios.

Primero, debemos ser el usuario administrador *root*, para convertirnos en él, usamos:

```
$ su
```

ya disponiendo de los privilegios del administrador, lo primero que haremos es conocer todos los servicios cargados en el sistema. Para ello, nos valemos del comando:

```
# systemctl status
```

Podemos listar todos los servicios en GNU/Linux. Para hacerlo, ejecuta el comando:

```
# systemctl list-unit-files –type service
```

Cuando ejecutes el comando, verás todos los servicios que están en el sistema. Sin embargo, también verás que algunos tienen un estado definido. Veamos qué significa todo esto:

- Los servicios habilitados (Enabled) son los que se están ejecutando actualmente. Generalmente estos no tienen problemas.
- Los servicios deshabilitados (Disabled) son los que no están activos, pero se pueden activar en cualquier momento sin ningún problema.
- Los servicios enmascarados (Masked) no se ejecutarán a menos que les quites esa propiedad.
- Los servicios estáticos (Static) solo se utilizarán en caso de que otro servicio o unidad los necesite.
- Finalmente, hay servicios generados (Generated) a través de un initScript SysV o LSB con el generador systemd.

En caso de que solo necesitemos saber cuáles son los servicios que están activos, usamos:

```
# systemctl | grep running
```

Administrar servicios de Linux Cuando el equipo arranca ocurren múltiples procesos, los cuales podemos conocer, administrar y optimizar, por ejemplo:

Tiempo total de carga para conocer el tiempo total de la carga del sistema operativo (desde que inicial la carga del Kernel hasta que tenemos control sobre la interfaz gráfica), usamos:

```
# systemd-analyze
```

con la siguiente salida para mi equipo:

```
Startup finished in 3.338s (Kernel) + 2.540s (userspace) =
5.907s
graphical.target reached after 2.596s in userspace
```

Detalles de la carga si deseamos conocer de forma detallada el tiempo de carga de cada servicio del sistema operativo usamos:

```
# systemd-analyze blame
```

con la siguiente salida para mi equipo:

```
647ms dev-sda2.device
543ms plymouth-start.service
542ms apparmor.service
542ms systemd-tmpfiles-setup.service
541ms console-setup.service
529ms systemd-logind.service
491ms lightdm.service
486ms plymouth-quit-wait.service
385ms systemd-backlight@backlight:intel_backlight.service
264ms upower.service
228ms udisks2.service
200ms nftables.service
171ms ModemManager.service
157ms accounts-daemon.service
132ms systemd-journald.service
105ms systemd-udev-trigger.service
102ms NetworkManager.service
98ms user@1000.service
...
```

Detalles de la carga gráfica también se puede conocer el tiempo que necesita la interfaz gráfica para inicializar sus servicios usando:

```
# systemd-analyze critical-chain graphical.target
```

con la siguiente salida para mi equipo:

```
The time after the unit is active or started is printed after the
"@" character.
The time the unit takes to start is printed after the "+" char-
acter.
graphical.target @2.596s
```

```
+-lightdm.service @2.152s +443ms
+-systemd-user-sessions.service @2.130s +18ms
+-network.target @2.123s
+-NetworkManager.service @1.987s +135ms
+-dbus.service @1.960s
+-basic.target @1.945s
+-sockets.target @1.945s
+-avahi-daemon.socket @1.945s
+-sysinit.target @1.944s
+-sys-fs-fuse-connections.mount @6.339s +23ms
+-systemd-modules-load.service @558ms +52ms
+-systemd-journald.socket @553ms
+-system.slice @546ms
+–.slice @546ms
```

Servicios en tiempo real podemos ver por grupos el uso de recursos, tareas,CPU, memoria, entrada y salida en tiempo real, mediante el comando:

```
#systemd-cgtop
```

Con esta información en mente y dado que antes vimos cómo listar los servicios en GNU/Linux, es el momento de aprender a administrar un servicio específico. Ten en cuenta que cada servicio representa un Software y cada uno de los Softwares funciona de manera diferente. En este texto solo mostraremos cómo iniciar, verificar el estado y detener los servicios: los controles básicos.

Para listar los servicios del sistema

```
# systemctl list-unit-files
```

podemos preguntar por (enabled, disabled, masked, failed, static, etc), por ejemplo:

```
# systemctl list-unit-files –state failed
```

permite enumerar los archivos de la unidad de punto de montaje:

```
# systemctl list-unit-files -t mount
```

además muestra los servicios instalados en el sistema, estén o no activos:

```
# systemctl list-unit-files -t service
```

pero sobre todo, listar los servicios que nunca se usaron:

```
# systemctl –all -t service
```

Para saber si un servicio está activo, preguntamos usando:

```
# systemctl is-enabled [nombre_servicio]
```

Para iniciar un servicio en Linux, necesitas ejecutar el siguiente comando:

```
# systemctl start [nombre_servicio]
```

si el servicio está configurado correctamente, se iniciará. Ahora, si queremos detenerlo, usamos:

```
# systemctl stop [nombre_servicio]
```

Por otro lado, para verificar el estado de un servicio, usamos:

```
# systemctl status [nombre_servicio]
```

para conocer las propiedades de un servicio incluyendo su configuración, usamos:

```
# systemctl show [nombre_servicio]
```

y para reiniciar un servicio, usamos:

```
# systemctl restart [nombre_servicio]
```

Para que un servicio arranque de forma automática al iniciar el equipo de cómputo tenemos que habilitarlo. A modo de ejemplo, para habilitar el servicio *bluetooth*, ejecutamos:

```
# systemctl enable bluetooth.service
```

Si queremos que al arrancar el equipo de cómputo no se cargue un servicio[279] ejecutamos:

```
# systemctl disable bluetooth.service
```

Aunque el servicio esté deshabilitado, cualquier usuario con permisos de administrador lo podrá iniciar de forma manual una vez iniciada la sesión.

Para ver el árbol de dependencias de un servicio o una unidad, por ejemplo para el servicio *cron* necesita para realizar su función tenemos que ejecutar:

```
# systemctl list-dependencies cron
```

pero podemos usar la opción –all, para expandirse sobre las otras unidades también:

```
# systemctl list-dependencies -all cron
```

Para conocer la totalidad de servicios que se usarán después de iniciar un servicio por ejemplo para el servicio *cron*, usamos:

```
# systemctl list-dependencies –before cron
```

Para saber los servicios y unidades que tienen que estar activos antes de ejecutar un servicio, por ejemplo para el servicio *cron,* usamos:

```
# systemctl list-dependencies –after cron
```

Al cambiar la configuración de cualquier servicio no se aplicará de forma instantánea. Para que se aplique la nueva configuración tenemos 3 opciones:

- Reiniciar el equipo de cómputo o servidor.
- Reiniciar el servicio.
- Recargar la configuración del servicio sin reiniciarlo ni pararlo en ningún momento.

[279] Hay que tener precaución al deshabilitar servicios. Si no saben que realiza el servicio que deshabilitan pueden presentarse inconvenientes graves.

Para recargar la configuración de un servicio sin llegar a pararlo ni a reiniciarlo, por ejemplo para el servicio *bluetooth* tenemos que ejecutar:

```
# systemctl reload bluetooth.service
```

al ejecutar el comando se aplicarán los cambios realizados en los archivos de configuración del servicio sin tan siquiera llegar a parar el servicio.

Hemos visto ya cómo deshabilitar un servicio. Si además de deshabilitarlo queremos que no se pueda iniciar manualmente ni automáticamente después de iniciar la sesión podemos enmascararlo.

Para enmascarar un servicio por ejemplo para *bluetooth.service* ejecutamos:

```
# systemctl mask bluetooth.service
```

De este modo nadie podrá iniciar el proceso manualmente o automáticamente a no ser que lo desenmascare previamente.

Para desenmascarar un servicio, usamos:

```
# systemctl unmask bluetooth.service
```

journalctl para ver los mensajes que generan los registros del sistema -Journal o Logs-, usamos:

```
# journalctl
```

para conocer los registros del sistema en tiempo real, usamos:

```
# journalctl -f
```

para conocer los registros del sistema en tiempo real pero de un servicio concreto, usamos:

```
# journalctl -f sshd.service
```

podemos conocer sólo los mensajes del Kernel, usando:

```
# journalctl -k
```

o ver los registro del actual boot, mediante:

```
# journalctl -b
```

para ver los mensajes de hace cinco arranques, usamos:

```
# journalctl -k -b -5
```

para ver los registros de un determinado proceso, usamos:

```
# journalctl _PID=3829
```

para ver los registros de un determinado usuario, usamos:

```
# journalctl _UID=1001
```

para ver los registros de un determinado programa, usamos:

```
# journalctl _COMM=vmtoolsd
```

si deseamos verlos sin paginación, usamos:

```
# journalctl –no-pager
```

si necesitamos verlos en orden cronológico inverso, usamos:

```
# journalctl -r
```

o solo ver los últimos N recientes líneas, usamos:

```
# journalctl -n 25
```

podemos filtrar los registros para un cierto intervalo de tiempo, mediante:

```
# journalctl –since=yesterday –until=now
```

o podemos especificar el día, mediante:

```
# journalctl –since "2020-12-01"
```

o especificar el día y hora, mediante:

```
# journalctl –since "2020-12-01 10:00:00" –until "2020-12-10"
```

pero podemos filtrar sobre aquellos que dan por ejemplo mensaje de error, usando:

```
# journalctl | grep -i error
```

para conocer el estatus histórico de un servicio, usamos:

```
# journalctl -u sshd.service
```

o especificar la fecha en tiempo UTC y conocer el estatus histórico de un servicio, usamos:

```
# journalctl -u sshd.service –since=yesterday –utc
```

y podemos listar los eventos de múltiple servicios usando:

```
# journalctl -u sshd.service -u vsftpd.service
```

también es posible filtrar por prioridad, mediante:

```
# journalctl -p 3 -x
```

donde las prioridades son: emerg(0), alert(1), crit (2), err (3), warning (4), notice (5), info (6) y debug(7).
podemos conocer cuánto ocupan en disco los registros del sistema, usando:

```
# journalctl –disk-usage
```

y forzar a que se roten los registros, mediante:

```
#journalctl –rotate
```

o pedir que se purguen los registros anteriores a cierta cantidad de días, por ejemplo dos días:

```
# journalctl –vacuum-time=2d
```

se puede restringir el tamaño de los registros a cierta cantidad de espacio, por ejemplo 100 MB:

```
# journalctl –vacuum-size=100M
```

si se quiere hacer permanente, se debe editar el archivo */etc/systemd/journald.conf* y especificar los cambios en el.

dmseg para ver los mensajes de diagnóstico del sistema, usamos:

```
# dsmeg
```

pero podemos filtrar sobre aquellos que dan por ejemplo mensaje de error, usando:

```
# dsmeg | grep -i error
```

pero podemos buscar cosas como: sda, usb, dma, memory, etc.

Controlar Servicios Utilizando Init.d Veamos cómo podemos hacer lo mismo pero echando mano de *Init.d*, que básicamente era el método que se utilizaba antes, cuando teníamos *System V* como gestor de arranque. La diferencia más importante con la que nos encontramos aquí es que para invocar un determinado servicio, lo que hacemos es especificar la ruta en la que se encuentra, dentro del directorio init.d, seguido de la orden concreta (*status, start, stop*, etc.).

Para comprobar el estado de un determinado servicio, en este caso *cron*, lo que hacemos es indicar la ruta del fichero de configuración dentro del directorio /etc/*init.d*, seguido de la orden *status*. Puesto en práctica quedaría del siguiente modo:

```
# /ent/init.d/cron status
```

Para iniciar el servicio, simplemente seguimos con el mismo criterio, pero cambiando la orden del final.

```
# /etc/init.d/cron start
```

Para detener el servicio, lo mismo. Al igual que en los otros casos, esto detiene la ejecución actual del servicio, pero no modifica su estado en el arranque del sistema.

```
# /etc/init.d/cron stop
```

Y lo mismo con reiniciar el servicio, simplemente cambiando la orden del final. Como en los otros casos, la orden de reiniciar el servicio es la que necesitas utilizar cada vez que realices algún cambio en el fichero de configuración asociado.

```
# /etc/init.d/cron restart
```

Controlar Servicios Utilizando Chkservice Por último veamos cómo podemos hacer lo mismo pero echando mano de *chkservice*, que es una forma de interactuar con *systemctl* para iniciar, parar, reiniciar y ver el estado de los servicios del sistema desde una ventana en modo de línea de comandos, en la cual usando las teclas podemos navegar y modificar las diferentes opciones que nos permite interactuar con los servicios del sistema. Para instalar el paquete usamos:

```
# apt install chkservice
```

y lo corremos usando:

```
# chkservice
```

13.7 Escaneo de Puertos

Antes de ver la forma de hacer un escaneo de puertos, debemos conocer algunas cosas como:

¿Qué es un puerto? es un pequeño código que se utiliza como punto de acoplamiento en nuestra máquina desde el cual podemos comunicarnos remotamente con otra máquina.

¿Qué es un puerto de Hardware? es un punto de conexión en modo periférico físico a una máquina desde otro dispositivo.

¿Qué es un Socket? se denomina Socket a la combinación de puerto de Software y dirección IP.

¿Cuántos puertos existen el Linux? el rango de puertos va de 0 a 65,535, por tanto tenemos 65,536 puertos.

¿Por qué solo tenemos 65,536 puertos? esto se debe a la limitación TCP/IP, donde cada número de puerto tiene un tamaño de solo 16 bits. Esto equivale a 2^{16}.

¿Qué puertos son los predeterminados? los puertos predeterminados y más utilizados, van del 0 a 1,023 (2^{10} puertos). Otras he-rramientas usan el resto de los puertos.

¿Qué es un puerto predeterminado? es un puerto designado para un servicio particular, como servidor web (80), servidor de correo (25), servidor SSH (22), etc.

¿Es posible modificar un puerto predeterminado? la respuesta es clara, si se puede. Tan solo debemos modificar el puerto de escucha en el archivo de configuración del servicio que nos interese.

¿Cuántos números de protocolos existen en TCP/UDP? hay que tener cuidado en no confundir los protocolos con los números de puerto, en TCP hay 6 y en UDP hay 17.

¿Dónde podemos ver información sobre los puertos? para ello, podemos hacer:

```
$ cat /etc/services
```

Diferencias entre TCP y UDP Cuando hablamos de protocolos de internet en tráfico, los usuarios pueden elegir entre una configuración TCP o UDP. Las características y funciones de TCP vs UDP son diferentes, cada protocolo tiene sus ventajas, desventajas y posibles problemas.

Dicho esto, UDP es mucho más rápido, aún así muchos sistemas siguen dependiendo de TCP para descargar paquetes de datos. En este artículo echaremos un vistazo a los dos protocolos, pero recordemos que antes de decidirnos por uno u otro, deberemos conocer en profundidad nuestras necesidades.

El protocolo de control de transmisión (TCP) está orientado a la conexión, esto quiere decir que una vez que se establece la conexión, los datos se transmiten en dos direcciones. Este protocolo tiene la capacidad de verificar posibles errores, esta fórmula nos garantiza que los datos se entreguen en el orden enviado.

Dicho lo anterior, TCP es el protocolo perfecto para transferir información relacionada con páginas Web, imágenes fijas y archivos de datos. El uso de la retroalimentación propia del protocolo genera una sobrecarga en la red, que se traduce en un mayor consumo de ancho de banda.

El protocolo de datagramas de usuario (UDP) es un protocolo de internet mucho más simple. No requiere de servicios de recuperación y verificación de errores. Tampoco existe consumo extra al abrir una conexión, mantenerla abierta o terminarla; Los datos se envían de forma continua al destinatario, independientemente de si los recibe o no.

Además, UDP admite el flujo de paquetes constante, es la gran diferencia sobre TCP. La conexión TCP, está obligada a reconocer un conjunto de paquetes (sea confiable o no), por lo tanto, se genera una retransmisión en cada reconocimiento cuyo resultado sea la pérdida de paquetes. El protocolo UDP evita estos consumos, por tanto, el efecto-resultado nos aporta una velocidad mucho más eficiente si hablamos de ancho de banda.

Las aplicaciones Web y de escritorio (de comunicación) priorizan UDP sobre TCP para el transporte de medios en tiempo real.

La herramienta nmap (Abreviatura de Network Mapper) Es la mejor de la línea de comandos en Linux (también disponible para *Windows* y *Mac OS X*) para realizar funciones de auditoría y seguridad de redes, rastreo y análisis en busca de sistemas para elaborar un inventario de red, etc. Todo esto de forma gratuita y bajo licencia Open Source. Lanzando un escaneo de puertos de red (1-65,535) TCP y UDP que corresponden a uno o más servicios que se corren en un sistema o servidor. Con nmap podemos visualizar una gran cantidad de información: Hosts activos en la red, sistema operativo que están ejecutando, puertos y servicios abiertos a través de la red, tipos de *Firewall* que están utilizando. Para los recelosos de la línea de comandos, existe la opción de utilizar nmap con GUI, para ello está *Zenmap*.

Este comando puede ser una valiosa herramienta de diagnóstico para los administradores de redes, mientras que también puede ser una herramienta de reconocimiento potente para la comunidad[280] *Black-hat (Hackers, Crackers, Script Kiddies, etc.)*.

[280] El escaneo lo puede hacer cualquier usuario sin privilegios de administrador, pero hay algunas opciones reservadas para el usuario *root*.

¿Por qué escanear puertos contribuye a la seguridad de tu sistema? los puertos desempeñan una función destacada a la hora de que los paquetes de datos encuentren el camino para alcanzar los objetivos deseados. En este sentido, funcionan como interfaz entre equipos de cómputo y servicios o programas del sistema y son usados por los protocolos de red como TCP y UDP. En combinación con la dirección IP, los puertos permiten al sistema operativo no solo saber a qué equipo de cómputo ha de enviar el flujo de datos sino también a qué aplicación ha de entregar los paquetes.

Curiosidades sobre los puertos A cada puerto se le asigna un número comprendido entre el 0 y el 65,535[281]. A este respecto, pueden diferenciarse tres áreas:

- Los puertos que van desde el número 0 al 1023 reciben la denominación de puertos estandarizados y la internet Assigned Numbers Authority (IANA) asigna protocolos fijos y otros recursos a la mayoría de ellos. De esta manera, por ejemplo, se ha reservado el puerto 80 para las conexiones HTTP y, por lo tanto, este se convierte en el puerto decisivo para las solicitudes realizadas a través del servidor Web.
- Los números de puerto que van desde el 1,024 hasta el 49,151 están destinados normalmente a servicios registrados, aunque se asignan también a clientes, en especial por parte de GNU/Linux.
- Los puertos comprendidos entre los números 49,152 y 65,535 son los que otorgan los sistemas operativos a los programas cliente de manera dinámica.

Para establecer una conexión a través de un puerto determinado este debe estar abierto, es decir, libre. Especialmente en lo referente a la transferencia de datos en internet, esto indica, como es lógico, que hay un gran número

[281] En Debian GNU/Linux se puede conocer los puertos TCP y UDP de los principales servicios, mediante:

```
$ cat /etc/services
```

y la dirección IP de la maquina que uso:

```
$ ip address
```

de puertos abiertos, lo que lleva aparejado ciertos problemas: cada puerto abierto se convierte en un posible acceso para atacantes, en caso de que la aplicación correspondiente presente brechas de seguridad. Por este motivo, es necesario tener siempre en mente cuáles son los puertos que están abiertos en tu sistema y cuáles son las aplicaciones que hay detrás del tráfico de datos. Los escáneres de puertos son los mejores recursos con los que puedes detectar la presencia de puertos activos.

¿Qué es el escaneo de puertos? por escaneo de puertos se entiende aquel procedimiento que tiene como objetivo analizar los puertos abiertos de un sistema informático con la ayuda de herramientas especiales. Para poder llevar a cabo dicho escaneo, no es necesario registrarse en el sistema de destino, sino solo estar conectado a él, por ejemplo, a través de una red local o de internet. Con ayuda de los analizadores de puertos (*Port Scanners)* se envían, a modo de prueba, paquetes de datos especiales a los diferentes puertos y se obtienen las correspondientes respuestas o mensajes de error que la herramienta analiza y evalúa. Independientemente de la funcionalidad del programa de Port Scanning que se use, no solo se pueden obtener datos acerca de cuáles son los puertos abiertos o cerrados, sino también sobre el sistema operativo que usa el equipo de destino, sobre el tiempo que hace que el PC permanece encendido o sobre los servicios o aplicaciones que utilizan los puertos correspondientes.

El escaneo de puertos representa un medio muy eficiente al que los administradores de sistemas pueden recurrir para controlar el tráfico de datos de una red y para filtrar sus posibles debilidades. En algunos casos determinados, también se pueden solucionar problemas de red concretos. Debido a que las herramientas no tienen una influencia significativa en la capacidad de rendimiento de los sistemas que se examinan, pueden utilizarse sin vacilaciones para dichas medidas de seguridad. También en el ámbito doméstico puede ser de utilidad el método del escáner de puertos, y es que en cuanto se instalan y se usan aplicaciones que requieren una conexión a internet, también se abren puertos automáticamente, siempre que el Firewall no lo impida.

El escaneo de puertos ayuda a mantener una visión general y muestra los puertos que ya han dejado de ser necesarios y que, en consecuencia, pueden cerrarse para minimizar los riesgos que pueden afectar a la seguridad. Pero hay que recordar siempre, que todos los resultados obtenidos por las her-

ramientas automáticas deben ser revisados por un profesional en Hacking Ético y Pruebas de Penetración. Pues casi siempre existen falsos positivos.

¿Cómo funciona el escaneo de puertos exactamente? para escanear puertos existen normalmente diferentes métodos, la mayoría de los cuales gira en torno al protocolo de conexión TCP. Para comprender cuáles son los procesos básicos que tienen lugar mediante el escaneo de puertos, es de utilidad echar un vistazo a los aspectos generales del establecimiento de la conexión mediante el protocolo TCP:

- En el marco del proceso denominado negociación en tres pasos (3-Way Handshake), el cliente envía, en primer lugar, un paquete SYN (Synchronize = "sincronizar") al puerto de destino correspondiente.
- Si este consigue llegar hasta una aplicación, recibe un paquete SYN/ACK combinado (Synchronize Acknowledge = "confirmar sincronización") que confirma el establecimiento de la conexión.
- El cliente responde en el tercer y último paso con un paqueteACK (Acknowledge = "confirmar"), por lo que se establece la conexión y ya puede comenzar el intercambio de datos.

Si se establece el contacto con un puerto cerrado, el cliente recibe en el segundo paso un paquete con el flag RST (reset = "restablecer") como respuesta, por lo que se interrumpe la negociación.

Puesto que el intercambio de datos con las diversas aplicaciones resultaría por un lado muy costoso y por otro muy complejo, el escáner de puertos solo está limitado a un sencillo intento de conexión, como ponen de relieve los métodos de escaneo que te presentamos a continuación.

TCP SYN en este caso se puede hablar de un escaneo medio abierto, ya que este no tiene como objetivo el establecimiento de una conexión *TCP* completa. En esta modalidad se pueden enviar paquetes *SYN* habituales a cada uno de los puertos con el port scanner, tras lo que se espera una respuesta por parte del Host de destino. Si este responde con un paquete *SYN/ACK*, esto indica que el puerto correspondiente está abierto y que es posible establecer la conexión. Si la respuesta consiste en un paquete *RST*, esto indicará que el puerto está cerrado. Si el Host de destino no ha enviado

todavía una respuesta, todo indica que se ha interpuesto un filtro de paquetes, como, por ejemplo, un Firewall. Los escaneos *TCP SYN* no son visibles para las aplicaciones revisadas y no generan, por lo tanto, datos de registro, de ahí que también reciben el nombre de stealth scans ("escaneos sigilosos").

TCP connect si haces uso del escaneo de puertos para llevar a cabo un sondeo del tipo *connect*, en este caso no generas, ni envias los paquetes de datos motu proprio, sino que recurres para ello a la llamada al sistema *connect*. Esta está disponible para casi todos los sistemas operativos y, por ejemplo, el navegador Web también hace uso de ella para establecer la conexión con un servidor. Esta herramienta de escaneo no está implicada en el establecimiento de la conexión, sino que es el sistema operativo el encargado de ello, el cual puede o bien crear una conexión con éxito y confirmar, por lo tanto, que el puerto está abierto o fracasar en el intento e indicar que el puerto correspondiente es un puerto cerrado. Si se establece la conexión por completo, en los archivos de registro de las aplicaciones con puertos abiertos se ve fácilmente si se utilizó este método de sondeo, pero no qué programas de filtrado se utilizan. Sin embargo, si careces de los derechos para enviar paquetes de datos en bruto, el método del *TCP* connect es una alternativa útil al escaneo *SYN*.

TCP FIN, Xmas y Null con estos tres métodos de escaneo de puertos también se puede diferenciar entre los puertos abiertos y los cerrados. Para ello se pueden aplicar los dos principios básicos registrados en las *RFC* (Request For Comments) del TCP. Por un lado, un puerto cerrado siempre tiene que responder a los paquetes entrantes que no sean paquetes *RST* con un paquete *RST* propio. Por otro lado, el puerto abierto tiene que ignorar todos los paquetes no marcados como *SYN, RST* o *ACK*. Los tres métodos de escaneo anteriormente mencionados se hacen eco de esta situación a la hora de sondear sistemas de conformidad con las publicaciones *Request For Comment* con sus paquetes individuales:

- El escaneo *Null* no coloca ninguna marca especial.
- En el *FIN*, el port scanner envía paquetes *FIN* (finish= terminar).
- Los escaneos *Xmas* combinan las marcas *FIN*, *PSH* (push= empujar) y *URG* (urgent= urgente), por lo que "iluminan" el paquete del mismo modo en que lo hace un árbol de Navidad.

Estos tres métodos se comportan exactamente de la misma manera. Los paquetes de prueba que se envían se ocupan de que, a causa de las disposiciones de *RFC*, un puerto cerrado responda con un paquete *RST* y de que un puerto abierto no muestra ninguna reacción por su parte. No obstante, debido a que sólo algunos Routers transmiten mensajes de error cuando se filtra un puerto, también puede darse una ausencia de reacción en el caso de un puerto filtrado. Aunque estos procedimientos resultan más discretos que los escaneos *SYN*, estos tienen la desventaja de que no funcionan cuando los sistemas no se ciñen exactamente al protocolo *RFC*. Windows se constituiría en este caso como el representante más importante.

UDP en los escaneos *UDP* se envían encabezados *UDP* vacíos y sin datos a todos los puertos de destino. Si un servicio responde con un paquete *UDP*, queda confirmado que el puerto que le pertenece está abierto. Si el Router envía al port scanner el mensaje de error "Port unreachable" (Type 3, Code 3), este sabrá que el puerto está cerrado. Otros tipos de mensajes de error informan de que un filtro de paquetes bloquea el puerto. El gran problema que se deriva de escanear puertos con *UDP* es que requiere mucho tiempo, ya que en numerosos sistemas, la tarea de emitir los correspondientes mensajes de error puede tardar mucho tiempo por motivos de seguridad y los puertos abiertos solo responden de forma muy irregular. El núcleo de Linux limita el número de mensajes a, por ejemplo, uno por segundo, lo que significa que se pueden escanear 65.535 puertos en unas 18 horas.

Por qué el escaneo de puertos no es siempre legal nmap no solo es popular entre los usuarios de equipos de cómputo, sino también en el gremio ciberdelincuentes. El control de los puertos no siempre se realiza de manera legal. Si se concluye en última instancia con un intento de ataque, como puede ser lo que se conoce en el lenguaje técnico como aprovechamiento de una brecha de seguridad, se puede incurrir en actos punibles. Algo menos clara parece la situación legal cuando, por ejemplo, se paraliza un sistema informático debido a un escaneo de puertos intensivo. Ya que los métodos de control pueden suponer una gran carga para el sistema de destino debido a la alta frecuencia de peticiones de conexión, esto puede ocasionar, entre otras consecuencias, el bloqueo del sistema.

Además, también es posible que los responsables del sistema de destino se den cuenta de los planes antes del fallo general y lo consideren como un primer

paso para llevar a cabo el ataque. En este sentido, no pueden perderse de vista las consecuencias legales. En caso de provocar una sobrecarga en el sistema externo de manera intencionada se hablaría de los famosos ataques *DoS* y *DDoS*, que casi con total seguridad pueden incurrir en un procesamiento penal.

De la información anteriormente expuesta se deduce que es conveniente asegurarse de que se tiene la autorización para realizar el denominado escaneo de puertos en el sistema correspondiente. También es aconsejable usar esta técnica únicamente por motivos de seguridad y no por pura curiosidad. Las medidas mostradas para poner en marcha el escaneo de puertos ponen de relieve la importancia de no perder de vista los puertos tanto del propio sistema o del equipo de cómputo de red como de los servicios a los que se puede acceder a través de ellos.

Instalación de nmap para su instalación, usamos:

```
# apt install nmap
```

también en caso de ser necesario podemos instalar una interfase gráfica para *nmap* usando:

```
# apt install nmapsi4
```

Ejemplos de Escaneo de Puertos la herramienta nmap ofrece varios métodos para analizar un sistema. En este ejemplo, realizó una exploración utilizando el nombre de Host para averiguar todos los puertos abiertos[282], servicios y la dirección MAC del sistema.

Si es nuestra propia máquina, primero debemos conocer el *IP* de ella, para ello usamos el comando *ip*, mediante:

```
$ ip address
```

con esta dirección *IP* (supongamos 192.168.0.2), podemos ver los puertos de la máquina que tiene abiertos hacia el exterior, usando:

```
$ nmap 192.168.0.2
```

[282] En ciertas redes no está permitido hacer revisión de puertos, pues se considera un ataque a los equipos que la integran, llegando a deshabilitar de la red al equipo que lo realizó.

y podemos ver además los puertos que tenemos abiertos para uso interno, mediante:

```
$ nmap localHost
```

El escaneo lo puede hacer cualquier usuario sin privilegios de administrador, pero hay algunas opciones reservadas para el usuario *root*. Pero por las repercusiones que puede tener el escaneo a equipos de la red, es mejor que se haga por un usuario avanzado.

Si necesitamos hacer escaneo a otros equipos, entonces podemos usar el nombre de Host:

```
# nmap www.mi-maquina.com
```

o bien escanear mediante direcciones *IP*:

```
# nmap 192.168.0.101
```

Se puede ver que el siguiente comando con la opción "-v" está dando una información más detallada acerca de la máquina remota:

```
# nmap -v www.mi-maquina.com
```

Puedes escanear múltiples Hosts simplemente escribiendo sus direcciones *IP* o nombres de Host con nmap:

```
# nmap 192.168.0.101 192.168.0.102 192.168.0.103
```

Se puede escanear toda una subred o rango de direcciones *IP* con nmap proporcionando el comodín * con el:

```
# nmap 192.168.0.*
```

o

```
# nmap 192.168.0.0/24
```

podemos excluir alguna *IP*, mediante:

```
# nmap 192.168.0.* –exclude 192.168.0.20
```

o múltiples *IP*, mediante:

```
# nmap 192.168.0.* –exclude direcciones.txt
```

Puede llevar a cabo exploraciones en varias direcciones *IP* simplemente especificando el último octeto de la dirección *IP*. Por ejemplo, aquí se realiza un análisis de las direcciones *IP* 192.168.0.101, 192.168.0.102 y 192.168.0.103, ejemplo:

```
# nmap 192.168.0.101,102,103
```

o

```
# nmap 192.168.2.101-103
```

Si se tiene más Hosts para escanear y todos los detalles de acogida están escritos en un archivo, puede hacer que nmap directamente lea ese archivo y realice el análisis. Primero creamos un archivo de texto, por ejemplo "test.txt" y definir todas las direcciones *IP* o nombre de Host del servidor que desea hacer una exploración.

```
# cat test.txt
localHost
www.mi-maquina.com
192.168.0.11
```

A continuación, ejecutamos:

```
# nmap -iL test.txt
```

Se puede especificar un rango de direcciones *IP* en el desempeño de escaneo con nmap:

```
# nmap 192.168.0.101-110
```

Podemos excluir algunos Hosts al realizar una exploración de red completa o cuando se escanea con los comodines con la opción de "-exclude":

```
# nmap 192.168.0.* –exclude 192.168.0.100
```

Con nmap, puede detectar qué sistema operativo y la versión que está ejecutando en el Host remoto. Para habilitar la detección de sistema operativo y versión, la exploración de la escritura y la ruta de seguimiento, podemos usar la opción "-A" con nmap:

```
# nmap -A 192.168.0.101
```

Utilizamos la opción "-O" y "-osscan-guess" también ayuda a descubrir la información del sistema operativo:

```
# nmap -O www.mi-maquina.com
```

El siguiente comando realizará una búsqueda en un Host remoto para detectar si los filtros de paquetes o Firewall se utiliza por el anfitrión:

```
# nmap -sA 192.168.0.101
```

Para escanear un Host si está protegido por ningún Software de filtrado de paquetes o cortafuegos:

```
# nmap -PN 192.168.0.101
```

Con la ayuda de la opción "-sP" simplemente podemos comprobar qué Hosts están vivos y en red, con esta opción se salta nmap detección de puertos y otras cosas:

```
# nmap -sP 192.168.0.*
```

Se puede realizar un análisis rápido con la opción "-F" para las exploraciones para los puertos que figuran en los archivos de nmap-services y deja todos los demás puertos:

```
# nmap -F 192.168.0.101
```

Utilizamos el indicador "-r" para hacerlo no de forma aleatoria:

```
# nmap -r 192.168.0.101
```

Hay varias opciones para descubrir los puertos en la máquina remota con nmap. Se puede especificar el puerto que desee nmap para escanear con la opción "-p", por defecto escanea nmap sólo puertos *TCP*:

```
# nmap -p 80 www.mi-maquina.com
```

También puede especificar los tipos de puertos y los números con nmap para escanear:

```
# nmap -p T:8888,80 www.mi-maquina.com
```

Escanear un puerto UDP:

```
# nmap -sU 53 www.mi-maquina.com
```

Se puede escanear múltiples puertos usando la opción -p:

```
# nmap -p 80,443 192.168.0.101
```

Puede escanear puertos con rangos utilizando expresiones -todos los puertos (1-65535) o rango de puertos 80-160-:

```
# nmap -p 80-160 192.168.0.101
```

Podemos encontrar versiones de servicios que se ejecutan en máquinas remotas con la opción "-sV":

```
# nmap -sV 192.168.0.101
```

A veces, los cortafuegos de filtrado de paquetes bloquea las solicitudes de *ping ICMP* estándar, en ese caso, podemos utilizar métodos *TCP ACK* y *TCP Syn* para escanear Hosts remotos:

```
# nmap -PS 192.168.0.101
```

Analizar Host remoto para puertos específicos con *TCP ACK*:

```
# nmap -PA -p 22,80 192.168.0.101
```

Analizar Host remoto para puertos específicos con *TCP Syn*:

```
# nmap -PS -p 22,80 192.168.0.101
```

Realizar un escaneo sigiloso:

```
# nmap -sS 192.168.0.101
```

Comprobar más puertos utilizando *TCP Syn*:

```
# nmap -sT 192.168.0.101
```

Realizar un análisis tcp nula para engañar a un servidor de seguridad:

```
# nmap -sN 192.168.0.101
```

Hay algunas formas de implementar el descubrimiento de Host a través de nmap. El más común de los cuales es a través de -sL. Por ejemplo:

```
# nmap -sL 192.168.0.1
```

IPv6 se está volviendo más común, y nmap lo admite del mismo modo que admite dominios y direcciones *IP* anteriores. *IPv6* funciona con cualquiera de los comandos nmap disponibles. Sin embargo, se requiere un indicador para indicar a nmap que se hace referencia a una dirección *IPv6*:

```
# nmap -6 ::ffff:c0a8:6
```

Puede ser necesario encontrar interfaces de Host, interfaces de impresión y rutas para depurar. Para hacer esto, use el comando iflist:

```
# nmap –iflist
```

Del mismo modo, "-packet-trace" mostrará los paquetes enviados y recibidos, proporcionando un valor similar para la depuración:

```
# nmap –packet-trace
```

A veces es posible que deba escanear de manera más agresiva o que desee ejecutar un escaneo rápido. Puede controlar esto mediante el uso de los mecanismos de sincronización. En nmap, el tiempo controla tanto la velocidad como la profundidad de la exploración:

```
# nmap -T5 192.168.0.1
```

Una exploración agresiva será más rápida, pero también podría ser más disruptiva y también imprecisa. Hay otras opciones, como los escaneos T1, T2, T3 y T4. Para la mayoría de los escaneos, los tiempos T3 y T4 serán suficientes.

A continuación, tienes un completo listado de las órdenes avanzadas que podremos utilizar y para qué sirven cada una de ellas:

- -PS n (envía un *TCP SYN* al puerto 80 por defecto para descubrir Hosts levantados, «n» puede ser otro puerto o puertos a probar)
- -PA n (envía un *TCP ACK* al puerto 80 por defecto para descubrir Hosts levantados, «n» puede ser otro puerto o puertos a probar)
- -PU n (envía un datagrama *UDP* al puerto 40125 por defecto para descubrir Hosts levantados, «n» puede ser otro puerto o puertos a probar)
- -sL (no escanea, únicamente lista los objetivos)
- -PO (ping por protocolo)
- -PN (No hacer ping)
- -n (no hacer *DNS*)
- -R (Resolver *DNS* en todos los sistemas objetivo)
- -traceroute (trazar ruta al sistema (para topologías de red))
- -sP (realizar ping, igual que con -PP -PM -PS443 -PA80)

La ejecución se realiza de la siguiente forma:

```
# nmap 192.168.1.1-20 -PS
```

El programa nmap nos permite realizar escaneo de puertos avanzados, enviando diferentes tipos de paquetes *TCP* y *UDP* entre otros, para descubrir que un puerto está abierto, filtrado o cerrado. Estas órdenes son fundamentales para comprobar cómo tienen los Hosts un puerto o varios puertos:

- -sS (análisis de puertos enviando paquetes *TCP SYN*)
- -sT (análisis de puertos enviando paquetes *TCP CONNECT*)
- -sA (análisis de puertos enviando paquetes *TCP ACK*)
- -sW (análisis de puertos enviando paquetes *TCP Window*)
- -sU (análisis de puertos enviando paquetes *UDP*)
- -sY (análisis de puertos enviando paquetes *SCTP INIT*)
- -sZ (análisis de puertos enviando paquetes *COOKIE ECHO* de *SCTP*)
- -sO (análisis de puertos enviando paquetes *IP* directamente)
- -sN (análisis de puertos enviando paquetes *TCP Null Scan*)
- -sF (análisis de puertos enviando paquetes *TCP FIN Scan*)
- -sX (análisis de puertos enviando paquetes *TCP Xmas Scan*)

nmap nos permite acelerar el escaneo de los diferentes puertos, aunque si lo hacemos demasiado rápido, es posible que puertos que realmente estén abiertos los marque como cerrados, es decir, no es recomendable hacer los escaneos de manera muy rápida. Si utilizamos el flag «-TX» siendo X un número entre 0 y 5, podremos configurar la velocidad del escaneo:

- -T0 paranoico
- -T1 sigiloso
- -T2 sofisticado
- -T3 normal
- -T4 agresivo
- -T5 locura

Este programa también nos permite paralelizar el escaneo de los diferentes puertos de los Hosts, para ello podemos paralelizarlo a un grupo de Hosts, y también nos permitirá enviar de manera simultánea diferentes paquetes:

- -min-Hostgroup
- -max-Hostgroup
- -min-parallelism
- -max-parallelism

Otras opciones que tenemos es la posibilidad de enviar paquetes no más lentos (-min-rate) que un determinado número, ni más rápido (-max-rate) que un determinado número. Esto es ideal para no colapsar un determinado Host y que un IDS pueda bloquearnos el acceso. También podemos configurar el *RTT* «Round trip time», en este caso tendremos hasta tres argumentos que podremos utilizar:

- -min-rtt-timeout
- -max-rtt-timeout
- -initial-rtt-timeout

También tenemos la opción de limitar a un máximo de reintentos el envío de paquetes a un determinado puerto de un Host, el argumento a utilizar es «-max-retries» y es muy útil para no «colapsar» un puerto, o que un *IDS salte* y nos bloquee.

- -max-retries

nmap es un programa tan potente que también nos va a permitir detectar la versión de los diferentes servicios que tenemos en el sistema, de hecho, es capaz de intentar adivinar qué sistema operativo está utilizando un Host remoto, con el objetivo de realizar posteriormente un pentesting. En esta sección tenemos unos argumentos muy interesantes:

- -O (habilitar la detección del sistema operativo)
- -sV (detección de la versión de servicios)
- -max-os-tries (establecer número máximo de intentos contra el sistema objetivo)

En la gran mayoría de redes empresariales tenemos tanto Firewalls como sistema de detección y prevención de intrusiones. Es posible intentar engañar a estos sistemas, realizando diferentes técnicas con nmap, algunos ejemplos son los siguientes:

- -f (fragmentar paquetes)
- -D d1,d2 (encubrir análisis con señuelos)
- -S ip (falsear dirección origen)
- -g source (falsear puerto origen)
- -randomize-Hosts orden
- -spoof-mac mac (cambiar *MAC* de origen)

Otros parámetros (incrementar verbose y más)

- -v (Incrementar el nivel de detalle del escaneo)
- -d (1-9) establecer nivel de depuración
- -packet-trace ruta de paquetes
- -resume file continuar análisis abortado (tomando formatos de salida con -oN o -oG)
- -6 activar análisis IPV6
- -A agresivo, igual que con -O -sV -sC -traceroute

Opciones interactivas (que se pueden ejecutar mientras está realizando el análisis)

- v/V aumentar/disminuir nivel de detalle del análisis
- d/D aumentar/disminuir nivel de depuración
- p/P activar/desactivar traza de paquetes

Scripts

- -sC realizar análisis con los Scripts por defecto
- -Script file ejecutar Script (o todos)
- -Script-args n=v proporcionar argumentos
- -Script-trace mostrar comunicación entrante y saliente

Formatos de salida

- -oN normal
- -oX *XML*
- -oG programable
- -oA todos

Identificar Vulnerabilidades utilizando NSE los NSE (Nmap Scripting Engine) son una de las características más poderosas y flexibles de Nmap. Permite a los usuarios escribir y compartir guiones simples para automatizar una amplia diversidad de tareas relacionadas a las redes. Estos guiones son ejecutados en paralelo con la velocidad y eficiencia esperada de Nmap. Los usuarios pueden confiar en la creciente diversidad de guiones distribuidos con Nmap, o escribir los propios en base a sus requerimientos.

Cuando se descubre una nueva vulnerabilidad, frecuentemente se requiere escanear la red rápidamente, con el propósito de identificar sistemas vulnerables. Aunque Nmap no es un escáner completo de vulnerabilidades, NSE es lo suficientemente poderoso para gestionar verificaciones de seguridad. Muchos guiones para la detección de vulnerabilidades están disponibles, y se planea distribuir más conforme sean escritos.

Por ejemplo, si ya escaneamos un equipo y conocemos los puertos abiertos, podemos usar algo como:

```
# nmap -Pn -n -O -sV -p21,22,80,445 192.168.0.50 –script vulners
```

la opción "–script vulners" define la ejecución de todos los guiones NSE incluidos en esta categoría. La lista completa de Scripts disponibles puede ser consultada en: */usr/share/nmap/scripts/*.

Para conocer lo que hace un Script en particular podemos usar:

```
# nmap –script-help http-headers 192.168.0.50
```

Podemos pedir que se realice la búsqueda con más de un Script usando:

```
# nmap –script http-headers,vulners 192.168.0.50
```

o múltiples Scrips mediante:

```
# nmap –script "ssh-*" 192.168.0.50
```

además podemos pasarle parámetros, por ejemplo:

```
# nmap –script mysql-audit –script-args "mysql-audit.username\
='root', mysql-audit.password='password_here', \
mysql-audit.filename='nselib/data/mysql-cis.audit'"
```

o pasar el número de puerto, usando *-p*:

```
# nmap -p 3306 –script mysql-audit –script-args \
"mysql-audit.username='root', mysql-audit.password\
='password_here' , mysql-audit.filename=\
'nselib/data/mysql-cis.audit'"
```

Hasta aquí hemos llegado con cómo instalar y utilizar *nmap*, uno de los programas de escaneo de puertos más conocidos y utilizados.

Otros Programas para Escaneo de Puertos Podemos instalar el programa *iproute2* para realizar el escaneo de puertos, mediante:

```
# apt install iproute2
```

y escaneamos puertos usando:

```
# ss -tulpn
```

Podemos instalar el programa *net-tools* para realizar el escaneo de puertos, mediante:

```
# apt install net-tools
```

y escaneamos puertos usando:

```
# netstat -tilnp
```

Podemos instalar el programa *lsof* para realizar el escaneo de puertos, mediante:

```
# apt install lsof
```

y escaneamos puertos usando:

```
# lsof -nP -iTCP -sTCP:LISTEN
```

Otras opciones son: Angry IP Scanner, Sandmap, Unicornscan, Netcat, Zeus, Vault, entre otros.

13.8 Cortafuegos

Cortafuegos o *Firewall* en inglés, es una solución diseñada para proteger tu equipo de cómputo. Son tipos de protección que posiblemente ya estés utilizando sin darte cuenta, pero eso no quiere decir que debas despreocuparte y no conocer lo que hacen y lo que no hacen. Todos los sistemas operativos tienen cortafuegos, tanto en cuanto son simples normas encargadas de reconducir el tráfico exterior de información hacia nuestro sistema operativo.

Empezaremos explicando de forma sencilla y entendible qué es un cortafuegos y para qué sirve exactamente. Luego, pasaremos a explicar cómo funcionan diciéndote los dos tipos principales de cortafuegos, y terminaremos recordándote que no debes delegar 100% en ellos ni ninguna otra solución para proteger tu equipo de cómputo.

Qué es un Cortafuegos y Para qué Sirve El cortafuegos en el mundo de la informática es un sistema de seguridad para bloquear accesos no autorizados a un equipo de cómputo mientras sigue permitiendo la comunicación de tu equipo de cómputo con otros servicios autorizados. También se utilizan en redes de equipos de cómputo, especialmente en Intranets o redes locales. Se trata de una de las primeras medidas de seguridad que empezó a implementarse en los equipos de cómputo tras el nacimiento de internet.

Su origen se remonta a finales de la década de los 80, cuando internet daba sus primeros pasos y los primeros Hackers descubrieron que con esta nueva

red podían infiltrarse y hacer travesuras en los equipos de cómputo de otras personas, lo que llevó a una serie de importantes violaciones de seguridad y ataques de *Malware*. internet necesitaba ser más segura para extenderse, por lo que varios investigadores empezaron a desarrollar las primeras versiones de cortafuegos informáticos en 1988 como método para el filtrado de los paquetes digitales que le llegaban a un equipo de cómputo.

Con el tiempo fueron evolucionando para conseguir analizar mejor la información entrante y filtrar las posibles amenazas. La finalidad siempre ha sido la misma que siguen teniendo hoy, la de establecer unos criterios de seguridad, y filtrar todas las comunicaciones que entran o salen del equipo de cómputo para interceptar las que no cumplan con ellos y dejar pasar al resto. Estos criterios van variando y evolucionando con el tiempo para mantenerse actualizados frente a unos ataques también en constante evolución. Es importante que sepas que los cortafuegos no eliminan el Malware que intenta entrar, sólo trata de bloquear su acceso.

Un equipo dedicado de cortafuegos es una máquina segura y confiable que se asienta entre una red privada y una red pública. La máquina cortafuegos se configura con un conjunto de reglas que determinan a qué tráfico de red se le permitirá pasar y cuál será bloqueado o rechazado. En algunas organizaciones grandes, puede que encuentre un cortafuegos localizado dentro de la red corporativa para separar áreas sensibles de la organización de otros empleados. Algunos casos de criminalidad informática acontecen dentro de la misma organización, no sólo provienen de fuera.

Se pueden construir cortafuegos en una variedad de maneras. La configuración más sofisticada involucra un número de máquinas separadas y se conoce como red perimetral. Dos máquinas, denominadas estranguladoras actúan como "filtros" para permitir pasar sólo ciertos tipos de tráfico de red, y entre estos estranguladores residen servidores de red como una pasarela de correo o un servidor intermediario de '*World Wide Web*'. Esta configuración puede resultar muy segura y permite de forma fácil un amplio rango de control sobre quién puede conectarse tanto desde dentro hacia fuera cómo desde fuera hacia dentro. Este tipo de configuración debería ser el utilizado por las grandes organizaciones.

Sin embargo, típicamente los cortafuegos son máquinas únicas que sirven todas estas funciones. Esto es algo menos seguro, porque si hay alguna debilidad en la propia máquina del cortafuegos que le permita a alguien conseguir el acceso al mismo cortafuegos, la seguridad de toda la red habrá sido comprometida. Sin embargo, estos tipos de cortafuegos son más baratos

y fáciles de mantener que la configuración más sofisticada descrita arriba.

Cómo Funcionan los Cortafuegos Los cortafuegos pueden ser de dos tipos, pueden ser Software, Hardware o una combinación de ambos. Esto quiere decir que pueden ser aplicaciones que instales en tu equipo de cómputo o dispositivos que se conecten a él para controlar el tráfico.

Los cortafuegos físicos pueden ser productos independientes o venir directamente integrados en un *Router*. Los independientes se suelen situar entre el punto de acceso a internet y el *Switch* que se encarga de distribuir la conexión entre los equipo de cómputo en una misma red. El hecho de que vaya antes de la distribución de la red entre los equipos significa que todos los que haya en una red interna quedan protegidos. Son buenos para muchos ataques exteriores, sobre todo para las redes internas e Intranets. Esto les convierte en buenas herramientas para empresas y grandes redes. Pero no son tan seguros con muchos tipos de ataque que vengan a través de otra aplicación, como los troyanos o las amenazas que recibes a través de correos electrónicos fraudulentos.

Los cortafuegos más populares en los usuarios, son los cortafuegos en forma de Software, que son aplicaciones que pueden instalarse en los equipos de cómputo[283]. Su desventaja es que sólo protegen de manera individual a cada equipo de cómputo que los tiene instalados.

Ahora vamos a hablar de sus funciones. Como hemos explicado, los cortafuegos se sitúan entre la red local e internet, y su misión es protegerte bloqueando el tráfico no solicitado o que considere peligroso. Pero puede hacer otras cosas, como aprovechar que analiza el tráfico que entra o sale para configurar filtros para diferentes tipos de tráfico con los que decidir qué hacer con él. En estas configuraciones se pueden hacer muchas cosas, como por ejemplo permitir únicamente las conexiones a servidores de direcciones *IP* concretas, descartando el resto por seguridad. Esto evidentemente a nivel doméstico no es muy efectivo, te impide navegar con facilidad, pero en ámbitos empresariales o más cerrados puede servir.

Al poder analizar el tráfico saliente, también pueden llegar a detectar si hay algún Malware comunicándose con la red, monitorizando el uso de redes empresariales, o filtrando el tráfico. Además, también puede configurarse, por ejemplo, para que sólo el navegador de los equipo de cómputo de una empresa

[283] En Windows, además de interceptar los intentos de acceso desde el exterior también suelen incluir protecciones adicionales contra los troyanos y virus de correo más comunes.

pueda conectarse a internet, bloqueando el acceso del resto de aplicaciones por seguridad.

El núcleo de GNU/Linux proporciona un rango de características internas que le permiten funcionar bastante bien como un cortafuegos de *IP*. La implementación de red incluye código para realizar filtros a nivel de *IP* en numerosas formas, y proporciona un mecanismo para configurar con precisión qué tipos de reglas le gustaría imponer. El cortafuegos en GNU/Linux es suficientemente flexible como para convertirle en algo muy útil en cualquiera de las configuraciones. El Software de cortafuegos de Linux proporciona otras dos características muy útiles que se discutirán en otras secciones: auditoría de *IP* y enmascaramiento de *IP*.

Por otro lado, recordemos los días en los que sólo las grandes compañías se podían permitir disponer de un cierto número de máquinas conectadas por una red local. Frente a aquello, hoy los precios de la tecnología de red han bajado y bajado hasta producir dos consecuencias: La primera, que las redes locales sean algo común, presentes incluso en entornos domésticos. Es seguro que tu tendrás en tu casa dos o más computadoras conectadas por algún tipo de Ethernet. La segunda, que los recursos de red, y de forma especial las direcciones IP, hayan llegado a ser algo escasos y, aunque no están lejanos los tiempos en que eran gratuitos, sean ahora objeto de compraventa.

La mayor parte de la gente que disponga de una *LAN* deseará también disfrutar de una conexión a internet que todas las máquinas de su red puedan utilizar al mismo tiempo. Las reglas del encaminamiento IP son muy estrictas respecto a la forma de manejar esta situación. Las soluciones tradicionales a este problema hubieran pasado por solicitar un conjunto de direcciones *IP*, probablemente un rango de clase C (*192.0.0.0 - 223.255.255.255*), dar a cada máquina de la *LAN* una dirección del rango asignado, y utilizar un enrutador para conectar la *LAN* a internet.

En el actual escenario de una internet mercantilizada, esa solución saldría bastante cara. En primer lugar habría que pagar por el rango de direcciones asignado, en segundo lugar habría que pagar con toda probabilidad al Proveedor de Servicios de internet (*ISP*) por el privilegio de disponer de una ruta hacia la red local en sus máquinas, de tal forma que el resto de internet supiera cómo llegar a ellas. Esto puede sonar posible para algunas empresas, pero en una instalación doméstica los costes no estarían justificados.

Afortunadamente GNU/Linux proporciona una solución al problema, solución que utiliza un componente de un grupo de funcionalidades avanzadas de red llamadas en conjunto Traducción de Direcciones de Red (*NAT*). *NAT* es

un conjunto de procedimientos para modificar las direcciones *IP* contenidas en las cabeceras de los datagramas *IP* mientras éstos viajan (al vuelo). Puede sonar extraño, pero mostraremos que se trata de la solución ideal al problema –real para muchos– que acabamos de plantear. '*IP masquerade*' es el nombre que recibe un tipo de traducción de direcciones de red que permite que todas las máquinas de una red privada utilicen internet contando con una única conexión (y una única dirección IP).

El enmascaramiento *IP* (en inglés «*IP masquerading*») permite utilizar un rango de direcciones privadas (reservadas) en la red local y que el encaminador GNU/Linux se encargue de hacer al vuelo ciertas traducciones de direcciones IP y puertos. Cuando le llega un datagrama IP de alguna máquina de la red local, se fija en el protocolo de nivel superior encapsulado en el mismo («*UDP*», «*TCP*», «*ICMP*», etc...) y modifica el datagrama para que parezca que fue generado por el propio encaminador (y recuerda qué ha sido modificado). A continuación saca el datagrama a internet donde aparece generado por la única dirección IP pública del encaminador. Cuando la máquina destino recibe el datagrama cree que se ha originado en la máquina GNU/Linux, y responde a su dirección de internet. Cuando el encaminador GNU/Linux recibe un datagrama en su interfaz de red conectada a internet, busca en su tabla de conexiones enmascaradas en curso para ver si el datagrama pertenece a alguna máquina de la *LAN* y, si es así, deshace la traducción que hizo en el primer datagrama y reenvía este datagrama de respuesta a la máquina local.

Tenemos una pequeña red ethernet en la que utilizamos uno de los rangos de direcciones reservadas. La red dispone de un encaminador con enmascaramiento, una máquina GNU/Linux, por supuesto, que proporciona acceso a internet. Una de las máquinas de la red (192.168.1.3) desea establecer una conexión con el Host remoto 209.1.106.178 en el puerto 8888. El equipo encamina su datagrama por el encaminador con enmascaramiento, que identifica la petición de conexión como requiriente de los servicios de enmascaramiento. El encaminador entonces acepta el datagrama y reserva un número de puerto (1035) para este menester, sustituye la dirección IP y número de puerto de la máquina origen del datagrama por los suyos propios, y transmite el datagrama al Host destino. El Host destino cree que ha recibido una petición de conexión de la máquina GNU/Linux enmascaradora, y genera un datagrama de respuesta. La máquina enmascaradora, al recibir ese datagrama, halla la asociación en su tabla de enmascaramiento y deshace la sustitución que llevó a cabo en el primer datagrama. Entonces transmite

el datagrama de respuesta a la máquina origen.

La máquina local cree que se está comunicando directamente con el Host remoto. El Host remoto no sabe nada de la existencia de la máquina local y cree que ha establecido una conexión con la máquina GNU/Linux enmascaradora. La máquina GNU/Linux enmascaradora sabe que las otras dos máquinas están hablando entre sí y en qué puertos, y realiza las traducciones de direcciones y puertos necesarias para que la comunicación tenga lugar.

Efectos Colaterales y Beneficios Accesorios la funcionalidad de enmascaramiento *IP* viene acompañada de su propio conjunto de efectos laterales, algunos son útiles y algunos pueden acabar siendo un problema.

Ninguna de las máquinas en la red detrás del encaminador enmascarador son jamás vistas directamente desde internet. Consecuentemente, solamente se necesita una dirección *IP* válida y rutable para permitir que todas las máquinas establezcan conexiones hacia internet. Esto tiene un lado no tan bueno: ninguna de esas máquinas es visible desde internet, y por lo tanto no se puede conectar directamente a ellas desde internet. La única máquina visible en una red enmascarada es el propio encaminador enmascarador. Se trata de algo importante cuando se piensa en servicios como el correo o el *FTP*. Resulta de utilidad decidir qué servicios deberían ser provistos por la máquina enmascaradora y para cuáles debería actuar como *Proxy* o tratar de algún otro modo especial.

Segundo, dado que ninguna de las máquinas enmascaradas son visibles, se encuentran relativamente protegidas de ataques del exterior. Eso puede simplificar (o eliminar) la necesidad de puesta a punto de funcionalidades de cortafuegos en la máquina enmascaradora. No se debe confiar demasiado en ésto, puesto que la red local estará únicamente tan segura como lo esté la máquina enmascaradora. Así, si la seguridad es un punto importante, se debería utilizar un cortafuegos para protegerla.

Tercero, el enmascaramiento *IP* tendrá cierto impacto negativo en el rendimiento de su red. En un escenario típico ese impacto negativo será probablemente insignificante. Si se tiene un gran número de sesiones enmascaradas activas puede ocurrir que se perciba cierta sobrecarga en la máquina enmascaradora que afecte negativamente al rendimiento de la red. El enmascaramiento *IP* implica un incremento considerable en el proceso que requiere cada datagrama comparado con el normalmente exigido.

Cuando utilizas un sistema Linux para conectar tu red local a internet,

tienes la posibilidad de permitir o no cierto tipo de tráfico. Las cabeceras de los paquetes *IP* contienen información sobre el destino (de forma que se puede prevenir el acceso a ciertos sitios de internet), el origen (se pueden evitar conexiones desde sitios concretos de internet). Otra información que se obtiene de las cabeceras es el protocolo utilizado (*ICMP, UDP, TCP*) y el puerto. Normalmente los protocolos de alto nivel utilizan para sus conexiones puertos determinados (también llamados *Well Known Sockets*). De esa forma, la mayor parte de las peticiones de documentos html se harán a destinos de internet por el puerto 80, el envío de correo se hará por el puerto 25, o las conexiones vía ssh se harán usando el puerto 22. Para más información ver */etc/services*.

Mediante el proyecto Nftables proporciona filtrado de paquetes y clasificación de paquetes en Linux. Es la evolución de *iptables*, y, de hecho, las reemplaza (no se puede mezclar *nftables* y *iptables*). *Nftables* es capaz de reemplazar en el mismo Framework a *iptables, ip6tables, arptables y ebtables*, y todo ello bajo el mismo espacio de usuario (nft) y compatibilidad hacia atrás (con sintaxis *iptables*). *Nftables* es el Framework por defecto en Debian 11 GNU/Linux.

Principales Características de Nftables *Nftables* usa una sintaxis más compacta e intuitiva que fue inspirada por la herramienta tcpdump. *Nftables* proporciona una capa de compatibilidad con *iptables*, usando su misma sintaxis sobre la infraestructura de nftables (lo que se utiliza en Debian 11 si no se instala *nft*). En *nftables*, las tablas y cadenas son totalmente configurables, no hay tablas predefinidas que siempre deben estar, aunque no las usemos (como sí ocurre con *iptables*). Los nombres también pueden ser arbitrarios.

Otras características importantes de *nftables*, es que los "match" -m y los "target" -j desaparecen, *nftables* tiene expresiones. *Nftables* permite hacer varias acciones en una sola regla (varios targets), algo que con iptables no era posible fácilmente. Por defecto no tenemos contadores integrados en las reglas y cadenas, ahora son opcionales y pueden habilitarse si queremos. Por último, nftables proporciona una administración más sencilla para conjuntos de reglas *IPv4* y *IPv6*, ahora no tendremos que «adaptar» las reglas de *iptables* a *ip6tables* como ocurría anteriormente.

Puesta en Marcha de *nftables* en Debian 11 Simplemente debemos instalar el paquete, todo lo demás ya está instalado (y se usa, aunque se instale *iptables*).

```
# apt install nftables
```

y de ser necesario habilitar el servicio mediante:

```
# systemctl enable nftables.service
```

e inicializar este, usando:

```
# systemctl start nftables.service
```

No es necesario reiniciar el equipo de cómputo o el servidor, no se toca nada del Kernel, solamente el paquete de administración del cortafuegos, que ahora pasa de estar en iptables, a estar con la sintaxis de *nftables*. Un detalle muy importante es que nftables hace una distinción entre las reglas temporales realizadas en la línea de órdenes, y aquellas otras permanentes cargadas o guardadas en un archivo.

El archivo predeterminado es */etc/nftables.conf*, que ya contiene una tabla simple de cortafuegos para *ipv4/ipv6* llamada "inet filter".

La utilidad de nftables en el espacio de usuario, *nft*, realiza la mayor parte de la evaluación del conjunto de reglas antes de pasarlas al Kernel del sistema operativo.

Si queremos consultar las reglas que tenemos dadas de alta en el cortafuegos, tendremos que poner la siguiente orden:

```
# nft list ruleset
```

En el caso de nftables, las reglas se almacenan en cadenas, que a su vez se almacenan en tablas. Para que los cambios permanezcan, deberemos guardar las reglas directamente en el archivo */etc/nftables.conf*.

Iniciemos por un cortafuegos básico, que nos permite salir en todos los puertos que tengamos servicios ejecutandose, pero desde el exterior no seamos visibles (pero podemos descomentar la línea que nos permite abrir puertos (en este ejemplo 22, 80, 443) para poner nuestro equipo como un servidor:

```
#!/usr/sbin/nft -f
flush ruleset
table inet filter {
chain input {
type filter hook input priority 0;
# aceptar cualquier tráfico de localHost
iif lo accept
# aceptar tráfico originado por nosotros
ct state established,related accept
        # aceptar ICMP y IGMP
        ip6 nexthdr icmpv6 icmpv6 type { destination-unreachable,
packet-too-big, time-exceeded, parameter-problem, mld-listener-
query, mld-listener-report, mld-listener-reduction, nd-router-solicit,
nd-router-advert, nd-neighbor-solicit, nd-neighbor-advert, ind-neighbor-
solicit, ind-neighbor-advert, mld2-listener-report } accept
        ip protocol icmp icmp type { destination-unreachable,
router-solicitation, router-advertisement, time-exceeded, parameter-
problem } accept
        ip protocol igmp accept
# descomentar la siguiente línea para aceptar servicios locales
comunes
#tcp dport { 22, 80, 443 } ct state new accept
# cuenta y descarta cualquier otro tráfico
counter drop
}
}
```

A continuación, como hemos incorporado nuevas reglas manualmente, debemos reiniciar nftables para aplicar los cambios, mediante:

```
# systemctl restart nftables.service
```

y checar su estatus mediante:

```
# systemctl status nftables.service
```

El comando nftables tiene una sintaxis rica que escapa del alcance de este texto, pero se puede obtener toda la información de las siguientes referencias:

https://www.netfilter.org/projects/nftables/manpage.html

https://wiki.nftables.org/wiki-nftables/index.php/Quick_reference-nftables_in_10_minutes

https://manpages.debian.org/testing/nftables/nft.8.en.html

https://home.regit.org/netfilter-en/nftables-quick-howto/

https://debian-handbook.info/browse/es-ES/stable/sect.Firewall-packet-filtering.html

Otras opciones de cortafuegos sobre línea de comando son: netfilter, iptables, ufw, apf y shorewall. Y otros con interfaz gráfica son: Gufw, Douane, OpenSnitch entre otros.

13.9 Acceso Remoto Mediante SSH

SSH o Secure Shell[284], es un protocolo de administración remota que permite a los usuarios controlar y modificar equipos de cómputo o servidores de forma remota, a través de internet mediante un mecanismo de autenticación.

Proporciona un mecanismo para autenticar un usuario remoto, transferir entradas desde el cliente al servidor y retransmitir la salida de vuelta al cliente. El servicio se creó como un reemplazo seguro para el Telnet sin cifrar y utiliza técnicas criptográficas para garantizar que todas las comunicaciones hacia y desde el servidor remoto sucedan de manera cifrada.

Cualquier usuario de Linux, MacOS o Windows[285] puede usar SSH para conectarse a un servidor remoto. Puedes ejecutar comandos Shell de la misma manera que lo harías si estuvieras operando físicamente el equipo remoto.

¿Cómo funciona SSH? si usas Linux o Mac, entonces usar el protocolo SSH es muy fácil. Si utilizas Windows, deberás utilizar un cliente SSH para abrir conexiones SSH. El cliente SSH más popular es PuTTY.

Para usuarios de Mac OS y Linux, debemos abrir el programa de terminal, el comando SSH consta de 3 partes distintas:

[284] SSH (o Secure SHell) es un protocolo que facilita las comunicaciones seguras entre dos sistemas usando una arquitectura cliente/servidor y que permite a los usuarios conectarse a un host remotamente. A diferencia de otros protocolos de comunicación remota tales como FTP o Telnet, SSH encripta la sesión de conexión, haciendo imposible que alguien pueda obtener contraseñas no encriptadas.

[285] Los usuarios de Windows pueden aprovechar los clientes SSH como Putty.

```
$ ssh {user}@{Host}[:<puerto>][comando]
```

El comando *ssh* o *mosh*[286] le indica a tu sistema que desea abrir una conexión de Shell Segura y cifrada.

- {user} representa la cuenta a la que deseas acceder. Por ejemplo, puede que quieras acceder al usuario root, que es básicamente para el administrador del sistema con derechos completos para modificar cualquier cosa en el sistema.
- {Host} hace referencia al equipo al que quieres acceder. Esto puede ser una dirección IP (por ejemplo, 244.235.23.19) o un nombre de dominio (por ejemplo, www.xyzdomain.com).
- puerto: Si el puerto de escucha de la máquina remota no es el habitual (el puerto 22) tendremos que especificarlo.

Al pulsar enter, se te pedirá que escribas la contraseña de la cuenta solicitada. Al escribirla, nada aparecerá en la pantalla, pero tu contraseña, de hecho, se está transmitiendo. Una vez que hayas terminado de escribir, pulsa enter una vez más. Si tu contraseña es correcta, verás una ventana de terminal remota. Ejemplos:

```
$ ssh google.com
$ ssh 192.168.1.2
$ ssh user@192.168.1.2
$ ssh user@google.com
$ ssh user@192.168.1.2:2222
$ ssh user@equipo "ps -ef"
$ ssh user@equipo "uptime; df -h"
$ ssh user@equipo "uptime && df -h"
```

[286] MOSH (Mobile Shell) como medio de conexión (no corta la comunicación por inactividad como SSH), lo podemos instalar usando:

```
# apt install mosh
```

y su uso es similar al de SSH:

```
$ mosh usuario@192.168.13.230
$ mosh -p 70 usuario@192.168.13.230
```

```
$ ssh user@equipo "sudo apt update && sudo apt upgrade"
$ ssh user@equipo 'bash -s' < script.sh
$ ssh user@equipo "nohup /home/antonio/respalda.sh &"
```

Aprendiendo las Diferentes Técnicas de Cifrado La ventaja significativa ofrecida por el protocolo SSH sobre sus predecesores es el uso del cifrado para asegurar la transferencia segura de información entre el Host y el cliente. Host se refiere al servidor remoto al que estás intentando acceder, mientras que el cliente es el equipo que estás utilizando para acceder al Host. Hay tres tecnologías de cifrado diferentes utilizadas por SSH:

- Cifrado simétrico
- Cifrado asimétrico
- Hashing

Cifrado Simétrico es una forma de cifrado en la que se utiliza una clave secreta tanto para el cifrado como para el descifrado de un mensaje, tanto por el cliente como por el Host. Efectivamente, cualquiera que tenga la clave puede descifrar el mensaje que se transfiere.

El cifrado simétrico a menudo se llama clave compartida (shared key) o cifrado secreto compartido. Normalmente sólo hay una clave que se utiliza, o a veces un par de claves donde una clave se puede calcular fácilmente con la otra clave.

Las claves simétricas se utilizan para cifrar toda la comunicación durante una sesión SSH. Tanto el cliente como el servidor derivan la clave secreta utilizando un método acordado, y la clave resultante nunca se revela a terceros. El proceso de creación de una clave simétrica se lleva a cabo mediante un algoritmo de intercambio de claves.

Lo que hace que este algoritmo sea particularmente seguro es el hecho de que la clave nunca se transmite entre el cliente y el Host. En lugar de eso, los dos equipos comparten datos públicos y luego los manipulan para calcular de forma independiente la clave secreta. Incluso si otra máquina captura los datos públicamente compartidos, no será capaz de calcular la clave porque el algoritmo de intercambio de clave no se conoce.

Debe tenerse en cuenta, sin embargo, que el token secreto es específico para cada sesión SSH, y se genera antes de la autenticación del cliente.

Una vez generada la clave, todos los paquetes que se mueven entre las dos máquinas deben ser cifrados por la clave privada. Esto incluye la contraseña escrita en la consola por el usuario, por lo que las credenciales siempre están protegidas de los fisgones de paquetes de red.

Existen varios códigos cifrados simétricos, incluyendo, pero no limitado a, AES (Advanced Encryption Standard), CAST128, Blowfish, etc. Antes de establecer una conexión segura, el cliente y un Host deciden qué cifrado usar, publicando una lista de cifrados soportados por orden de preferencia. El cifrado preferido de entre los soportados por los clientes que está presente en la lista del Host se utiliza como el cifrado bidireccional.

Cifrado Asimétrico a diferencia del cifrado simétrico, el cifrado asimétrico utiliza dos claves separadas para el cifrado y el descifrado. Estas dos claves se conocen como la clave pública (public key) y la clave privada (private key). Juntas, estas claves forman el par de claves pública-privada (public-private key pair).

La clave pública, como sugiere el nombre, se distribuye abiertamente y se comparte con todas las partes. Si bien está estrechamente vinculado con la clave privada en términos de funcionalidad, la clave privada no se puede calcular matemáticamente desde la clave pública. La relación entre las dos claves es altamente compleja: un mensaje cifrado por la clave pública de una máquina, sólo puede ser descifrado por la misma clave privada de la máquina. Esta relación unidireccional significa que la clave pública no puede descifrar sus propios mensajes ni descifrar nada cifrado por la clave privada.

La clave privada debe permanecer privada, es decir, para que la conexión sea segura, ningún tercero debe conocerla. La fuerza de toda la conexión reside en el hecho de que la clave privada nunca se revela, ya que es el único componente capaz de descifrar mensajes que fueron cifrados usando su propia clave pública. Por lo tanto, cualquier parte con la capacidad de descifrar mensajes firmados públicamente debe poseer la clave privada correspondiente.

A diferencia de la percepción general, el cifrado asimétrico no se utiliza para cifrar toda la sesión SSH. En lugar de eso, sólo se utiliza durante el algoritmo de intercambio de claves de cifrado simétrico. Antes de iniciar una conexión segura, ambas partes generan pares de claves públicas-privadas temporales y comparten sus respectivas claves privadas para producir la clave secreta compartida.

Una vez que se ha establecido una comunicación simétrica segura, el servi-

dor utiliza la clave pública de los clientes para generar y desafiar y transmitirla al cliente para su autenticación. Si el cliente puede descifrar correctamente el mensaje, significa que contiene la clave privada necesaria para la conexión. Y entonces comienza la sesión SSH.

Hashing el Hashing unidireccional es otra forma de criptografía utilizada en Secure Shell Connections. Las funciones de hash unidireccionales difieren de las dos formas anteriores de cifrado en el sentido de que nunca están destinadas a ser descifradas. Generan un valor único de una longitud fija para cada entrada que no muestra una tendencia clara que pueda explotarse. Esto los hace prácticamente imposibles de revertir.

Es fácil generar un hash criptográfico de una entrada dada, pero imposible de generar la entrada del hash. Esto significa que si un cliente tiene la entrada correcta, pueden generar el hash criptográfico y comparar su valor para verificar si poseen la entrada correcta.

SSH utiliza hashes para verificar la autenticidad de los mensajes. Esto se hace usando HMACs, o códigos de autenticación de mensajes basados en hash. Esto asegura que el comando recibido no se altere de ninguna manera.

Mientras se selecciona el algoritmo de cifrado simétrico, también se selecciona un algoritmo de autenticación de mensajes adecuado. Esto funciona de manera similar a cómo se selecciona el cifrado, como se explica en la sección de cifrado simétrico.

Todo mensaje transmitido debe contener un MAC, que se calcula utilizando la clave simétrica, el número de secuencia de paquetes y el contenido del mensaje. Se envía fuera de los datos cifrados simétricamente como la sección final del paquete de comunicaciones.

¿Cómo Funciona el Protocolo SSH con estas Técnicas de Cifrado? la forma en que funciona SSH es mediante el uso de un modelo cliente-servidor para permitir la autenticación de dos sistemas remotos y el cifrado de los datos que pasa entre ellos.

SSH opera en el puerto TCP 22 de forma predeterminada (aunque esto se puede cambiar si es necesario). El Host (servidor) escucha en el puerto 22 (o cualquier otro puerto SSH asignado) para las conexiones entrantes. Organiza la conexión segura mediante la autenticación del cliente y la apertura del entorno de Shell correcto si la verificación tiene éxito.

El cliente debe iniciar la conexión SSH iniciando el protocolo TCP con el servidor, asegurando una conexión simétrica segura, verificando si la identidad mostrada por el servidor coincide con los registros anteriores (normalmente grabados en un archivo de almacén de claves RSA) y presenta las credenciales de usuario necesarias para autenticar la conexión.

Hay dos etapas para establecer una conexión: primero ambos sistemas deben acordar estándares de cifrado para proteger futuras comunicaciones, y segundo, el usuario debe autenticarse. Si las credenciales coinciden, se concede acceso al usuario.

Negociación de Cifrado de Sesión cuando un cliente intenta conectarse al servidor a través de TCP, el servidor presenta los protocolos de cifrado y las versiones respectivas que soporta. Si el cliente tiene un par similar de protocolo y versión, se alcanza un acuerdo y se inicia la conexión con el protocolo aceptado. El servidor también utiliza una clave pública asimétrica que el cliente puede utilizar para verificar la autenticidad del Host.

Una vez que esto se establece, las dos partes usan lo que se conoce como Algoritmo de Intercambio de Claves Diffie-Hellman para crear una clave simétrica. Este algoritmo permite que tanto el cliente como el servidor lleguen a una clave de cifrado compartida que se utilizará en adelante para cifrar toda la sesión de comunicación.

Aquí es cómo el algoritmo trabaja en un nivel muy básico:

1. Tanto el cliente como el servidor coinciden en un número primo muy grande, que por supuesto no tiene ningún factor en común. Este valor de número primo también se conoce como el valor semilla (seed value).

2. Luego, las dos partes acuerdan un mecanismo de cifrado común para generar otro conjunto de valores manipulando los valores semilla de una manera algorítmica específica. Estos mecanismos, también conocidos como generadores de cifrado, realizan grandes operaciones sobre la semilla. Un ejemplo de dicho generador es AES (Advanced Encryption Standard).

3. Ambas partes generan independientemente otro número primo. Esto se utiliza como una clave privada secreta para la interacción.

4. Esta clave privada recién generada, con el número compartido

y el algoritmo de cifrado (por ejemplo, AES), se utiliza para calcular una clave pública que se distribuye a la otra computadora.

5. A continuación, las partes utilizan su clave privada personal, la clave pública compartida de la otra máquina y el número primo original para crear una clave compartida final. Esta clave se calcula de forma independiente por ambos equipos, pero creará la misma clave de cifrado en ambos lados.

6. Ahora que ambas partes tienen una clave compartida, pueden cifrar simétricamente toda la sesión SSH. La misma clave se puede utilizar para cifrar y descifrar mensajes (leer: sección sobre cifrado simétrico).

Ahora que se ha establecido la sesión cifrada segura simétricamente, el usuario debe ser autenticado.

Instalación de OpenSSH y Puesta en Marcha OpenSSH es el programa servidor/cliente SSH más utilizado por los Routers, Switches, servidores y un largo etcétera de dispositivos. Este programa es completamente gratuito y de código abierto. La instalación de este servidor SSH (si es que no lo tienes ya instalado por defecto) es muy sencilla, simplemente debemos poner en un terminal la siguiente orden:

```
# apt install openssh-server
```

una vez instalado, estamos listos para que nuestro equipo -ahora ya es un servidor- reciba usuarios remotos.

Hay que tener en cuenta ciertos directorios y órdenes que nos permiten iniciar, parar y reiniciar el servicio SSH.

Para editar la configuración del servidor SSH debemos hacer en consola:

```
# nano /etc/ssh/sshd_config
```

otro directorio que tenemos que tener muy en cuenta es el de Hosts conocidos, ya que aquí también es donde configuraremos las claves criptográficas *RSA/DSA*. El directorio donde se encuentran los Hosts conocidos y las claves públicas es el siguiente:

```
~/.ssh/
```

este directorio por defecto está oculto (.ssh) y hay un directorio por cada usuario que haya en el equipo de cómputo y que se conecte a un servidor remoto.

Si necesitamos arrancar manualmente el servidor, usamos:

```
# /etc/init.d/sshd start
```

o

```
# systemctl start sshd
```

para parar el servidor manualmente, usamos:

```
# /etc/init.d/sshd stop
```

o

```
# systemctl stop sshd
```

y si cambiamos la configuración o necesitamos reiniciar el servidor, usamos:

```
# /etc/init.d/sshd restart
```

o

```
# systemctl reload sshd
```

Una vez que hemos instalado el servidor SSH y sabemos dónde están los archivos de configuración del servidor y el directorio donde se almacenan las claves públicas, notemos que el servicio del servidor SSH se puede configurar para hacer un poco más eficiente y seguro. Para ello es necesario hacer cambios a la configuración por omisión de OpenSSH mediante la edición del archivo *sshd_config*. Algunos de estos cambios son:

Cambiar el Puerto por Defecto del Servidor SSH Por defecto los servidores SSH utilizan el puerto 22 para las conexiones. Es recomendable cambiar este número de puerto, para evitar que bots o cibercriminales puedan intentar iniciar sesión, aunque por sí solo esto no proporciona seguridad, sí podremos pasar desapercibidos a los escaneos masivos desde internet. Si por ejemplo queremos usar el puerto 22445 debemos poner en el fichero de configuración lo siguiente:

```
Port 22445
```

otra opción es limitar las IPs que pueden tener acceso a nuestro equipo, usando por ejemplo:

```
ListenAddress 192.168.13.230
ListenAddress 132.248.182.159
```

Bloquear el Acceso root en las Conexiones Remotas Por defecto, cualquier usuario en el sistema operativo que tenga permisos de Shell, podrá iniciar sesión en el servidor. Además, debemos tener en cuenta que si tenemos activado el usuario root, también podrá conectarse al servidor de forma local o remota, evitando al atacante tener que «adivinar» el nombre de usuario. Por defecto, los bots siempre intentan atacar el puerto 22 y al usuario «root».

OpenSSH también nos permitirá deshabilitar el login del usuario root para dotar al sistema de mayor seguridad:

```
PermitRootLogin no
```

De esta manera las conexiones root quedarán bloqueadas evitando que usuarios no autorizados puedan realizar ataques de fuerza bruta contra nuestro servidor SSH para adivinar las credenciales del usuario Root. También tenemos otras opciones en este apartado, como por ejemplo «PermitRootLogin without-password» donde se permite autenticación pero no con usuario y contraseña, sino con claves criptográficas RSA.

Configuraciones de Seguridad Adicionales Existen otras configuraciones recomendadas para evitar las conexiones no deseadas a nuestro servidor SSH. Estas conexiones son:

- PermitEmptyPasswords: Indica si se permiten o no claves de usuario vacías, por seguridad el valor debe ser: No
- LoginGraceTime: Estableceremos el tiempo necesario para introducir la contraseña, evitando que el atacante tenga que «pensar mucho».
- ClientAliveInterval: Indica el tiempo en segundos que la conexión estará activa sin ninguna actividad.
- ClientAliveCountMax: Controla cuántas veces se envía al mensaje vivo antes de desconectarnos
- MaxAuthTries: Número de intentos permitidos al introducir la contraseña antes de desconectarnos.
- MaxStartups: Número de accesos simultáneos desde una IP, para evitar que se pueda utilizar la fuerza bruta con varias sesiones a la vez.
- Protocol: Indicar que solo se permitirá el uso del protocolo 2 y no el 1.
- AllowUsers: Es crear una lista blanca de usuarios. Este parámetro nos permite configurar los usuarios que podrán conectarse. Una medida muy restrictiva pero a la vez muy segura ya que bloqueará todas las conexiones de los usuarios que no estén en el listado. Los usuarios que tengamos aquí podrán conectarse, y el resto no.
- DenyUsers: Parecido al anterior, pero ahora creamos una lista negra. Los usuarios que tengamos aquí no podrán conectarse, y el resto sí.
- AllowGroups: Nos permite controlar a qué grupos daremos acceso al sistema, es complemento de la opción anterior.
- DenyGroups: Parecido al anterior, pero ahora creamos una lista negra. Los grupos que tengamos aquí no podrán conectarse, y el resto sí.
- X11Forearding: indica si daremos o no acceso al despliegue de gráficos a través del sistema X11.

Por ejemplo, un archivo de configuración de sshd_config sería el siguiente:

```
Port 22445
Protocol 2
LoginGraceTime 20
ClientAliveInterval 60
ClientAliveCountMax 4
MaxAutTries 3
MaxStartups 3
MaxSessions 3
X11Forwarding no
AllowUsers antonio
DenyUsers karla
PermitEmptyPasswords no
PermitRootLogin no
```

para conocer todas las opciones habilitadas en *sshd*, usamos:

```
$ sshd -T
```

Permitir Autenticación Basada en Llaves Desde el equipo de cómputo donde nos queramos conectar al servidor con claves criptográficas, debemos crear dichas claves usando *ssh-keygen* y pasárselas al servidor. Tenemos varias opciones a saber:

- RSA es un antiguo algoritmo basado en la dificultad de factorizar números primos grandes. Se recomienda un tamaño de clave de al menos 2048 bits, pero 4096 es mejor (este algórtimo está envejeciendo y se están logrando avances significativos en la factorización y es aconsejable elegir un algoritmo diferente), para crear una clave RSA tenemos que poner en el cliente SSH la siguiente orden:

  ```
  $ ssh-keygen -t rsa -b 4096
  ```

- DSA es un antiguo algoritmo de firma digital del gobierno de EE.UU., se basa en la dificultad de calcular logaritmos discretos, normalmente se usaria un tamaño de clave de 1024 (ya no se recomienda DSA en su forma original), para crear una clave DSA tenemos que poner en el cliente SSH la siguiente orden:

  ```
  $ ssh-keygen -t dsa -b 1024
  ```

- ECDSA es un nuevo algoritmo de firma digital estandarizado por el gobierno de EE.UU., que utiliza curvas elípticas (este es probablemente un buen algoritmo para las aplicaciones actuales, solo se admiten tres tamaños de clave: 256, 384 y 521bits, recomendamos usar 521 bits), para crear una clave ECDSA tenemos que poner en el cliente SSH la siguiente orden:

  ```
  $ ssh-keygen -t ecdsa -b 521
  ```

- ED25519 es un nuevo algoritmo agregado en OpenSSH, el soporte aún no es universal pero ya está muy difundido para su uso en aplicaciones de propósito general, para crear una clave ed25519 tenemos que poner en el cliente SSH la siguiente orden:

  ```
  $ ssh-keygen -t ed25519
  ```

En todos los casos, el asistente de generación de estas claves, nos pondrá si queremos guardarlas en ~/.ssh/id_rsa, le decimos que sí. Posteriormente nos permitirá poner a la clave privada una contraseña de paso (no usar contraseña es adecuado para Scripts), de esta forma, si perdemos la llave privada no pasará nada porque no podrán conectarse, debido a que es necesario siempre introducir una contraseña de paso para poder realizar la conexión correctamente.

Una vez que hayamos creado la clave pública y privada en nuestro equipo, debemos enviar la clave pública al servidor SSH donde nos queramos conectar. Para copiarlo al servidor usamos:

```
$ ssh-copy-id usuario@servidor
```

el usuario deberá confirmar que quiere añadir la identidad e introducir las credenciales de inicio de sesión para la cuenta que se quiere utilizar en ese servicio. Por este motivo es importante que en el servidor aún mantengamos la posibilidad de autenticarnos con usuario/clave. Notemos, por ejemplo para RSA que la llave pública se almacena en:

```
~/.ssh/id_rsa.pub
```

y la llave privada se almacena en:

~/.ssh/id_rsa

Automáticamente la clave pública se copiará a dicho servidor, y ya podremos habilitar la autenticación con solo clave pública y automáticamente nos habremos dado de alta. Este proceso agrega la llave pública al archivo del servidor:

~/.ssh/authorized_keys

Una vez completado el proceso de copiado de la llave, tendríamos que ser capaces de hacer inicio de sesión en este equipo sin introducir la contraseña:

```
$ ssh usuario@servidor
```

Si usamos contraseña de paso en la generación de la llave y para evitar estar capturando la contraseña en cada conexión, podemos cargar en memoria la llave una sola vez, usando:

```
$ ssh-add[287]
```

en caso de no permitirlo usar:

```
$ eval ‘ssh-agent -s‘
$ ssh-add
```

Para listar todas las llaves privadas en Cache usamos:

```
$ ssh-add -l
```

para ver la lista de todos los parámetros de clave pública de todas las identidades, usamos:

```
$ ssh-add -L
```

y para eliminar todas las claves privadas almacenadas en Cache, usamos:

```
$ ssh-add -D
```

[287] Se puede indicar un tiempo máximo de vida, por ejemplo 3 horas y 42 minutos:

```
$ ssh-add -t 3h42
```

o bien usando segundos, por ejemplo 1800 segundos:

```
$ ssh-add -t 1800
```

Uso de sshpass Esta utilidad está diseñada para ejecutar *SSH* usando el modo de autenticación de contraseña interactiva con el teclado, pero de una manera no interactiva. *SSH* utiliza acceso directo *TTY* para garantizar que la contraseña sea emitida por un usuario de teclado interactivo, *sshpass* ejecuta *SSH* en un *TTY* dedicado, engañando a *SSH* haciéndole creer que está obteniendo la contraseña de un usuario interactivo.

Para su instalación usamos:

```
# apt install sshpass
```

Para usar *sshpass*, debemos especificar el comando que deseamos ejecutar después de las opciones de sshpass. Normalmente, el comando es *ssh* con argumentos, pero también puede ser cualquier otro comando. Sin embargo, la solicitud de contraseña *SSH* está actualmente codificada en *sshpass*.

La sinopsis del comando *sshpass* se describe a continuación:

```
$ sshpass [-farchivo|-dnumero|-ppassword|-e] [opciones] comando
argumentos
```

Dónde:

-*ppassword* es la contraseña que se proporciona en la línea de comandos.

-*farchivo* es la contraseña en la primera línea del nombre del archivo.

-*dnumero* es el número es un descriptor de archivo heredado por *sshpass* del corredor. La contraseña se lee del descriptor de archivo abierto.

-*e* es la contraseña que se toma de la variable de entorno "SSH-PASS".

Ejemplos: Utilizaremos *sshpass* para iniciar sesión en un servidor remoto mediante *SSH*. Supongamos que la contraseña es: *MiContrasena2%*. A continuación, se muestran varias formas de utilizar las opciones de *sshpass*.

Usamos la opción -p (esta se considera la opción menos segura y no debe evitarse usar):

```
$ sshpass -p MiContrasena2% ssh usuario@maquina
```

La opción -p se ve así cuando se usa en un script de shell:

```
$ sshpass -p MiContrasena2% ssh -o StrictHostKeyChecking=no usuario@maquina
```

Utilizamos la opción -f (la contraseña debe ser la primera línea del nombre del archivo):

```
$ echo 'MiContrasena2%' > pas_archivo
$ chmod 0400 pass_archivo
$ sshpass -f pass_archivo ssh usuario@maquina
```

Aquí usamos la opción -f cuando se usa en un script de shell:

```
$ sshpass -f pass_archivo ssh -o StrictHostKeyChecking=no usuario@maquina
```

Utilizamos la opción -e (la contraseña debe ser la primera línea del nombre del archivo):

```
$ SSHPASS='MiContrasena2%' sshpass -e ssh -o StrictHostKeyChecking=no usuario@maquina
```

Utilizamos *sshpass* con *rsync*:

```
$ SSHPASS='MiContrasena2%' rsync –rsh="sshpass -e ssh -l usuario" /ruta/ maquina:/ruta/
```

lo anterior usa la opción -e, que pasa la contraseña a la variable de entorno SSHPASS, podemos usar el modificador -f así:

```
$ rsync –rsh="sshpass -f pass_archivo ssh -l usuario" /ruta/ maquina:/ruta/
```

Utilizamos *sshpass* con *scp*:

```
$ scp -r /ruta/ –rsh="sshpass -f pass_archivo ssh -l usuario" maquina:/ruta/
```

También podemos utilizar *sshpass* con un archivo cifrado con *GPG*. Cuando se usa el modificador -f, el archivo de referencia está en texto plano. Veamos cómo podemos cifrar un archivo con *GPG* y usarlo.

Primero, creamos un archivo de la siguiente manera:

$ echo '*MiContrasena2%*' > sshpasswd

Luego, ciframos el archivo usando el comando *gpg*:

$ gpg -c sshpasswd

Eliminamos el archivo que contiene el texto sin formato
$ rm sshpasswd
Finalmente, lo utilizamos de la siguiente manera:

$ gpg -d -q sshpassword.gpg > pass_archivo; sshpass -f pass_archivo ssh usuario@maquina

sshpass es una herramienta sencilla que puede ser de gran ayuda para los administradores de sistemas. Esto no anula, de ninguna manera, la forma más segura de autenticación *SSH*, que es la autenticación de clave pública. Sin embargo, sshpass también se puede agregar a la caja de herramientas del administrador de sistemas.

13.10 Registros del Sistema

Los registros del sistema o Logs son ficheros de texto que registran cronológicamente la totalidad de actividades e incidencias importantes que ocurren en el sistema operativo o red.

Debemos de ser conscientes que GNU/Linux registra absolutamente todo lo que pasa en el sistema operativo mediante los Logs. Este es uno de los motivos por los que GNU/Linux es un sistema operativo excelente.

¿Qué registran los Logs? principalmente los Logs se clasifican en cuatro categorías, que son:

- Logs de aplicaciones.
- Logs de eventos.

- Logs de servicios.
- Logs del sistema.

Algunos ejemplos de la información que contienen los Logs son:

- Los paquetes que se instalan y desinstalan en el sistema operativo.
- Información sobre los accesos remotos a nuestro equipo.
- Los intentos fallidos de autenticación de los usuarios al equipo.
- Registro de errores que se dan en los programas o servicios que usamos.
- Los accesos o salidas bloqueadas por nuestro cortafuego.
- Etc.

Como se puede ver, los Logs son una fuente básica para saber que está pasando en nuestro equipo. Por lo tanto, todo administrador de sistemas debería saber cómo consultar los Logs para de este modo:

- Detectar la causas de los problemas que puede tener un equipo o servidor.
- Registrar y detectar ataques informáticos por parte de un Hacker.
- Detectar comportamientos anómalos de programas o servicios que tengamos instalados en nuestro equipo.
- Ver el rendimiento de un ordenador o servidor.
- Averiguar que ocurrió en una determinada actualización de un equipo.
- Etc.

Una gran parte de los archivos Logs se almacenan en */var/log/* y sus subdirectorios. No obstante, es posible encontrar programas que guarden los Logs en otras ubicaciones del sistema operativo.

Algunos de los archivos Logs más importantes son:

- /var/log/message: registro de mensajes generales del sistema

- /var/log/auth.log: log de autenticación
- /var/log/kern.log: registro del Kernel
- /var/log/cron.log: registro de crond
- /var/log/maillog: registro del servidor de mails
- /var/log/qmail/ : registro de Qmail
- /var/log/httpd/: registro de errores y accesos a Apache
- /var/log/lighttpd: registro de errores y accesos a Lighttpd
- /var/log/boot.log : registro de inicio del sistema
- /var/log/mysqld.log: registro de la base de datos MySQL
- /var/log/secure: log de autenticación
- /var/log/utmp or /var/log/wtmp : registro de logins

El demonio *rsyslogd* es responsable de recolectar los mensajes de servicio que provienen de aplicaciones y el núcleo para luego distribuirlos en archivos de registros (usualmente almacenados en el directorio */var/log/*). Obedece a su archivo de configuración: */etc/rsyslog.conf*.

Cada mensaje de registro es asociado con un subsistema de aplicaciones (llamados «facility» en la documentación):

- auth y authpriv: para autenticación;
- cron: proviene servicios de programación de tareas, cron y atd;
- daemon: afecta un demonio sin clasificación especial (DNS, NTP, etc.);
- ftp: el servidor FTP;
- kern: mensaje que proviene del núcleo;
- lpr: proviene del subsistema de impresión;
- mail: proviene del subsistema de correo electrónico;
- syslog: mensajes del servidor syslogd en sí;

- user: mensajes de usuario (genéricos);
- local0 a local7: reservados para uso local.

Cada mensaje tiene asociado también un nivel de prioridad. Aquí está la lista en orden decreciente:

- emerg: «¡Ayuda!» Hay una emergencia y el sistema probablemente está inutilizado.
- alerta: apúrese, cualquier demora puede ser peligrosa, debe reaccionar inmediatamente;
- crit: las condiciones son críticas;
- err: error;
- warn: advertencia (error potencial);
- notice: las condiciones son normales pero el mensaje es importante;
- info: mensaje informativo;
- debug: mensaje de depuración.

¿Qué contienen los Logs de sistema?

/var/log/auth.log proporciona un registro de todas las actividades que implican un proceso de autenticación. Por ejemplo registra los usuarios logueados al sistema operativo. Registra el día, hora, usuario y órdenes que se han ejecutado con el comando sudo, los cronjobs que se han ejecutado, los intentos fallidos de autenticación, etc.

/var/log/debug en él se registran datos de los programas que están actuando en modo depuración. De esta forma los programadores pueden obtener información si sus programas están funcionando adecuadamente.

/var/log/syslog contiene la totalidad de logs capturados por *rsyslogd*. Por lo tanto en este fichero encontraremos multitud de Logs y será difícil de consultar y filtrar. Por este motivo, los Logs se distribuyen en otros ficheros siguiendo la configuración del fichero */etc/rsyslog.conf*.

/var/log/kern.log: proporciona información detallada de mensajes del Kernel. También puede ser útil para intentar detectar y solucionar problemas con la detección de Hardware.

/var/log/dmesg dentro del fichero encontraremos información relacionada con el Hardware de nuestro equipo. Por lo tanto podremos obtener información para concluir si nuestro Hardware funciona de forma adecuada.

/var/log/messages contiene mensajes informativos y no críticos de la actividad del sistema operativo. Acostumbra a contener los errores que se registran en el arranque del sistema que no estén relacionados con el Kernel. Por lo tanto, si no se inicia un servicio, como por ejemplo el servidor de sonido, podemos buscar información dentro de este archivo.

/var/log/faillog registra los intentos fallidos de autenticación de cada usuario. Dentro del archivo se almacena una lista de usuarios, los fallos totales de cada usuario, el número de fallo máximos que permitimos y la fecha y hora del último fallo. Si un usuario supera el número de fallos máximos establecidos se deshabilitará el usuario por el tiempo que nosotros fijemos.

/var/log/btmp almacena los intentos fallidos de logins en un equipo. Si alguien realizará un ataque de fuerza bruta a un servidor *ssh*, el fichero registraría la *IP* del atacante, el día y hora en que ha fallado el login, el nombre de usuario con que se ha intentado loguear, etc.

/var/log/lastlog ayuda a ver la fecha y la hora en que cada usuario se ha conectado por última vez.

/var/log/wtmp en todo momento contiene los usuarios que están en el sistema operativo.

/var/log/boot.log contiene información relacionada con el arranque del sistema. Podemos consultarlo para analizar si se levantan los servicios del sistema, si se levanta la red, si se montan las unidades de almacenamiento, para averiguar un problema que hace que nuestro equipo no inicie, etc.

/var/log/cron registra la totalidad de información de las tareas realizadas por *cron*. Si tienen problemas con la ejecución de tareas tienen que consultar este log para ver si el trabajo se ha ejecutado o da errores. Debian no dispone de este log, pero encontrarán la misma información en /var/log/syslog. En Debian pueden configurar el fichero de configuración /etc/rsyslog.conf para generar un log específico para *cron*.

/var/log/daemon.log registra la actividad de los demonios o programas que corren en segundo plano. Para ver si un demonio se levantó o está dando errores podemos consultar este Log. Dentro de daemon.log encontraremos información sobre el demonio que inicia el gestor de inicio, el demonio que inicia la base de datos de MySQL, etc.

/var/log/dpkg.log contiene información sobre la totalidad de paquetes instalados y desinstalados mediante el comando dpkg.

/var/log/apt/history.log detalle de los paquetes instalados, desinstalados o actualizados mediante el gestor de paquetes apt.

/var/log/apt/term.log contiene la totalidad de información mostrada en la terminal en el momento de instalar, actualizar o desinstalar un paquete con apt.

/var/log/mail.log información relacionada con el servidor de email que tengamos instalado en el equipo y registra la totalidad de sus acciones en mail.log.

/var/log/alternatives.log registra todas las operaciones relacionadas con el sistema de alternativas. Por lo tanto, todas las acciones que realicemos usando el comando update-alternatives se registrarán en este log. El sistema de alternativas permite definir nuestro editor de texto predeterminado, el entorno de escritorio predeterminado, la versión de java que queremos usar por defecto, etc.

/var/log/Xorg.0.log registra la totalidad de eventos relacionados con nuestra tarjeta gráfica desde que arrancamos el ordenador hasta que lo apagamos. Por lo tanto puede ayudar a detectar problemas con nuestra tarjeta gráfica.

/var/run/utmp Ver los usuarios que actualmente están logueados en un equipo.

¿Cómo podemos consultarlos?

El comando tail use el comando tail -f para ver las últimas líneas de un archivo y las actualizaciones del mismo. Por ejemplo, haciendo:

```
# tail -f /var/log/auth.log
```

nos muestra eventos de autenticación como sesiones nuevas y el uso de *sudo* en el sistema. Si un usuario inicia una nueva sesión, verá la actualización del archivo con la nueva información usando este comando. Finalice el comando tail pulsando las teclas Ctrl y c simultáneamente.

Usamos el comando *tail* para leer las últimas líneas del archivo, por ejemplo muestra las 10 últimas líneas del Log de errores:

```
# tail -n 10 error.log
```

y en este caso, se muestran las últimas 100 líneas una sola línea al mismo tiempo usando el comando *more*:

```
# tail -n 100 error.log | more
```

El comando less el comando *less* +F es casi igual a usar *tail* -f, pero proporciona acceso al archivo entero en vez de las últimas líneas y también muestra cualquier actualización en tiempo real. Por ejemplo:

```
# less +F /var/log/kern.log
```

el kern.log muestra mensajes del Kernel. Finalice el comando pulsando las teclas Ctrl y c simultáneamente, seguido por la tecla q.

El comando grep el comando *grep* permite buscar una cadena en uno o más archivos, para ello:

```
# grep -ir cadena /var/log/*
```

en donde el parámetro -i ignora distinciones como mayúsculas/minúsculas; -r busca de forma recursiva en todos los subdirectorios, cadena lo substituiremos por la cadena de texto a buscar, y /var/log/* es el directorio donde vamos a buscar esas coincidencias de texto, por ejemplos:

```
# grep -ir syslog /var/log/*
# grep -ir ssh /var/log/*
# grep -ir warning /var/log/*
# grep -ir critical /var/log/*
```

El comando grep comparte funcionalidad con egrep y fgrep. ¿Cuál es la diferencia entre grep, egrep y fgrep?, la e en egrep significa extendido (grep -E) y la f en fgrep significa fijo (grep -F). egrep permite el uso de expresiones regulares extendidas y fgrep no permite la expresión regular en absoluto.

Otras opciones existe la opción de gestionar ficheros comprimidos, gracias a los comandos *zgrep, zegrep, zless, zmore, zdiff, zcmp, zcat*. Como te puedes dar cuenta la función de cada uno de estos comandos será la misma que su homónimos sin «z» (*grep, egrep, less, more, diff, cmp, cat*) pero para ficheros comprimidos con *gzip* y extensión *.gz*. De forma análoga para archivos comprimidos usando *bz2*, están los comandos *bzgrep, bzegrep, bzless, bzmore, bzdiff, bzcmp, bzcat* y para archivos comprimidos usando *xz*, están los comandos *xzgrep, xzegrep, xzless, xzmore, xzdiff, xzcmp, xzcat*.

El Anillo Buffer del Kernel es una estructura de datos que registra los mensajes relacionados a la operación del Kernel Linux. Es como una especie de Log del Kernel, salvo que al ser un buffer, su tamaño es constante y funciona como una anillo (una vez que se llena se sobrescriben los mensajes viejos con los nuevos entrantes).

En el anillo buffer típicamente se registran mensajes relacionados al Hardware, memoria, CPU y dispositivos. Cabe destacar que no es una característica específica del Kernel Linux, sino que es común a todos los sistemas operativos de la familia Unix.

La herramienta *dmesg* permite examinar el contenido del anillo buffer del Kernel.

```
# dmesg | tail
```

Por defecto muestra todos los mensajes en el buffer, por ello típicamente resulta práctico filtrar la salida con grep para buscar mensajes específicos. Por ejemplo:

```
# dmesg | grep file
```

dmesg lee el anillo buffer del Kernel Linux desde el dispositivo */dev/kmsg*. Sin embargo es capaz de leer desde cualquier archivo (mediante la opción -F) o desde la interfaz del Kernel de syslog (-S). Esto permite leer mensajes del Kernel de encendidos previos:

```
# dmesg -F /var/log/dmesg.0 | tail
```

Tengamos en cuenta que un anillo del buffer se mantiene durante todo el tiempo de encendido. Por eso hay muchos archivos dmesg.*

Lógicamente éstos son los archivos de log generados por syslog y no funcionan en modo anillo, sino que todos los mensajes generados por el Kernel persisten.

El dispositivo */dev/kmesg* retorna datos crudos, en cambio syslog utiliza un formato en particular. Es posible leer directamente el dispositivo */dev/kmesg* con el siguiente comando:

```
# dd if=/dev/kmsg iflag=nonblock
```

Por ejemplo:

```
# dd if=/dev/kmsg iflag=nonblock | head
```

Sin embargo, por compatibilidad, *dmesg* retorna la información en el formato de *syslog*. En los sistemas donde el dispositivo */dev/kmesg* es legible (por ejemplo los Kernels Linux a partir de la versión 3.5.0), la opción -w permite mantener la salida abierta esperando nuevos mensajes. Similar a lo que ocurre cuando se lanza tail con la opción follow.

Opciones de dmesg más allá del uso básico que se le pueda dar, *dmesg* tiene varias opciones interesantes, especialmente para el formato de la salida. Por defecto *dmesg* muestra una estampilla de tiempo para cada entrada expresada en segundos relativos al momento de inicio del Kernel. La opción -H permite mostrar la fecha y hora, con deltas entre mensajes consecutivos, en lugar de la estampilla de tiempo:

```
# dmesg -H | tail
```

-T es similar pero muestra la fecha en formato humano para cada mensaje:

```
# dmesg -T | tail
```

No resulta práctica porque la unidad de tiempo es poco precisa.
Con -t directamente se suprimen las estampillas de tiempo:

```
# dmesg -t | tail
```

Puede ser de gran utilidad para simplificar el parseo de la salida con algún Script.

-d muestra los deltas (diferencias de tiempo entre dos mensajes consecutivos) además de la estampilla:

```
# dmesg -d | tail
```

-e resulta similar a -H:

```
# dmesg -e | tail
```

por último, es posible filtrar los mensajes por nivel utilizando –level:

```
# dmesg –level=err
```

Para listar los mensajes de error soportados, abrir la ayuda (no el manual) de dmesg ejecutando:

```
# dmesg –help
```

Los niveles de error soportados dependen del sistema, alguno de ellos son los siguientes:

Supported log levels (priorities):

- emerg - system is unusable
- alert - action must be taken immediately
- crit - critical conditions
- err - error conditions
- warn - warning conditions
- notice - normal but significant condition
- info - informational
- debug - debug-level messages

Existen diversos programas para consultar los Logs a través de una interfaz gráfica. Algunos de ellos son:

- gnome-logs: Programa del entorno gnome usado para visualizar los logs registrados por journald.
- KSystemLog: Utilidad elaborada por el equipo de KDE que nos permitirá consultar los logs del sistema operativo registrados por journald.
- gnome-system-log: Aplicación que al igual que las anteriores nos permitirá consultar los registros del sistema.
- Etc.

Como se Registran los Logs en Linux La totalidad de Logs de nuestro equipo los registran y gestionan los demonios *rsyslogd* y/o *journald*. Existen otros demonios para gestionar los logs en Linux. No obstante *rsyslogd* es el más habitual. Otros demonio que se pueden usar son *syslog-ng o syslog*.

A continuación detallaremos el funcionamiento de *rsyslogd*:

- Recolecta la totalidad de mensajes que provienen de servicios, aplicaciones y Kernel del sistema operativo.
- Redirige cada uno de estos mensajes al archivo de registro o Log correspondiente. Los archivos de registro acostumbran a estar almacenados en la ubicación /var/log.

Nota: *Rsyslog* también es capaz de redirigir los mensajes hacia un servidor Linux externo u otro equipo que esté en la misma red. Los protocolos usados para enviar los logs a equipos remotos pueden ser el *TCP, UDP y RELP*. De este modo *rsyslog* actúa como un servidor o cliente de Logs.

Para realizar las tareas que acabo de citar *rsyslogd* opera del siguiente modo:

- Las aplicaciones, los servicios y el Kernel escribirán sus mensajes en diversos *socket* y buffer de memoria. Algunos de los *socket* y *buffer* de memoria son */dev/log, /dev/kmsg, /proc/kmsg*, etc.
- Acto seguido el demonio rsyslogd leerá/obtendrá el contenido de los distintos socket y buffer de memoria.
- Rsyslogd procesará los mensajes de los socket y buffer de memoria en función de la configuración/reglas de los ficheros /etc/rsyslog.conf o /etc/rsyslog.d/50-default.conf en Ubuntu. Los ficheros de configuración que acabo de citar especifican el fichero o log en que se deben almacenar cada uno de los mensajes adquiridos por rsyslogd.

Nota: Un socket es un fichero especial que se usa para transferir información entre distintos procesos. En otras palabras, los procesos pueden leer y escribir datos en un socket/fichero con el fin de comunicarse entre ellos.

Para ver los *sockets* de nuestro sistema operativo podemos usar el comando:

```
# systemctl list-sockets –all
```

Para comprobar que el demonio *rsyslogd* está activo podemos ejecutar el siguiente comando en la terminal:

```
# service rsyslog status
```

Fichero de configuración de rsyslogd como acabamos de ver, *rsyslogd* lee los mensajes de varios *Socket y buffer* de memoria. Acto seguido rsyslogd ubica cada uno de los mensajes en los ficheros/logs pertinentes. Las reglas que rsyslogd utiliza para procesar los mensajes provenientes del Socket se definen en los ficheros */etc/rsyslog.d/50-default.conf* y/o */etc/rsyslog.conf*. A través de estos ficheros podremos configurar los siguientes aspectos:

- Definir el propietario y el grupo al que pertenecen los Logs.
- Establecer los permisos de los Logs y de los directorios que contienen los Logs.
- Definir si queremos disponer un log individual para registrar la actividad de un determinado servicio o utilidad. Por ejemplo, para registrar todos los mensajes del servicio cron en la ubicación */var/log/cron.log* descomentaremos la línea cron.* */var/log/cron.log* del fichero de configuración */etc/rsyslog.conf* o */etc/rsyslog.d/50-default.conf* en Ubuntu.
- Definir el nivel de prioridad de las actividades que queremos registrar. A modo de ejemplo, con el código "*kern.crit /var/log/kernlog*" el fichero kernlog únicamente registrará las actividades del Kernel que se consideren críticas. Los niveles de prioridad existentes son: *emerg, alert, crit, err, warning, notice, info y debug.* Si usamos el símbolo * el Log contendrá la totalidad de Logs independientemente de su nivel de prioridad.

Para consultar y editar el fichero de configuración deben ejecutar los siguientes comandos:

```
# nano /etc/rsyslog.conf
# nano /etc/rsyslog.d/50-default.conf
```

Antes de hacer cambios en la configuración estándar, es necesario hacer una copia de seguridad.

Apagado del Sistema Muy Lento Hay veces que el apagado del sistema Linux es muy lento, esto se debe a que el período de espera predeterminado para el apagado generalmente se establece en 90 segundos. En caso necesario, si alguna aplicación no responde, su sistema intenta forzar la detención de los servicios después de este período de tiempo.

Si queremos que el sistema Linux se apague rápidamente, podemos cambiar este período de espera. Encontraremos todas las configuraciones de *systemd*[288] en el archivo de configuración ubicado en: */etc/systemd/system.conf.*

Primero es recomendable comprobar qué procesos están causando un apagado prolongado en Linux, para ello usamos:

[288] Este archivo está lleno con muchas líneas que comienzan con #, representan los valores predeterminados de las entradas del archivo.

```
# journalctl -rb -1
```

una vez identificado es recomendable hacer una copia del archivo de configuración, usando:

```
# cp /etc/systemd/system.conf /etc/systemd/system.conf.orig
```

localizamos la línea que contiene el texto:

```
#DefaultTimeoutStopSec=90s
```

y la cambiamos por un valor a algo más conveniente como 5 o 10 segundos (no olvidemos quitar el # del inicio de línea):

```
DefaultTimeoutStopSec=5s
```

grabamos el archivo y reiniciamos el sistema. Esto ayudará a reducir el tiempo de espera para el apagado del sistema.

Mecanismo de Rotación de Logs Si observamos los archivos Logs, notaremos que muchos tienen un nombre parecido. A modo de ejemplo supongamos la siguiente situación[289]:

```
auth.log
auth.log.1
auth.log.2.gz
auth.log.3.gz
auth.log.4.gz
```

[289] Existe la opción de gestionar ficheros comprimidos, gracias a los comandos *zgrep, zegrep, zless, zmore, zdiff, zcmp, zcat*. Como te puedes dar cuenta la función de cada uno de estos comandos será la misma que su homónimos sin «z» (*grep, egrep, less, more, diff, cmp, cat*) pero para ficheros comprimidos con *gzip* y extensión *.gz*. De forma análoga para archivos comprimidos usando *bz2*, están los comandos *bzgrep, bzegrep, bzless, bzmore, bzdiff, bzcmp, bzcat* y para archivos comprimidos usando *xz*, están los comandos *xzgrep, xzegrep, xzless, xzmore, xzdiff, xzcmp, xzcat*.

La totalidad de archivos que he nombrado contienen información del mismo tipo. En este caso existen 4 archivos que han sido creados por el mecanismo de rotación de Logs.

La función del mecanismo de rotación de Logs es evitar que los Logs consuman todo el espacio de almacenamiento y sean fáciles de consultar. Si analizamos el comportamiento del Log *auth.log* veremos que semanalmente realiza las siguientes operaciones:

- El contenido que estaba en el fichero *auth.log.4.gz* se borrará y perderá de forma definitiva.
- El contenido que estaba en el fichero *auth.log.3.gz* se trasladará al fichero comprimido *auth.log.4.gz*.
- El contenido que estaba en el fichero *auth.log.2.gz* se traspasará al fichero comprimido *auth.log.3.gz*.
- El contenido que estaba en el fichero *auth.log.1* se comprimirá y trasladará al fichero *auth.log.2.gz*.
- Finalmente, la información almacenada en *auth.log* se trasladará al fichero *auth.log.1* y el fichero auth.log quedará vacío y listo para seguir registrando información.

Trabajando de esta forma únicamente estaremos almacenando los logs de las últimas 4-5 semanas. Por lo tanto, para consultar las autenticaciones al sistema operativo de 3 semanas atrás deberemos consultar el Log *auth.log.3.gz*.

La rotación de los Logs es una tarea programada que se realiza de forma periódica. Los Logs se rotan en función de un criterio que nosotros mismos podemos configurar. El criterio de rotación de Logs se establece en los ficheros de configuración ubicados en */etc/logrotate.d*

A modo de ejemplo si quieren ver la configuración establecida para rotar el Log *auth.log* ejecuten el siguiente comando:

```
# less /etc/logrotate.d/rsyslog
```

13.11 Particionamiento Estándar y LVM

La capacidad de almacenamiento de los equipos de cómputo se ha gestionado a través de los tamaños de las unidades de disco y las configuraciones de las particiones durante décadas. Claramente, esas estrategias funcionan bien y son confiables. Sin embargo, hay muchos beneficios al repensar la gestión del almacenamiento en los equipos. Así que en este apartado comparamos la administración y el particionamiento del almacenamiento estándar con el Administrador de Volumen Lógico (LVM).

Gestión de Almacenamiento Tradicional se utiliza el término de gestión de almacenamiento tradicional para describir el proceso de particionamiento, formateo y montaje de la capacidad de almacenamiento desde una unidad de disco. Solo que así suena mejor y lo engloba, evitando ser repetitivo.

El espacio de almacenamiento generalmente se administra en función de la capacidad máxima de las unidades de disco individuales. El resultado es que cuando un administrador de sistemas piensa en el almacenamiento, lo hace en función de cada unidad. Un ejemplo práctico, si un equipo tiene tres unidades de disco duro de 4 TB cada una, el administrador del sistema considera el almacenamiento literalmente, tengo tres unidades de 4 TB para trabajar, no más no menos.

El flujo de trabajo de esta gestión, para recordar, sería:

- Instalar una nueva unidad de disco, para nuestro ejemplo disco de 4 TB
- Particionar la unidad, utilizando *fdisk* o *gparted* para crear una o más particiones. Por ejemplo usando:

  ```
  # fdisk /dev/sdb
  ```

- Crear un sistema de archivos, usando:

  ```
  # mkfs.ext4 /dev/sdb1
  ```

  para crearla bajo *ext4* apuntado a la partición sdb1

- Crear un punto de montaje, usando:

```
# mkdir /nuevo_storage
# mount /dev/sdb1/ nuevo_storage
```

- Confirmar la capacidad de almacenamiento

```
# du -h /nuevo_storage
```

Finalmente:

- Configurar el espacio para montar en el arranque, editamos el archivo */etc/fstab* para montar el sistema de archivos en el arranque. Y listo.

Administrador de Volumen Lógico (LVM) la capacidad de almacenamiento tradicional se basa en la capacidad del disco individual. LVM usa un concepto diferente, ya que el espacio de almacenamiento se administra combinando o agrupando la capacidad de las unidades disponibles. Para aclarar, con el almacenamiento tradicional, tres discos de 4 TB se manejan individualmente, mientras que con LVM, esos mismos tres discos se consideran 12 TB de capacidad de almacenamiento agregado. Esto se logra designando los discos de almacenamiento como volúmenes físicos PV (Physical Volume) o capacidad de almacenamiento utilizable por LVM. Luego, los PV se agregan a uno o más grupos de volumen VG (Volume Group). Los VG se graban en uno o más volúmenes lógicos LV (Logical Volume), haciendo la magia, y logrando que se trabajen como particiones tradicionales.

Su flujo de trabajo sería similar a lo siguiente:

- Instale una nueva unidad de disco duro
- Designa volúmenes físicos

Los volúmenes físicos (PV) son discos o particiones que están disponibles para LVM como capacidad de almacenamiento potencial, teniendo identificadores y metadatos que describen cada uno. Es interesante notar que, a diferencia de RAID[290], los PV no tienen que ser del mismo tamaño o en discos que tengan la misma velocidad. Puede mezclar y combinar tipos de unidades para crearlos.

[290] Un RAID (Redundant Array of Independent Disks) es una matriz de al menos dos unidades de almacenamiento distintas convertidas en una única unidad lógica grande. La finalidad de un sistema RAID de discos es la de proteger los datos en caso de que un disco duro falle, o en algunos casos tiene como función principal mejorar la velocidad de lectura

```
# pvcreate /dev/sdb1
# pvcreate /dev/sdc
```

Como puedes ver, estos dos ejemplos son ligeramente diferentes. El primero designa la partición 1 del HDD b como PV. El segundo establece la capacidad total del HDD c como PV. Puedes verificarlo usando

```
# pvdisplay
```

ya que te permite ver todos los PV configurados en el equipo.

Administrar Grupos de Volumen designados los PV tocaría combinarlos en una VG, pueden convivir más de una VG y los HDD pueden ser parte de más de un VG (pero cada PV solo puede participar de una VG).

PV 1→ VG 1

VG 1 – VG 2 – VG 3

HDD b = VG 1 – VG 2 – VG 3

Utiliza el comando *vgcreate* para crear un nuevo VG, este debe tener al menos un miembro (o sea las PV). La sintaxis del comando es así:

```
# vgcreate VG_name PV_members
```

Por ejemplo, crear el VG, llamado *vg01* con */dev/sdb1* y */dev/sdc* como miembros PV:

```
# vgcreate vg01 /dev/sdb1 /dev/sdc
```

Como para los PV, existe *pvdisplay*, para los VG se usa *vgdisplay*, seguido de la VG que queremos revisar:

```
# vgdisplay vg01
```

de varios discos que conforman un único volumen.

En otras palabras, consiste en crear un único volumen con varios discos duros funcionando en conjunto, y con este conjunto se puede conseguir redundancia (tolerancia a fallos en el caso de que uno falle, conocido como Disk Mirroring) o mayor velocidad (conocido como Disk Striping), haciendo que ese conjunto sea en realidad un tándem.

En los RAID por Hardware, la organización de los dispositivos de almacenamiento individuales recae en un Hardware especial y potente, también llamado controladora RAID.

En un RAID por Software, un Software que se ejecuta directamente en la CPU del servidor asume la gestión de la cuota de almacenamiento.

Administrar Volúmenes Lógicos Propiamente Dicho el VG se puede subdividir en uno o más volúmenes lógicos (LV). Estos volúmenes lógicos se utilizan luego como si fueran particiones tradicionales. El VG tiene una capacidad total y luego una parte de esa capacidad se asigna a un volumen lógico. Para el caso de los LV, usamos *lvcreate*, para crearlos, grabándose en la VG designada. Siendo su sintaxis, con algunas opciones prácticas:

```
# lvcreate -L tamaño -n LV_name VG_name
```

donde -L brinda el tamaño en Gb o Tb, por ejemplo 4T (por cuatro Terabytes), si fueran 2 Terabytes, sería 2T, etcétera, mientras que -n para nombrar el LV. Por ejemplo:

```
# lvcreate -L 10G -n LV_almacen vg01
```

aquí está el comando para crear un LV de 10 GB llamado LV_almacen extraído del grupo de volúmenes vg01, y si es que lo has adivinado puedes usar *lvdisplay*, para verificar la configuración de la LV

```
# lvdisplay /dev/vg01/LV_almacen
```

Establecer un sistema de archivos y un punto de montaje. una vez que se crea un LV, se administra como cualquier otra partición. Se necesita un sistema de archivos y un punto de montaje, de la misma forma que en la partición estándar.

- Ejecute el comando mkfs.ex4 en la LV.
- Cree un punto de montaje utilizando mkdir.
- Montar manualmente el volumen con el comando mount o edite el archivo /etc/fstab para montar el volumen automáticamente cuando se inicie el sistema.
- Utilice el comando df -h para verificar que la capacidad de almacenamiento esté disponible.

Escalabilidad Uno de los beneficios de las configuraciones LVM es la capacidad de escalar el almacenamiento de manera fácil y rápida. Por lo general, los administradores necesitan escalar (aumentar la capacidad), en varios escenarios, siendo esta una solución recurrente. Vale la pena señalar que también se ve en la necesidad de reducir la capacidad de almacenamiento, así que tocare ambos puntos, a continuación:

Incrementar la Capacidad puedes agregar capacidad de almacenamiento al LV. Esto es útil si los usuarios consumen más espacio del previsto. El proceso es bastante lógico:

- Instala un disco de almacenamiento y luego configura como PV. Si ya hay un disco con espacio libre disponible, puedes usarlo también.
- Una vez que se designa la nueva capacidad para LVM, puede agregarla al VG, aumentando el tamaño del grupo con:

  ```
  # vgextend vg01 /dev/sdb2
  ```

- A continuación, agregue parte o todo el nuevo espacio de almacenamiento de VG al LV que necesita expandirse. Ejecutando el comando:

  ```
  # lvextend -L 4T /dev/vg01/LV_almacen
  ```

- Finalmente, amplíe el sistema de archivos. Tanto ext4 como XFS admiten esta capacidad, por lo que cualquier sistema de archivos está bien. Desmonta el sistema de archivos usando el comando *umount*:

  ```
  # umount /nuevo_storage
  ```

  seguido de el comando básico para ext4:

  ```
  # resize2fs /dev/vg01/LV_almacen 4T
  ```

Reducir Capacidad reducir el espacio de almacenamiento es una tarea menos común, pero vale la pena conocerlo. El proceso ocurre en el orden opuesto al de la expansión del almacenamiento. Un detalle, los sistemas de archivos XFS en realidad no se reducen. En su lugar, haga una copia de seguridad del contenido y luego restauré a un LV recién redimensionado. Puede usar la utilidad *xfsdump* para lograr esto. Mientras que ext4 si lo hace.

- Comenzamos por desmontar:

  ```
  # umount /dev/vg01/LV_almacen
  ```

- A continuación, use el comando *resize2fs* para reducir el tamaño del sistema de archivos. Se recomienda que ejecute *fsck* en sistemas de archivos ext4 antes de reducirlos, además de una copia de seguridad de los datos en el LV en caso de que ocurra algo inesperado (siempre vale la pena tomarse el tiempo y evitar pérdidas de datos sensibles).

  ```
  # resize2fs /dev/vg01/LV_almacen 4T
  ```

Tampoco se puede reducir un sistema de archivos a un tamaño menor que la cantidad de datos almacenados en él. Utilizamos el comando *lvreduce* para reducir el espacio de almacenamiento asignado al LV. Esto devuelve la capacidad de almacenamiento potencial al VG.

```
# lvreduce -L 2T /dev/vg01/LV_almacen
```

es fundamental darse cuenta de que el comando anterior establece el nivel de almacén en 2T. No elimina dos terabytes del LV existente. Configura el LV en dos terabytes. Es posible decirle a *lvreduce* que reste una cantidad de espacio de la capacidad existente usando un comando muy similar:

```
# lvreduce -L -2T /dev/vg01/LV_almacen
```

atento al - (guión) antes del tamaño de 2T, lo que indica que quiero que se reste esa cantidad de espacio de la capacidad de nivel de ventas existente. La diferencia entre los dos comandos es pequeña pero importante. Ahora tiene la capacidad devuelta al VG para usarla en otro LV. El VG también se puede encoger, por si se te ocurrió la consulta.

Flexibilidad la capacidad también se puede reasignar fácilmente con LVM. Puedes reducir la capacidad en un VG y agregarlo a otro. Esto se logra reduciendo el sistema de archivos y luego quitando el LV del VG. Digamos que tienes un servidor con 10 TB de capacidad. Utilizando los procesos anteriores, haz creado dos LV de 5 TB cada uno. Después de unas semanas, descubres que debiste haber creado LV de 7 TB y 3 TB en su lugar. Puede eliminar 2 TB de capacidad de uno de los grupos de volumen y luego agregar esa capacidad al otro VG. Esto es mucha más flexibilidad que las ofertas de particiones tradicionales. También admite configuraciones RAID, duplicación y otras configuraciones avanzadas que lo convierten en una solución aún más atractiva.

Sin lugar a duda LVM (Logical Volume Manager) es una excelente manera de agregar flexibilidad a tus necesidades de almacenamiento y, en realidad, no es mucho más complicado que la administración de unidades tradicional.

13.12 Instalación de los Paquetes más Usados

Existen distintas distribuciones de Linux[291] para instalar, una de las más ampliamente usadas es Debian GNU/Linux y sus distribuciones derivadas como Ubuntu.

Debian GNU/Linux es una distribución sólida como una roca con más de 64,419 paquetes disponibles en sus repositorios oficiales. Es adecuado para servidores, estaciones de trabajo, dispositivos móviles y sistemas integrados. Además tiene un sistema de instalación simple y limpio que permite instalarlo con poco esfuerzo y soporta las siguientes arquitecturas:

- 32-bit PC (i386) y 64-bit PC (amd64)
- 64-bit ARM (arm64)
- ARM EABI (armel)
- ARMv7 (EABI hard float ABI, armhf)
- little-endian MIPS (mipsel)
- 64-bit little-endian MIPS (mips64el)

[291] Una lista de las distribuciones de Linux y su árbol de vida puede verse en la página Web http://futurist.se/gldt/

- 64-bit little-endian PowerPC (ppc64el)
- IBM System z (s390x)

Entre los diferentes entornos de escritorio que pueden ser seleccionados al momento de instalar son:

- GNOME
- KDE Plasma
- LXDE
- LXQt
- MATE
- Xfce

Versiones de Debian GNU/Linux siempre mantiene al menos tres versiones en mantenimiento activo: "estable", "en pruebas" e "inestable" ("stable", "testing" y "unstable"). Estas permiten a cada tipo de usuario actualizar la distribución con paquetes estables, en pruebas o inestables pero que proporcionan características de última generación.

Estable

- La distribución "estable" contiene la publicación oficial más reciente de Debian GNU/Linux.
- Esta es la versión de producción de Debian GNU/Linux , cuyo uso recomendamos principalmente.
- La versión "estable" actual de Debian GNU/Linux es la 12, llamada "bookworm". Fue publicada originalmente el 10 de junio del 2023.

En pruebas

- La distribución "en pruebas" ("testing") contiene paquetes que aún no han sido aceptados en la rama "estable", pero están a la espera de ello. La principal ventaja de usar esta distribución es que tiene versiones más recientes del Software.
- Vea las Preguntas frecuentes de Debian GNU/Linux si desea más información sobre qué es "pruebas" y cómo se convierte en "estable".
- La distribución actual de "en pruebas" es "trixie".

Inestable

- La distribución "inestable" es donde tiene lugar el desarrollo activo de Debian GNU/Linux. Generalmente, esta distribución es la que usan los desa-rrolladores y aquellos que quieren estar a la última. Es recomendable que las personas que usan "inestable" se suscriban a la lista de correo debian-devel-announce para recibir notificaciones de los cambios importantes, por ejemplo sobre actualizaciones que pueden fallar.
- La distribución "inestable" siempre se llama "sid".

Hay muchas razones por las que Debian GNU/Linux es distinta al resto de distribuciones Linux del mercado. Es distinta en primer lugar por contar con tres documentos que determinan su evolución: su Contrato Social, su Constitución y su guía de desarrollo de Software Libre.

En esos documentos se establecen los principios de una distribución que "se mantendrá 100% libre" –tuvieron problemas con proyectos como Firefox por esa filosofía– y que "no esconderá los problemas", mientras que en su Constitución se define el funcionamiento interno de una comunidad en el que las decisiones las toman diversos tipos de miembros, tales como los desarrolladores o los líderes del proyecto.

Esa parte algo más organizativa deja clara la seriedad de un proyecto que siempre ha sido conservador en su aproximación al software: en Debian GNU/Linux (sobre todo en la rama Stable) uno no puede contar habitualmente con lo último de lo último, pero de lo que sí puede estar seguro es de que la distribución es especialmente estable y fiable.

Esa actitud algo más conservadora probablemente acabó provocando que usuarios independientes, grupos de usuarios y empresas y organismos desarrolladores de Software acabaran tomando Debian GNU/Linux como base para sus propias distribuciones derivadas.

Es así como se creó Ubuntu, la que probablemente es la distribución GNU/Linux y que debutó en 2004 para tratar de popularizar el uso de unas distribuciones con fama de complejas para el usuario novel.

De hecho en su logotipo se leyó durante mucho tiempo la frase "Linux for human beings" ("Linux para seres humanos"), y buena parte de su oferta inicial estaba dirigida a lograr facilitar el uso de Linux a todo tipo de usuarios, no a los tradicionales usuarios de Linux que habitualmente contaban con ciertos conocimientos técnicos.

En Distrowatch donde desde hace años se mantiene un seguimiento de la aparición (y desaparición) de distribuciones Linux, se contabilizan en septiembre del 2020 a 391 distribuciones derivadas de Debian GNU/Linux, con más de 121 de ellas aún activas. Debian GNU/Linux mantiene su propio censo al respecto, por supuesto, pero es que además invita a todo el que quiera a crear su propia derivada –de hecho cuentan con unas versiones especiales llamadas Debian Blends– con una serie de guías para lograrlo.

En realidad aquí cuentan no solo las distribuciones derivadas directas, sino también aquellas "derivadas de derivadas", y es que por ejemplo un gran número de distribuciones no parten ya de Debian GNU/Linux, sino directamente de Ubuntu para luego ofrecer ciertas características diferenciales que sus creadores estiman interesantes para cierto nicho de usuarios.

Debian GNU/Linux es Grande por Muchas Cosas muchas han sido las características que han llevado a Debian GNU/Linux donde está ahora, pero sin duda una de ellas es su gestor de paquetes, Advanced Package Tool o APT, un conjunto de comandos que desde 1998 hicieron que la instalación, actualización y eli-minación de nuevos paquetes software fuera tan potente, eficiente y sencilla para los usuarios.

También es legendaria su atención a todo tipo de arquitecturas, descartadas a menudo por grandes desarrolladores comerciales como Apple, Microsoft o Google, pero que tienen en Debian GNU/Linux a su mejor aliada.

De hecho es uno de los proyectos software que más arquitecturas soporta, con versiones tanto oficiales como no oficiales. Aquí tenemos soporte tanto para plataformas muy extendidas (las x86-64 e IA-32 que se utilizan en proce-

sadores de Intel y AMD, así como MIPS y distintas versiones de ARM) como para otras mucho menos populares como DEC Alpha, HP PA RISC, Intel Itanium, Motorola 68k, PowerPC, SPARC o RISC-V.

Debian GNU/Linux en particular (como Linux en general) es el mejor ejemplo de esa ubicuidad y versatilidad de un sistema operativo que permite por ejemplo que sea protagonista en todo ese segmento de mini-pcs orientados al segmento Maker y de la educación que nos ha conquistado con proyectos como las Raspbery Pi o Arduino.

Su presencia es también notable en un segmento radicalmente distinto. De lo "pequeño" que proponen las Raspberry Pi Debian GNU/Linux también es referente en el segmento de los servidores: según W3Techs la presencia de sistemas Unix en servidores Web es del 68,1% frente al 31,9% de servidores Windows.

De esos servidores basados en distintas versiones de Unix y sobre todo -las estadísticas aquí son imprecisas- distribuciones Linux, Debian GNU/Linux es protagonista con un 23,8% del total, sólo por debajo del 35,3% de una Ubuntu que desde hace años se ha ido dando cuenta de que ese segmento era importante para su parte comercial.

Ciclo de Vida de las Versiones Debian GNU/Linux anuncia su nueva versión estable de manera regular. Los usuarios pueden esperar unos 3 años de soporte completo para cada versión, y 2 años de soporte extra «LTS».

Índice de Versiones la siguiente versión de Debian 13 GNU/Linux se llama "trixie" -se espera para mediados o finales del 2026-, Debian 12 "bookworm" -la actual estable-, Debian 11 ("bullseye"), Debian 10 ("buster"), Debian 9 ("stretch"), Debian 8 ("jessie"), Debian 7 ("wheezy"), Debian 6.0 ("squeeze"), Debian 5.0 ("lenny"), Debian 4.0 ("etch"), Debian 3.1 ("sarge"), Debian 3.0 ("woody"), Debian 2.2 ("potato"), Debian 2.1 ("slink"), Debian 2.0 ("hamm").

Cómo Descargar una Distribución Linux La mayoría de las distribuciones Linux están disponibles bajo un formato llamado imagen *.iso* -no hay que pensar que el término imagen se refiere a un gráfico-. Estamos hablando de un archivo informático que utiliza un estándar que permite garantizar que la copia descargada es igual que el original almacenado en el servidor.

Esto es muy importante. Al poder hacer una verificación de integridad podemos confirmar que la distribución Linux fue descargada correctamente y que lo que vamos a instalar es una copia exacta de la publicada por los desarrolladores. Cualquier problema durante la descarga podría terminar en una instalación incompleta o fallida. Además, la verificación permite evitar sustituciones maliciosas.

¿Cómo se Hace la Verificación? todas las distribuciones permiten verificar la integridad de la imagen descargada comprobando su valor de Hash. Cada archivo contiene datos únicos, y cuando se le aplica un cierto algoritmo llamado «función de Hash criptográfico», se obtiene un valor de Hash que sólo es válido para ese archivo en su estado actual. Los algoritmos Hash más populares son MD5 (Compute and Check MD5 Message Digest), SHA (Secure Hash Algorithm) y BLAKE2 y las distribuciones incluyen el resultado correcto en algún lugar cerca del enlace de descarga para que puedas hacer la comprobación.

Generar la suma de verificación con md5sum, shasum y openssl, ejemplos:

```
$ md5sum debian-testing-amd64-netinst.iso
$ shasum debian-testing-amd64-netinst.iso
$ shasum -a256 debian-testing-amd64-netinst.iso
$ shasum -a512 debian-testing-amd64-netinst.iso
$ b2sum debian-testing-amd64-netinst.iso
$ openssl dgst -sha256 debian-testing-amd64-netinst.iso
```

Instalación de Debian GNU/Linux Debian[292] GNU/Linux se puede descargar de múltiples ligas (sin y con Firmware no libre integrado en el *.ISO*), una de ellas es:

https://www.debian.org/distrib/

Réplicas de Debian GNU/Linux en el Mundo En el mundo hay decenas de réplicas de Debian GNU/Linux, para acceder a alguna de ellas (en este apartado supondremos que se trabaja con la versión estable de Debian GNU/Linux), hay que modificar el archivo */etc/apt/sources.list*[293], mediante:

[292] Algunas de las razones para instalar GNU/Linux Debian están detalladas en su página Web https://www.debian.org/intro/why_debian.es.html

[293] En los inicios de cada línea, los *deb* se refiere a paquetes binarios y *deb-src* a paquetes de código fuente, solo necesarios para labores de desarrollo.

```
#[294] nano /etc/apt/sources.list
```

o

```
# apt edit-sources
```

para agregar la réplica más cercana a nosotros (supongamos qu es: ftp.us.debian.org), usamos:

```
deb http://ftp.us.debian.org/debian/ buster main
deb-src http://ftp.us.debian.org/debian/ buster main
```

Una vez actualizado el archivo */etc/apt/sources.list,* hay que actualizar las definiciones de paquetes disponibles en Debian GNU/Linux, usando:

```
# apt update
# apt upgrade
```

Una vez que *apt* ha terminado de instalar los paquetes, necesitamos solicitar que borre los archivos descargados para la actualización, usando:

```
# apt clean
```

Paquetes contrib y non-free Por omisión en Debian GNU/Linux sólo se instalan paquetes libres, pero hay otro tipo de paquetes útiles que no son libres o que tienen licencia distinta a la usada por Debian GNU/Linux, para poder tener acceso a ellos hay que modificar el archivo */etc/apt/sources.list*, mediante:

```
#[295] nano /etc/apt/sources.list
```

[294] En el Bourne Shell y sus derivados como BASH el *Prompt* que nos permite escribir los diferentes comandos, generalmente termina con el carácter:

- $ para usuario sin privilegios
- # para el administrador, conocido como *root*

[295] En el Bourne Shell y sus derivados como BASH el *Prompt* que nos permite escribir los diferentes comandos, generalmente termina con el carácter:

- $ para usuario sin privilegios
- # para el administrador, conocido como *root*

o

```
# apt edit-sources
```

en nuestro caso el contenido de */etc/apt/sources.list*, es:

```
deb http://ftp.us.debian.org/debian/ buster \
main contrib non-free
deb-src http://ftp.us.debian.org/debian/ buster \
main contrib non-free



deb http://security.debian.org/debian-security \
buster/updates main contrib non-free
deb-src http://security.debian.org/debian-security \
buster/updates main contrib non-free
```

esto permite tener acceso a paquetes libres (main) y no libres o con licencias distintas a Debian GNU/Linux GPL de Linux (contrib y non-free).

Una vez actualizado el archivo */etc/apt/sources.list,* hay que actualizar las definiciones de paquetes disponibles en Debian GNU/Linux, usando:

```
# apt update
# apt upgrade
```

Una vez que *apt* ha terminado de instalar los paquetes, necesitamos solicitar que borre los archivos descargados para la actualización, usando:

```
# apt clean
```

Ahora el sistema está listo para poder instalar los paquetes que el usuario de la máquina requiere (hay más de 59,000 disponibles en la versión 11).

Conocer la Réplica con Mejor Velocidad de Descarga si necesitamos conocer cuales son las diez réplicas de Debian GNU/Linux que nos de la mejor velocidad de descarga podemos usar *netselect-apt*, se instala mediante:

```
# apt-get install netselect-apt
```

para luego correrlo usando:

```
# netselect-apt
```

el cual generará un archivo *source.list* en el Home del usuario *root* con la configuración óptima además genera una salida indicando las velocidades de acceso.

Rama de Desarrollo de Backports Algunas veces es necesario instalar en la versión estable algún paquete de la versión testing, pero que no rompa la estabilidad del sistema, para ello se desarrolló Backports[296]. Para usarlo, lo primero es agregar en el archivo */etc/apt/sources.list* las líneas:

```
deb http://deb.debian.org/debian/ bullseye-backports main
deb-src http://deb.debian.org/debian/ bullseye-backports main
```

también existen para los paquetes contrib y non-free:

```
deb http://deb.debian.org/debian/ bullseye-backports main \
contrib non-free
deb-src http://deb.debian.org/debian/ bullseye-backports main
\
contrib non-free
```

después de concluir la edición, es necesario ejecutar:

```
# apt update
```

Para instalar algún paquete, usar:

```
# apt -t bullseye-backports install nombreDelPaquete
```

Pese a no ser una opción recomendada, podemos actualizar todos los paquetes, usando:

```
# apt -t bullseye-backports update
# apt -t bullseye-backports upgrade
```

[296] https://backports.debian.org

Rama Experimental Además de la posibilidad de usar Backports, es posible también usar los repositorios de Debian experimental, pero esto debe ser la última opción para la instalación de paquetes, para ello debemos agregar en el archivo */etc/apt/sources.list* las líneas:

```
deb http://deb.debian.org/debian/ experimental main
deb-src http://deb.debian.org/debian/ experimental main
```

también existen para los paquetes contrib y non-free:

```
deb http://deb.debian.org/debian/ experimental main \
contrib non-free
deb-src http://deb.debian.org/debian/ experimental main \
contrib non-free
```

después de concluir la edición, es necesario ejecutar:

```
# apt update
```

Para instalar algún paquete, usar:

```
# apt -t experimental install nombreDelPaquete
```

13.12.1 Diferentes Kernels de Linux

Estable como su nombre indica, es la versión estable del Kernel de Linux, y es la que desarrolla Linus Torvalds junto a sus colaboradores. Se lanza una nueva actualización cada dos meses aproximadamente, y se sube el primer número una vez cada tres años, poco más o menos. Es lo que se usa en la mayoría de distribuciones, siempre y cuando no se decanten por uno como los que se describen a continuación.

LTS son las siglas de Long Term Support, lo que se traduce en que está soportado durante más tiempo. El tiempo de soporte depende de los mantenedores. Lo habitual es que estén soportados durante seis años, pero los mantenedores pueden decidir que este tiempo se vea reducido hasta los 3 años.

Lo bueno de un Kernel LTS es que no recibe nuevas funciones que podrían romper compatibilidades, eso y que reciben muchas actualizaciones correctivas. Por lo tanto, son más estables que los estables, valga la redundancia, pero no reciben novedades que no sean para arreglar algo.

Hardened es una versión «endurecida» del Kernel estable de Linux, más centrada en la seguridad y viene con parches que la versión estable aún no ha recibido. Tiene una configuración de seguridad.

Hay que tener en cuenta que la capa extra de seguridad podría hacer que algunos programas no funcionarán con este Kernel, por lo que sólo se debe usar si la seguridad es lo más importante para nuestro uso y equipo.

Linux-libre es un proyecto que mantiene varias versiones modificadas del Kernel de Linux a las que les quitan todo lo que no incluye código fuente y otro Software que use licencias privativas. Está recomendado por la Free Software Foundation.

Kernels de Tiempo Real (Real Time rt) y Baja Latencia estos Kernels están optimizados para aplicaciones que requieren un rendimiento en tiempo real predecible, como la producción de audio o la automatización industrial. Priorizan la latencia mínima entre la asignación de una tarea a la CPU y su ejecución. La gente suele confundir este Kernel y cree que con el la pc funcionara mas rápido, pero deben entender que esta optimizado simplemente llamadas en tiempo real, ideal para aquellos que monitorean lo que graban y realizan producción musical. No obstante este Kernel NO debería usarse si nos dedicamos a jugar, visualizar contenido o Stremear, ya que podría ser contradictorio.

Kernels Zen y Liquorix estos Kernels se enfocan en mejorar el rendimiento del escritorio de Linux. Implementan optimizaciones para el programador de CPU, la compresión de memoria y la velocidad de lectura/escritura de RAM. Si buscas un mayor rendimiento para juegos o uso general del escritorio, estos Kernels pueden ser una buena opción. Es una gran Kernel para ordenadores de sobremesa, y genera mejoras de rendimientos. Sin embargo no hace magia, esto se traduce en mas calor, por lo cual no se recomienda tanto en Notebooks dado que las temperaturas elevadas y sus sistemas de refrigeración son muy inferiores a los de los ordenadores de sobremesa.

Kernel Xan Mod similar a los Kernels Zen y Liquorix, Xan Mod ofrece optimizaciones para mejorar el rendimiento del sistema. Se destaca por su enfoque en la personalización, permitiendo a los usuarios ajustar y configurar diversos aspectos del Kernel para adaptarlo a sus necesidades específicas.

Tiene una configuración que, dependiendo del hardware, genera un mejor rendimiento que el Kernel Zen (liquorix). Sin embargo es probable que veas un poco mas de uso de RAM/CPU lo cual es una parte necesaria para visualizar esa alta de performance que vamos a sentir.

TKG (The Kernel Git) no es un Kernel en sí, sino un sistema de compilación que permite a los usuarios crear Kernels personalizados. Ofrece gran flexibilidad para elegir parches específicos e integrar optimizaciones personalizadas. Es ideal para desarrolladores y usuarios avanzados que desean un control total sobre el Kernel. En base a esto salen algunos Kernel preconfigurados optimizados para usuarios finales, pero la idea principal del proyecto es facilitar a los desarrolladores o usuarios avanzados la configuración y/o compilación del Kernel.

Kernel de Android es una versión especializada diseñada para dispositivos móviles como teléfonos inteligentes y tabletas. Incluye optimizaciones para la gestión de energía, la suspensión y reactivación, y controladores de dispositivos específicos para dispositivos móviles. Este particularmente deberíamos usarlo si viene instalado en nuestro dispositivo, pero rara vez deberíamos intentar compilarlo nosotros dado que no mantiene la compatibilidad con los ordenadores de sobremesa.

Elegir el Kernel Adecuado la mejor opción de Kernel para ti dependerá de tus necesidades y prioridades. Si buscas estabilidad y confiabilidad, el Kernel estándar o LTS es una buena elección. Para mayor seguridad, considera el Kernel reforzado. Si requieres un rendimiento en tiempo real, opta por un Kernel de tiempo real o baja latencia. Para dispositivos móviles, el Kernel de Android es esencial. Y si buscas optimizar el rendimiento del escritorio, prueba los Kernels Zen, Liquorix o Xan Mod. TKG es ideal para usuarios avanzados que desean un control total sobre el Kernel.

Recuerda: Puedes probar diferentes Kernels reiniciando en ellos y comparando el rendimiento y la estabilidad. Si experimentas problemas, siempre puedes volver al Kernel original. Asegúrate de instalar los controladores adecuados para tu Hardware con cada Kernel que uses.

13.12.2 Última Versión del Kernel GNU/Linux en Debian

Una de las razones más importantes para usar la rama de desarrollo de Backports, es poder instalar una de las últimas versiones del Kernel GNU/Linux (en la versión de Debian GNU/Linux 12 se usa el Kernel 6.1, en la rama de desarrollo de Debian GNU/Linux 13 es 6.12 y en la rama experimental es 6.13, la última versión estable del Kernel GNU/Linux es 6.14 y la de desarrollo es 6.15), mediante:

Para ver que versiones están disponibles usamos:

```
# apt -t buster-backports search linux-image
```

Para instalar una, por ejemplo usamos:

```
# apt -t buster-backports install \
linux-image-6.13.0-1-amd64
```

o instalar Kernel y Header correspondiente:

```
# apt -t buster-backports install \
linux-headers-6.13.0-1-amd64 \
linux-image-6.13.0-1-amd64
```

Para conocer qué versiones del Kernel tenemos instaladas usamos:

```
$ dpkg –list | egrep -i –color 'linux-image|linux-headers'
```

En caso de necesitar quitar cualquier versión del Kernel que no se esté usando, por ejemplo hacemos:

```
# apt purge linux-image-6.12.0-1.bpo.8-amd64
```

o

```
# apt-get –purge remove linux-image-6.12.1-0.bpo.8-amd64
```

Otros Kernels GNU/Linux Actualmente existen una gran cantidad de distribuciones de GNU/Linux que vienen muy optimizadas intentando conseguir la mejor desenvoltura de su arquitectura y configuraciones de serie. En el caso de la configuración por omisión de Debian GNU/Linux y Ubuntu, están pensadas para que sean lo más robusta posible y que se use en todas las circunstancias imaginables, por ello están optimizadas de forma muy conservadora para tener un equilibrio entre eficiencia y consumo de energía. Pero es posible agregar uno o más Kernels GNU/Linux generados por terceros que contenga las optimizaciones necesarias para hacer más eficiente y competitivo en cuestiones de gestión y ahorro de recursos del sistema.

Hay varias opciones del Kernel GNU/Linux optimizado (Liquorix viene optimizado para multimedia y Juegos, por otro lado XanMod tiene uno para propósito general, otro aplicaciones críticas en tiempo real y otro más que se adapta al tipo de tarea en ejecución), estos se pueden instalar[297] mediante el uso de los comandos *dpkg* o *apt* (después de agregarlo a nuestro repositorio */etc/apt/sources.list.d/*), de esta forma siempre podremos tener la última versión del Kernel junto con la actualización básica de nuestro sistema GNU/Linux.

Además, sí instalamos cualquiera de los distintos Kernels, siempre podemos seleccionar alguno de los instalados al momento de arrancar nuestro equipo para usarlo de acuerdo a las actividades requeridas en ese momento. Y en caso necesario, es fácil su desinstalación y continuar usando el Kernel que teníamos por defecto.

Por otro lado, existe una versión completamente libre del Kernel de GNU/Linux (Linux-Libre) el cual es un Kernel despojado de elementos de Firmware y controladores que contienen componentes no libres o fragmentos de código cuyo alcance está limitado por el fabricante.

Linux-libre es el núcleo recomendado por la Free Software Foundation y una pieza principal de las distribuciones GNU totalmente libres de fragmentos privativos o Firmwares incluidos en Linux sirven para inicializar los dispositivos o aplicarles parches que solventan fallas del Hardware que no pudieron ser corregidas antes de ser puestos a disposición de los usuarios.

Además, Linux-libre deshabilita las funciones del Kernel para cargar componentes no libres que no forman parte del suministro del Kernel y elimina la mención del uso de componentes no libres de la documentación. El Kernel de

[297] Para ver las opciones de optimización del Kernel y como instalarlo ver la página Web de cada proyecto: Liquorix, XanMod, Linux-Libre.

Linux-libre se utiliza en distribuciones como Dragora Linux, Trisquel, Dyne, Bolic, gNewSense, Parabola, Musix y Kongoni.

Compilar Nuestro Kernel La distribución del Kernel se realiza como un paquete o en forma de código fuente, aunque lo habitual es utilizar el Kernel que nos proporciona el creador de nuestra distribución, es posible compilarnos un Kernel a medida para nuestra máquina.

Si usamos un «Kernel de Linux» pre-compilado desde los repositorios, se pueden conseguir estas «ventajas»:

- Una tarea que no requiere mayores niveles de conocimientos técnicos sobre la materia.
- Un proceso de instalación rápido y automatizado, programado por la comunidad experta de la Distribución.
- Un proceso de instalación fácil y seguro de realizar, al igual que al actualizarlo y/o desinstalarlo.

La compilación de un nuevo Kernel ofrece las siguientes «ventajas»:

- Aumentar las prestaciones: compilando un Kernel para que se ajusten a las características propias de nuestra máquina
- Soporte nuevo de Hardware: a medida que se lanzan nuevas versiones, se incrementan la cantidad de dispositivos soportados.
- Seguridad: como es lógico las prioridades a la hora de lanzar una nueva versión del Kernel incluyen la corrección de todos los problemas de seguridad detectados.
- Nuevas características: cada versión del Kernel que se lanza, se mejora sus funciones de gestión de memoria, sistema de ficheros, escalabilidad, etc.

Por el contrario, el proceso de compilar el Kernel presenta los siguientes «inconvenientes»:

- Complejo: el proceso de compilar el Kernel puede ser complejo para usuarios poco avanzados. Se debe tener un cierto conocimiento de Hardware y de compilación si queremos afrontar con éxito la operación. En algunas ocasiones, al compilar el Kernel, también debemos actualizar diversos programas del sistema con lo que la complicación es mayor.

- Problemas de inestabilidad: dado que el Kernel es una parte fundamental del sistema operativo, una compilación realizada con opciones equivocadas puede volver inestable la máquina, los cuales pueden producir "cuelgues", corrupción de datos, etc. Por lo que conviene tenerlo muy en cuenta por lo que debemos tener todo respaldado para evitar pérdidas de información.

Si queremos compilar nuestro propio Kernel, es necesario considerar que se necesitan al menos 25 GB de espacio disponible, porque compilar el Kernel de Linux ocupa alrededor de 22 GB e invertiremos entre 15 minutos hasta más de una hora (esto dependerá de las características de nuestro equipo). Aunque los archivos generados para el nuevo Kernel ocupa muchísimo menos.

Para realizar todo el proceso sin problemas, necesitamos tener instalados los siguientes paquetes: build-essential, libncurses6-dev, fakeroot, libssl-dev, gcc, bc, bison, flex, libelf-dev, kmod, cpio. Aunque realmente el manual oficial de Debian nos dice que solo se necesitan los paquetes: build-essential, libncurses-dev y fakeroot.

```
# apt install build-essential libncurses-dev fakeroot libssl-dev
gcc bc bison flex libelf-dev kmos cpio
```

Descargaremos el último Kernel estable, y evitaremos descargar Kernels de versiones RC, ya que pueden ser inestables. Se recomienda elegir un Kernel de lanzamiento estable y de largo plazo. Para ello nos vamos a La Web oficial del Kernel Linux: http://www.kernel.org.

Una vez que haya elegido la versión del Kernel que desea descargar. Con el link copiado, podremos descargarlo desde consola usando wget o git (también puede ser descargado gráficamente desde el navegador, por supuesto):

Antes de que podamos compilar la fuente de Kernel Linux que hemos descargado, necesitamos tener un archivo de configuración, llamado .config. Este archivo de configuración le dice al compilador qué características tiene que configurar, tales como: controladores, sistemas de archivos, seguridad,

etc. Es muy difícil crear y/o editar manualmente el archivo de configuración .config, ya que cuenta con miles de opciones y módulos.

Entonces, tenemos dos formas de crear el archivo de configuración:

1- Creamos un nuevo archivo de configuración (.config) basado en los dispositivos que actualmente tenemos conectados a nuestro ordenador.

2- Copiar el archivo de configuración del Kernel actual de Debian (.config).

Al compilar una versión más reciente del núcleo o Kernel Linux, probablemente usemos la misma configuración propuesta por Debian. En este caso, y en lugar de reconfigurar todo desde cero, es suficiente copiar el archivo /boot/config-versión (donde versión: es la del núcleo utilizado actualmente, que se encuentra en el directorio /boot) y lo copiamos en el directorio /kernel/linux-X.Y.Z cambiándole el nombre a .config

Si ya estamos seguros de la configuración, solo necesitamos lanzar el proceso de compilación y una vez concluido procedemos a instalarlo y reiniciamos nuestro equipo. Dado que el Kernel es vital para nuestro equipo, siempre debemos documentarnos sobre las nuevas características y proceder con cautela al configurarlo, pero vale la pena este proceso para determinadas situaciones.

13.12.3 Actualizar Versión de Debian

Si nuestro sistema esta funcionando en la versión estable de Debian y no queremos reinstalar el sistema operativo, es posible pasar a la versión de prueba usando el procedimiento que describimos a continuación[298].

Actualizar los paquetes existentes usando

```
# apt update && sudo apt upgrade -y
```

Cambiar en: /etc/apt/sources.list

De la versión estable (bullseye):

deb http://security.debian.org/debian-security bullseye-security main contrib non-free

deb-src http://security.debian.org/debian-security bullseye-security main contrib non-free

[298] Siempre es mejor opción hacer una instalación limpia y completa en la nueva versión, pero si no lo desea hacer, entonces puede ser buena opción la actualización de versión. Claro, es recomendable después de hacer los respaldos pertinentes.

deb http://deb.debian.org/debian/ bullseye-updates main contrib non-free
deb-src http://deb.debian.org/debian/ bullseye-updates main contrib non-free
deb http://security.debian.org testing-security main

A la versión de prueba (bookworm):

deb http://deb.debian.org/debian/ bookworm main non-free contrib
deb-src http://deb.debian.org/debian/ bookworm main non-free contrib
deb http://security.debian.org/debian-security bookworm-security main contrib non-free
deb-src http://security.debian.org/debian-security bookworm-security main contrib non-free

Actualizar el índice de los paquetes:

```
# apt update
```

Actualizar paquetes básicos:

```
# apt upgrade
```

Actualizar la versión:

```
# apt full-upgrade
```

para concluir debemos:

```
# apt clean
# apt autoremove
```

y al terminar ya podemos reiniciar el equipo y al arrancar tendremos la versión de prueba actualizada en nuestro sistema.

13.12.4 Deshabilitar Mitigaciones del Kernel para Meltdown, Spectre, etc

De los problemas detectados en el 2017 (Meltdown y Spectre) en los procesadores de los equipos de cómputo, comunicaciones y redes de internet que usamos, se han seguido encontrando una larga sucesión de fallos[299] hasta el día de hoy en múltiples procesadores actuales y anteriores. Estos son problemas de diseño de los procesadores de Intel, AMD, IBM POWER y ARM, esto significa que procesos con privilegios bajos (aquellos que lanzan las aplicaciones de usuarios convencionales) podían acceder a la memoria del Kernel del sistema operativo.

Un ataque que explotase dicho problema permitiría a un Software malicioso espiar lo que están haciendo otros procesos y también espiar los datos que están en esa memoria en el equipo de cómputo (o dispositivo móvil) atacado. En máquinas y servidores multiusuario, un proceso en una máquina virtual podría indagar en los datos de los procesos de otros procesos en ese servidor compartido.

En GNU/Linux, el Kernel (si usamos una versión actualizada) nos indica las fallas del procesador a las que es vulnerable, usando:

```
$ cat /proc/cpuinfo
$ lscpu
```

Ya que los fallos encontrados son problemas de diseño en los procesadores, no es posible encontrar solución por Hardware para los procesadores existentes y dado que constantemente aparecen nuevas formas de explotar dichos fallos, la única manera de mantener el equipo de cómputo, comunicaciones y redes de internet a salvo es mediante Software que debe implementar las soluciones en los sistemas operativos. En particular en el Kernel de Linux se trabaja en parchar en cada versión del Kernel todos los fallos reportados, por

[299] Entre las distintas vulnerabilidades detectadas y sus variantes resaltan: Meltdown (AC, DE, P, SM, SS, UD, GP, NM, RW, XD, BR, PK, BND), Spectre (PHT, BTB, RSB, STL, SSB, RSRE), PortSmash, Foreshadow, Spoiler, ZombieLoad (1 y 2), Kaiser, RIDL, Plundervolt, LVI, Take a Way, Collide+Probe, Load+Reload, LVI-LFB, MSD, CSME, RYZENFALL (1, 2, 3, 4), FALLOUT (1, 2, 3) , CHIMERA (FW, HW), MASTERKEY (1, 2, 3), SWAPGS, ITLB_Multihit, SRBDS, L1TF, etc. Más información en:

https://cve.mitre.org
https://meltdownattack.com/

esto y por otra gama de fallos e inseguridades es necesario mantener siempre el sistema operativo y sus aplicaciones actualizadas.

Se ha considerado que una solución basada únicamente en Software para estas fallas degrada el desempeño de los equipos de cómputo entre un 20 y un 40 por ciento dependiendo de la tarea que realizasen y el procesador del equipo. Si bien es necesario tener estas mitigaciones en el Kernel, bajo ciertas circunstancias se pueden prescindir de ellas. Por ejemplo cuando nuestro equipo es para uso personal y del que se puede prescindir temporalmente del uso de internet mientras es usado para cálculo intensivo, procesamiento de vídeo, imágenes, entre otros o es una máquina virtual.

Si necesitamos deshabilitar las mitigaciones del Kernel a los fallos, podemos agregar en la configuración de arranque en el GRUB del sistema: **mitigations=off**. Esto deshabilitará todas y cada una de las mitigaciones (también es posible indicar cuales mitigaciones aplicar según el procesador y nuestras necesidades) permitiendo usar la totalidad del rendimiento del CPU y si además nuestro equipo es sacado de red, entonces es totalmente seguro su uso.

El proceso de deshabilitar las mitigaciones al Kernel se pueden hacer de forma temporal en el proceso de arranque del GRUB, al solicitar editar la configuración, en donde aparece:

```
linux /vmlinuz-5...  ro single
```

agregando al final: **mitigations=off.** Si se desea hacer permanente, hacemos lo mismo en el archivo:

/boot/grub/grub.cfg

pero si se actualiza la versión del Kernel, se deberá hacer nuevamente la edición correspondiente.

13.12.5 Paquetes para Diferentes Necesidades

Metapaquetes Son una solución para grupos de personas con necesidades específicas, no solamente proporciona colecciones manejables (metapaquetes) de paquetes de programas específicos, sino que también facilita la instalación y configuración para el propósito convenido. Cubren los intereses de distintos grupos de personas, su objetivo común es simplificar la instalación y administración de ordenadores para una audiencia determinada[300].

[300] https://www.debian.org/blends/

Por ejemplo los hay para:

- Ciencia (science-)
- LaTeX (texlive-)
- Astronomía (astro-)
- Química (debichem-)
- Sistema de Información Geográfica (gis-)
- Medicina (med-)
- Multimedia (multimedia-)
- Educación (education-)
- Junior (junior-)
- Juegos (games-)

para conocer todos los paquetes que inician (por ejemplo textlive), usar:

```
$ apt search textlive
```

para saber qué hace cada paquete usar:

```
$ apt show texlive
```

para instalar, usar:

```
# apt install texlive
```

A continuación daremos un listado paquetes[301] agrupados según su uso, muchos de ellos hacen cosas parecidas o se complementan, para saber qué hace cada paquete usar:

```
$ apt show paquete
```

para conocer todos los paquetes que inician (por ejemplo textlive), usar:

```
$ apt search textlive
```

para instalar, usar:

```
# apt install paquete(s)
```

[301] https://packages.debian.org/stable/

Eliminar las versiones locales de documentación no usadas

localepurge

Buscar y/o quitar paquetes no usados y huérfanos

deborphan bleachbit

Drivers para nuestro Hardware

drive

Firmware libre y no libre

firmware-linux firmware-linux-free firmware-linux-nonfree \

firmware-misc-nonfree

Actualización del Microcódigo

amd64-microcode intel-microcode

Linux Vendor Firmware Service (LVFS) un demonio para actualizar Firmware

fwupd

```
# fwupdmgr get-devices
# fwupdmgr refresh
# fwupdmgr refresh –force
# fwupdmgr get-updates
# fwupdmgr update
# fwupdmgr report-history
# fwupdmgr clear-history
```

en caso de haber descargado el archivo .CAB del Firmware del proveedor, lo instalamos con:

```
# fwupdmgr install archivo.cab
```

Trusted Plataform Module (TPM) de Trusted Computing Group (TCG)

```
tmp2-tools
tpm-tools trousers
```

Manejo de máquinas virtuales QEMU/KVM

```
qemu-kvm qemu qemu-utils xtightvncviewer
```

Compartir escritorio remoto

```
remmina remmina-plugin-* tigervnc-standlone-server vinagre \
rdesktop krdc
```

VNC server

```
tightvncserver
```

VNC cliente

```
xtightvncviewer
xrdp gnome-boxes remmina
```

Aplicaciones de monitoreo y rendimiento

```
sysstat dfc htop atop vtop gtop iotop iftop ifstat lsof \
glances nmon collectl itop vnstat vnstati dnsutils fio \
nmap iptraf iptraf-ng nethogs nload bmon tcpdump \
nethogs slurm tcptrack bwm-ng cbm speedometer \
netwatch trafshow netload traceroute bashtop stress \
nslookup sysbench whowatch macchanger sysstat \
netstat-nat jnettop fping pktstat dstat tuptime \
darkstat iperf3 netperf monitorix psmisc smem \
smemcap smemstat
```

Aplicaciones para conocer el Hardware de un equipo

procinfo lshw hwinfo inxi cpuid hardinfo cpufrequtils i7z \
dmidecode cpufreq-info lspcmcia lsscsi gparted

Aplicaciones para administración

rcconf powertop cpulimit timelimit

Aplicaciones para probar el ancho de banda entre dos servidores

iperf iperf3 bwctl-client bwctl-server netcat

Comandos usuales, nueva implementación

dfc gcp ripgrep ack ncdu most dog htop tldr fd jq bat \
exa pydf fd-find zoxide

Manejador de tareas

tasque

Seguridad

wipe secure-delete scrub

Recuperador de datos

gddrescue

Sincronización de archivo y directorios

rsync grsync rclone luckybackup syncthing

Trabajar en múltiples servidores

clusterssh pssh pscp tmux-cssh

Descarga de archivos URL

wget uget curl lftp woof httpie youtube-dl
persepolis

Acelerador de descargas de archivos URL

aria2 axel

Cliente de correo

thunderbird evolution kmail geary sylpheed claws-mail
mutt alpine

Logo de la distribución

linuxlogo screenfetch neofetch

Grabar sesiones de la terminal

asciinema

Terminales

sakura miterm xterm rxvt kitty lilyterm guake qterminal \
eterm roxterm terminator tilix terminology tilda yakuake \
konsole terminix deepin-terminal kitty gnome-terminal \
stterm mate-terminal xfce4-terminal sakura lxterminal \
cool-retro-term

Editor hexadecimal

wxhexditor ghex bless okteta hexer
xxd hexedit hexcurse hexyl ncurses-hexedit

Intérpretes de órdenes

csh tcsh ksh fish zsh tsch dash dsh busybox

Manejo de archivo sobre CLI

mc ranger vifm nnn lfm wcm ytree

Navegadores de red

chimera2 chromium conkeror dillo edbrowse epiphany-browser \
iceweasel konqueror midori netrik netsurf netsurf-fb arora hv3 \
surf uzbl tfirefox-esr qupzilla netsurf-gtk falkon

Actualizar el navegador por omisión

```
# update-alternatives –config x-www-browser
```

Navegadores de red CLI

elinks elvis-tiny links links2 lynx lynx-cur w3m browsh

Actualizar el navegador por omisión

```
# update-alternatives –config www-browser
```

Búsquedas en Google CLI

googler

Búsquedas en Duck Duck Go CLI

ddgr

Manejo de particiones NTFS

ntfs-3g ntfsprogs scrounge-ntfs

Manejo de SAMBA

samba smbclient cifs-utils

Manejo de particiones

parted partimage gparted testdisk genisoimage f3

Servidor de SSH

openssh-server
dropbear

Clientes de SSH

openssh-client filezilla gftp mosh

Respaldo y recuperación de datos

timeshift bacula dump

Manejo de documentos e imágenes

science-typesetting texlive-full texlive-science \
texlive-extra-utils texlive-latex-base \
texlive-latex-recommended texlive-publishers \
texlive-bibtex-extra texstudio pandoc texmaker \
inkscape kile lyx gummi texstudio latexila kexi \
latexila texworks libreoffice calligra abiword \
gnumeric

htmldoc converseen djview4 okular gv evince diffpdf \
mupdf pdf-presenter-console evince xpdf okular atril pdfcrack \
qpdf pdfsam pdfshuffler pdfmod pdfposter pdfchain pdf2djvu \
catdoc chktex cxref cxref-doc latex2rtf antiword a2ps \
unoconv bookletimposer qpdfview rst2pdf xchm \
archmage qpdfview-ps-plugin qpdfview qpdfview-djvu-plugin \
kchmviewer pdfarranger zathura zathura-djvu mupdf \
pdftk poppler-utils pdf2svg imagemagick pdfgrep \

ispanish wspanish texlive-lang-spanish myspell-es \
translate-Shell myspell-en-us

Microsoft Fonts:

ttf-mscorefonts-installer fonts-arkpandora

Manejo de archivos DJVU

djview djview4 okular evince zathura zathura-djvu \
qpdfview qpdfview-djvu-plugin

Edición y manejo de archivos Epub

mupdf calibre fbreader sigil

Manejo de notas

rednotebook cherrytree zim focuswriter tomboy basket \
elpa-org feathernotes xpad gnote nvpy
remind

Tomar notas, dibujar y convertir a PDF

xournal cherrytree

Tipografías

^ttf-
^fonts-*
^xfonts-*

para conocer todos los paquetes asociados, usar:

$ apt search ^ttf-

Microsoft Fonts:

ttf-mscorefonts-installer fonts-arkpandora

Tipografía para programadores

fonts-hack-ttf fonts-powerline fonts-ubuntu-console \
fonts-inconsolata fonts-fantasque-sans fonts-firacode

Editores de gráficos

gpaint inkscape imagemagick dia xfig scribus blender \
calibre kdenlive kazam pinta krita pencil2d graphviz \
calligra feh ufraw showfoto rawtherapee mypaint fotoxx \
kolourpaint darktable pencil2d photoflare openshot \
textdraw converseen
gimp gim-data gimp-data-extra gimp-cbmplugs gimp-dcraw \
gimp-dds gimp-gap gimp-gluas gimp-gmic gimp-gutenprint \
gimp-lensfun gimp-texturize gimp-ufraw gimp-plugin-registry

Editores de vídeo blender pitivi shotcut lives lives-plugins

Editores de vídeo no lineales

kdenlive openshot flowblade

Paquetes para generar videos a partir de imágenes

openshot imagination photofilmstrip kdenlive openshot

Paquetes para tomar vídeos o imágenes del escritorio

simplescreenrecorder gtk-recordmydesktop byzanz kazam \
recordmydesktop vlc vokoscreen obs-studio peek ksnip \
screengrab flameshot imagemagick gnome-screenshot \
gtk-vector-screenshot deepin-screenshot kde-spectacle \
grim scrot shutter gimp

Digitalización de imágenes y documentos

gscan2pdf simple-scan skanlite xsane

Multimedia y conversión de formatos de audio y video

vlc amarok mplayer xbmc ffmpeg mpg123 clementine audacity \
libxine2-bin libxine2-ffmpeg libxine2-x libxine2-plugins ffmpeg \
libavcodec-extra gstreamer1.0-fluendo-mp3 vorbis-tools \
gstreamer1.0-plugins-ugly gstreamer1.0-plugins-bad \
gstreamer1.0-pulseaudio kodi pitivi ffmpegthumbnailer \
libvpx5 handbreak lmms handbrake soundconverter \
converseen libav-tools

Reproducir DVDs

libdvd-pkg

Reproductor música CLI

cmus mpd moc

Messenger y IRC Chat

pidgin pidgin-guifications pidgin-plugin-pack pidgin-themes \
finch skype4pidgin kopete kmess empathy telegram-desktop \
weechat hexchat irssi konversation quassel
oysttyer rtv

Paquetes matemáticos

science-statistics science-mathematics education-mathematics \
science-viewing science-numericalcomputation xabacus euler \
gretl lybniz mathomatic pari-gp rkward xaos yacas bliss \
geogebra mumps-test nauty netgen netgen-doc kmplot \

eukleides genius pspp pdl yorick gnuplot scilab maxima\
wxmaxima xmaxima mathomatic-primes octave octave-gsl \
julia sagemath plplot-tcl grace labplot gle-graphics

R

r-base 'r-cran*' 'r-bioc*'

para conocer todos los paquetes asociados, usar:

```
$ apt search ^r-cran
```

Editor de R para KDE:

```
# apt install rkward rkward-data
```

Generación de gráficos:

```
# apt install rlplot
```

Algunos paquetes internos de uso común:

```
R2WinBugs mcmcplots devtools Rattle e1071 FactiMineR \
Ellipse xlsx hsaur shiny
```

Lenguaje de programación C y C++

build-essential manpages-dev glibc-doc glibc-doc-reference \
gcc-doc-base gcc-doc c++-annotations-pdf catch clang\
g++ cppcheck splint cccc autoconf automake make \
cmake scons jam ohcount lldb

Lenguaje de programación Fortran

gfortran gfortran-doc fortran77-compiler fortran95-compiler \
fortran-compiler cfortran

Lenguaje de programación Java

default-jdk

o

openjdk-14-jre openjdk-14-jdk openjdk-14-doc

default-jre

o

openjdk-14-jre

Actualizar la versión por omisión de JAVA

```
# update-alternatives –config java
$ java –version
```

Editores

https://netbeans.apache.org/download/index.html
https://www.eclipse.org/downloads/
http://brackets.io/
https://www.jetbrains.com/idea/download/#section=Linux
https://www.oracle.com/tools/downloads/Jdeveloper-12c-downloads.html
http://www.drjava.org/
https://www.jgrasp.org/
https://www.bluej.org/
http://www.jcreator.com/index.htm
https://codenvy.com/
https://atom.io/
https://www.sublimetext.com/

Lenguaje de programación Python 2

ipython python-matplotlib python-numpy python-scipy \
python-pandas python-sympy python-nose python-dev \
spyder python-rpy2 python-rpy python-axiom python-pip \
python python-tk idle python-pmw python-imaging bpython \
python-pandas python-sklearn python-sklearn-doc python-pip \
wxglade python-rpy2 pypy pypy-tk python-gnuplot pypy-six \
python-mpi4py pyqt5-dev pyqt5-dev-tools pyflakes2

Lenguaje de programación Python 3

ipython3 python3 idle3 python3-pip python3-matplotlib \
python3-rpy2 python3-numpy python3-scipy spyder3 \
bpython3 python3-pandas python-wxgtk3.0-dev xonsh \
python3-pyqt5 python3-pyqtgraph mypy python3-numba \
python3-mpi4py pyreport jython sagemath-jupyter \
jupyter-console jupyter-notebook jupyter python3-plplot \
pyflakes3 flake8 black

Jupiter (instalar después de python 3)

```
# pip3 install jupyter
# pip3 install matplotlib
# pip3 install ipywidgets
# jupyter nbextension enable –py –sys-prefix widgetsnbexten-
sion
```

Editores Python

eric pyzo pyzo-doc thonny

https://www.jetbrains.com/pycharm/
http://www.pydev.org/
https://wingware.com/

Otros Lenguajes de programación

mono-complete

haskell-platform

golang

scala

ghc ghc-prof ghc-doc ghc-mod

swi-prolog swi-prolog-doc swi-prolog-java swi-prolog-x

coq coq-doc coq-theories

Librerías matemáticas

gsl-bin libgsl-dev gsl-ref-html

Editores, IDEs, Debugers y herramientas para programación

scite jedit kate gedit nedit emacs qtcreator kwrite pluma \
geany geany-plugins anjuta anjuta-extras vim-gtk mousepad \
codelite codelite-plugins tea eric neovim neovim-qt \
bluefish bluefish-plugins codeblocks codeblocks-contrib \
gambas3

fte fte-console fte-terminal nano joe vim vim-python-jedi \
vim-tlib vim-latexsuite vim-nox neovim micro kakoune \
vim-athena jed tilde emacs ne alpine-pico mle

kdiff3 meld diffuse dirdiff kompare numdiff colordiff dwdiff \
wdiff xxdiff tkdiff ndiff ccdiff xxdiff diffmerge
ddd-doc ddd gdb xxgdb kdbg nemiver lldb cppcheck strace \
kcachegrind time valgrind valgrind-mpi
ggcov lcov gcov
alleyoop astyle c2html java2html code2html c2html autodia \
txt2html html2text moreutils indent

Generador de documentación

aptitude install doxygen graphviz

Control de cambios en un proyecto

git git-all gitk gitg git-cola git-gui qgit tig vim-fugitive git-extras
mercurial
subversion rapidsvn
cvs tkcvs

Manejo de UML

umbrello dia plantuml

Remplazo de grep para proyectos grandes

ripgrep

silversearcher-ag

ack-grep

```
# dpkg-divert –local –divert /usr/bin/ack –rename –add \
/usr/bin/ack-grep
```

Cómputo en paralelo usando MPI (es necesario instalar ssh cliente y servidor)

lam-runtime xmpi libmpich-dev mpich mpi-default-dev \

mpi-default-bin openmpi-bin valgrind-mpi

Para que en la ejecución de MPI no pida la clave de usuario:

```
$ ssh-keygen -t rsa
```
en cada pregunta responder con ENTER, para después copiar usando:
```
$ ssh-copy-id usuario@servidor
```

Cómputo distribuido

pvm pvm-dev

perl python-mpi

OpenDx Visualization Data Explorer

dx dx-dox dxsample

HDF5

hdfview hdf5-tools

Generador de mallas en 3D

gmsh

Servidores

apache2 libapache2-mod-evasive apachetop httpd
php libapache2-mod-php php-mysql php-gd phpmyadmin
mysql-common mysql-client mysql-server mytop mysql-admin \
ferret mysql-workbench mysql-workbench-data mycli \
automysqlbackup
postgresql postgresql-client postgresql-doc postgresql-contrib \
autopostgresqlbackup
sqlite3 sqlite3-doc sqlitebrowser
redis-server
mongodb mongodb-clients mongodb-server
mariadb-client mariadb-server mariadb-common
nginx lighttpd tomcat9 nodejs caddy openlitespeed cherokee
firebird-server firebird-utils
httpie
samba smbclient
vsftpd
openssh-server openssh-client
nfs-kernel-server
rsync
cups

Paquetes para Notebook

wireless-tools acpi-support cpufrequtils acpi wpasupplicant \
powertop vbetool acpid wicd-curses apmd pcmciautils \
pm-utils radeontool anacron avahi-autoipd bluetooth iban \
kmilo Laptop-detect Laptop-mode-tools Laptop-net guessnet \
noflushd bluez-utils bluez-pcmcia-support

Paquetes para redes inalámbricas

wireless-tools wireshark kismet kwavecontrol wicd-curses \
aircrack-ng macchanger prismstumbler swscanner wavemon \
wmwave ifrename ndisgtk ndiswrapper-common \
ndiswrapper-dkms

Construccion de CDlive

live_helper mbr sysLinux

Instalar distribuciones Linux/BSD en USB drives

gnome-disk-utility unetbootin unetbootin-translations gksu \
testdisk

Poder correr aplicaciones de Windows mediante WINE

wine

```
# dpkg –add-architecture i386 && aptitude update && \
aptitude install wine32
```

Lector de correo tipo PINE

alpine

Paquetes para instalar impresoras locales y remotas

system-config-printer system-config-printer-udev cups \
cups-pk-helper
hplip

FEniCS

fenics

FreeFem

freefem freefem-doc
freefem++ freefem++-doc
freefex-examples

GMT (The Generic Mapping Tool)

gmt gmt-coastline-data
gmt-examples gmt-manpages gmt-doc-pdf gmt-tutorial-pdf

Manejo de escáner

gscan2pdf xsane simple-scan eikazo

Reconocimiento Óptico de Caracteres OCR

tesseract-ocr tesseract-ocr-spa tesseract-ocr-eng gocr ocrad

Búsqueda de archivos duplicados

fslint fdupes jdupes findimagedupes duff rdfind hardlink

Utilerias de compactación y descompactación

arj bzip2 clzip cpio dtrx gzip kgb lbzip2 lhasa lrzip \
lzip lziprecover lzop ncompress p7zip-full p7zip-rar pax \
pbzip2 pigz pixz plzip rar rarcrack tarlz unace unar \
unace-nonfree unp unrar unrar-free unzip xz-utils zip \
zpaq zstd zutils
ark file-roller nemo-fileroller xarchiver engrampa
cabextract kgb xdms archivemount

Monitoreo de múltiples equipos

sinfo conky-all

Cliente para sincronización de hora NTP

ntpdate

```
# ntpdate -s -u 1.debian.pool.ntp.org
```

Manejadores de ventanas

openbox fluxbox dwm awesome xmonad i3 bspwm jwm \
herbstluftwm tilix ratpoison

Multiplexor de ventanas CLI

tmux tmate screen dvtm spectrwm qtile konsole \
qterminal terminator

Animaciones y Otros sobre CLI

sl figlet cmatrix bb fortune cowsay xcowsay toilet libaa-bin

Calculo de π

pi

uso

```
$ pi 200000
```

Sistema de Información Geográfica

gis-*

para conocer todos los paquetes asociados, usar:

```
$ apt search ^gis-
```

Crear y montar sistemas de archivos cifrados

gocryptfs cryfs encfs ecryptfs-utils

Cifrado de archivos y discos

tomb cryptmount gnupg cryptsetup gnome-disk-utility
gnupg openssl ccrypt mcrypt bcrypt zip 7-zip age

Generar contraseñas

gnupg openssl apg pwgen makepasswd

Gestor de contraseñas

kwalletmanager seahorse keepass2 keepassx keepassxc \
password-gorilla passwordsafe
kwalletcli lastpass-cli pass gopass yapet kpcli

Cortafuegos

nftables iptables

Sanbox

firejail jailkit

Recobrar archivos borrados

testdisk

Respaldo de información

deja-dup grsync timeshift bacua duplicity kopano-backup \
amanda-client amanda-server backupninja luckybackup \
backintime-gnome backintime-kde backintime-qt4 kup-backup \
backup-manager backup2l backupchecker backuppc kbackup \
rsync rdiff-backup rsbackup vbackup zbackup borgbackup \
rsnapshot rear rclone bup restic syncthing

Reproductores de música

cmus moc mpg123 mp3blaster

Clientes de Torrent

qbittorrent deluge transmission ktorrent uget biglybt
transmission-cli aria2 rtorrent delug-console

Conocer velocidad de la red

speedtest-cli speedmeter

Rootkit, Sniffer y Exploits

rkhunter chkrootkit debsecan aide lynis tiger

Sintetizador de Texto a Voz

speak-ng

Gadgets/Widgets

screenlets screenlets-pack-all

Otros

lolcat

13.12.6 Manejadores de Paquetes Multiplataforma

La principal diferencia entre los formatos de paquetes nativos y los formatos de paquetes independientes es que los primeros comparten dependencias con otros programas instalados en el sistema operativo. Es decir que si el programa Y necesita la dependencia 1 y esa dependencia fue instalada por el programa X que también la necesita, esa dependencia no volverá a instalarse.

Los programas empaquetados con formatos independientes incluyen todas las dependencias que necesitan para su funcionamiento. Es decir que la dependencia 1 se instalará cada vez que se instale un programa que la necesite.

La segunda diferencia es que los formatos de paquetes tradicionales deben ser construidos con las especificaciones de cada distribución. Es por eso que por más que Ubuntu sea una distribución derivada de Debian GNU/Linux, las diferencias son lo suficientemente importantes como para que los repositorios de la primera no puedan ser usados en la segunda.

La tercera diferencia es que cualquier modificación a una dependencia de los paquetes tradicionales puede afectar el funcionamiento de todos los demás que la necesitan. En cambio, las modificaciones a un programa en un formato independiente, no afectará al resto del sistema.

Dependiendo de las particularidades de cada distribución, es posible instalar las aplicaciones en formatos independientes desde un gestor de paquetes y automatizar su actualización con el gestor encargado de las mismas.

Flatpak, Snap, AppImage, seguro que son nombres con los que estás más que familiarizado. Los paquetes universales han irrumpido en el mundo Linux para poder funcionar en cualquier distribución y así quitar el problema de la fragmentación en cuanto a paquetes. Sin embargo, aún no son mayoría, aunque poco a poco va creciendo el número de Software que se empaqueta en estos tipos de paquetes.

¿Qué es Flatpak? Flatpak es un tipo de paquete universal y para la virtualización de aplicaciones para entornos GNU/Linux. Proporciona una sandbox aislada de procesos conocida como Bubblewrap o envoltorio burbuja. En él los usuarios pueden ejecutar las aplicaciones aisladas del resto del sistema, para mayor seguridad.

Lennart Pöttering fue el programador que lo propuso en 2013, y publicó un artículo al respecto un año más tarde para finalmente desarrollar la idea y formar parte del proyecto freedesktop.org., bajo el nombre de xdg-app, que es lo mismo que Flatpak. Y su popularidad desde el lanzamiento fue en aumento, actualmente cuenta con soporte en más de 20 distribuciones de las más populares.

¿Qué es Snap? Mientras que Flatpak tuvo sus orígenes en la comunidad de desarrollo de Fedora/Red Hat, Snap lo tuvo en Canonical, la empresa que desarrolló este tipo de gestión de paquetería tan peculiar. Un tipo de paquete universal que ya aceptan gran cantidad de distribuciones y apps empaquetadas en él. En este caso, los paquetes se ejecutan dentro de AppArmor, aunque se pueden ejecutar fuera de la sandbox.

Por cierto, hay que reconocer que existen otros paquetes como los AppImage, que cada vez cobra más y más importancia por su sencilla instalación, o mejor dicho, no instalación. Solo descargas y ejecutas el paquete y listo, como una especie de versión portable. Además, en el sitio oficial AppImage Hub podrás encontrar multitud de herramientas empaquetadas en este formato binario. En cuanto a la aseguridad, se pueden ejecutar dentro de la caja de arena o dentro de AppArmor, Bublewrap o Firejail.

En Ubuntu, el Centro de Software permite instalar tanto programas en formatos tradicionales como Snap, dándole preferencia a estos últimos. Por más que existe un plugin que permite que el Centro de Software de GNOME (del cuál se deriva el de Ubuntu) no funciona con esta distribución. En el caso de Ubuntu Studio, es posible activar la opción de usar paquetes Snap mientras que KDE Neon y Manjaro pueden funcionar con ambos formatos.

Las dependencias de los paquetes Flatpak y Snap que nadie habla En Linux hay muchas maneras de instalar un mismo Software, pero uno de los reclamos que podemos encontrar es que al instalar se incluye Software principal y dependencias en un mismo paquete, lo que les hace funcionar desde un principio, son más limpios y esas cosas, pero esto es una verdad a

medias.

Supongamos que no usamos ningún paquete Flatpak y queremos instalar sólo uno porque lo necesitamos. Por ejemplo la aplicación llamada *Immagini* con la que podremos crear *AppImages*, ese tipo de aplicación portable que se puede ejecutar, en teoría, en cualquier distribución Linux si la arquitectura es compatible. *Immagini* tiene un peso aproximado de 22.4 MB, pero para poder instalarla necesitamos... 1,325 MB de espacio. ¿Cómo?

Dependencias compartidas, pero dependencias al fin y al cabo Por ejemplo, cuando queremos instalar un programa que convierte archivos multimedia a otros formatos, si no lo tenemos ya es probable que nos descargue *FFmpeg* e *ImageMagick*, cada uno con unas cuantas dependencias más. Estas sí son dependencias al uso, pero las que se instalan junto a un paquete Flatpak o snap son lo necesario para que se pueda ejecutar ese programa en nuestra plataforma. Si la aplicación está escrita en GTK o tiene componentes de GNOME, nos instalará la plataforma de GNOME y sus traducciones. Cuando instalemos otro programa GTK/GNOME, esto ya lo tendremos, por lo que no será necesario y el peso de la aplicación ya será el que vemos en las tiendas de Software. En el caso de los paquetes Snap tenemos un poco lo mismo.

Vigilando las dependencias de los Flatpak y Snap Controlar los paquetes Flatpak que tenemos de más es más sencillo, puesto que hay varios comandos para eliminar lo que no está siendo usado. Los datos y Cache de la aplicación suelen estar almacenados en ~/.var/app, y se pueden eliminar perfectamente a mano porque está dentro de nuestra carpeta personal y sin protección, algo así como lo que hay dentro de .config. Si queremos eliminarlo con el terminal, tendremos que usar este comando:

```
$ flatpak uninstall –delete-data
```

Para eliminar las dependencias de un paquete, que para usar el nombre correcto deberíamos decir «runtimes», el comando sería:

```
$ flatpak uninstall –unused
```

Si lo que queremos es eliminarlo todo, deberemos escribir:

```
$ flatpak uninstall –all
```

El último está diseñado como medio para restablecer todo lo relacionado a Flatpak. Se podrá volver a instalar un paquete Flatpak, pero empezaremos de cero. Es para hacer una limpieza general.

En cuanto a los paquetes snap, no conozco nada parecido. Cuando instalamos una aplicación, ésta aparece dentro de la carpeta snap. Si eliminamos el paquete, su contenido desaparece, pero sus archivos de configuración no, y pueden estar en .config, .caché o en otra carpeta. Los runtimes o dependencias, junto a los paquetes, suelen estar en /var/snap/ o /var/lib/snapd, pero hay que tener cuidado con lo que tocamos aquí. Mi recomendación sería tirar de tienda de Software, y si tiene un apartado para ello, ir a la pestaña de snaps instalados. Si vemos algo que sabemos que no estamos usando, eliminarlo desde ahí.

También podemos escribir snap list, encontrar lo que sepamos que no estamos usando y eliminarlo con snap remove "paquete".

Aunque hay que saber que existen, y en ocasiones al ver lo que puede ocupar una aplicación al instalarla, no todo es malo. Las aplicaciones de Linux pesan muy poco, y eso es gracias a que existe Software y dependencias que se comparten con otros programas. Esto es perfectamente aplicable a los paquetes Flatpak y Snap: si no existieran estas dependencias, cada nuevo paquete que las necesitara debería incluirlas en él mismo, por lo que las aplicaciones podrían ser muy pesadas. Tal y como están las cosas, las únicas pesadas serán las primeras; las siguientes ya no tendrán que descargar nada extra.

Snap es el más nuevo de los formatos independientes ya que su desarrollo comenzó en el 2014. Está pensado no solo para ser usado en distribuciones Linux de escritorio si no también para Internet de las cosas, dispositivos móviles y servidores. Aunque es posible crear tiendas de aplicaciones independientes, en este momento solo existe una operada por Canonical, Snapcraft.

Aunque Snapcraft tiene un surtido de las aplicaciones más populares de código abierto, su fuerte son los programas desarrollados por empresas desarrolladores de software privativo y prestadoras de servicios en la nube.

Para instalar Snap. usamos:

```
# apt install snapd
```

Para buscar un paquete usamos:

```
$ snap find libreoffice
```

Para instalar un paquete usamos:

```
$ snap install <snap_name>
```

Para listar los paquetes instalados usamos

```
$ snap list
```

Para actualizar un paquete previamente instalado usamos:

```
$ snap refresh <snap_name>
```

Para desinstalar un paquete usamos:

```
$ snap remove <snap_name>
```

Flatpak aunque oficialmente Flatpak se lanzó en el 2015, es la continuidad de otro proyecto de formato universal conocido como xdg-app. Este proyecto nació con el objetivo de poder ejecutar aplicaciones en una caja de arena virtual segura, que no requiera privilegios de root ni suponga una amenaza de seguridad para el sistema.

Flatpak está enfocado en las distribuciones de escritorio también utiliza el concepto de tienda de aplicaciones siendo Flathub la más conocida. El punto fuerte es que suele tener las versiones más actualizadas de las principales aplicaciones de código abierto.

Para instalar Flatpak, usamos:

```
# apt install flatpak
```

para activar su integración con Gnome, usamos:

```
# apt install gnome-software-plugin-flatpak
```

Para buscar algún paquete, usamos:

```
$ flatpak search aplicación
```

Para instalar paquetes usamos:

```
$ flatpak install <remotes> <aplicacion>
```

por ejemplo:

```
$ flatpak install flathub org.libreoffice.LibreOffice
```

Algunos desarrolladores tienen su propio repositorio, podemos usarlo mediante:

```
$ flatpak install –from \
https://flathub.org/repo/appstream/com.spotify.Client.flatpakref
```

Si hemos descargado un paquete de la red, podemos instalarlo usando:

```
$ flatpak install <ApplicationID>.flatpakref
```

Supongamos que hemos descargado *net.poedit.Poedit.flatpakref*, lo instalamos con:

```
$ flatpak install net.poedit.Poedit.flatpakref
```

Si al instalar un paquete nos manda un error, para tratar de corregirlo podemos usar:

```
$ flatpak update -v
```

Podemos correr un Flatpak mediante:

```
$ flatpak run <ApplicationID>
```

si hemos instalado spotify, lo podemos correr usando:

```
$ flatpak run com.spotify.Client
```

Para listar los Flatpak instalados usamos:

```
$ flatpak list
```

Para desinstalar un Flatpak, usamos:

$ flatpak uninstall <ApplicationID>

por ejemplo:

$ flatpak uninstall com.spotify.Client

Para actualizar todas las aplicaciones Flatpak instaladas usamos:

$ flatpak update

Podemos remover las aplicaciones Flatpak no usadas mediante:

$ flatpak uninstall –unused

Appimage es el más antiguo de los formatos de paquetes independientes ya que se lanzó por primera vez en 2004. Fue el primer formato en seguir el paradigma de «Una aplicación-un archivo». Eso significa que cada vez que descargamos un archivo Appimage estamos descargando la aplicación y todo lo que necesita para funcionar. Si queremos usar la aplicación solo debemos darle permisos de ejecución y hacer doble clic sobre el icono que la identifica.

Appimage no utiliza el sistema de tienda de aplicaciones, pero, hay una página web en la que podemos encontrar una lista de todos los títulos disponibles.

Cargo es el gestor de paquetes predeterminado del lenguaje de programación Rust, el cual se usa para descargar dependencias de los paquetes Rust creados para compilar exitosamente los mismos, haciéndolos distribuibles y facilitando su carga a Crátes (crates.io), el registro de paquetes de la comunidad Rust. Para instalarlo usamos:

apt install cargo

para auto actualizar el paquete usamos:

$ cargo update

para instalar algún paquete usamos:

$ cargo install paquete

para desinstalar un paquete usamos:

$ cargo uninstall paquete

Alien es un convertidor de paquetes de línea de comandos que convierte entre diferentes formatos de paquetes de Linux como Red Hat *rpm*, Debian *deb*, Stampede *slp*, Slackware *tgz* y Solaris *pkg*, etc. Actualmente, Alien admite los siguientes formatos de paquete:

- Linux Standard Base (LSB)
- LSB-compliant .rpm packages
- .deb
- Stampede (.slp)
- Solaris (.pkg)
- Slackware (.tgz, .txz, .tbz, .tlz)

El programa *alien* viene al rescate cuando un paquete específico o una versión específica de un paquete no está disponible para su distribución de Linux. Podemos convertir fácilmente dicho paquete al formato de paquete preferido utilizando Alien e instalarlo en su sistema.

Alien no es sólo un convertidor de paquetes, también puede instalar los paquetes generados automáticamente después de la conversión de paquetes. Incluso puede tener la opción de convertir los Scripts que deben ejecutarse cuando se instala el paquete (es recomendable examinar las secuencias de comandos detenidamente y comprobar qué hacen estas secuencias de comandos antes de utilizar esta opción).

Para instalar el paquete usamos:

```
# apt install alien
```

La sintaxis general para convertir paquetes de Linux usando Alien de un formato a otro es:

```
$ alien [–to-deb] [–to-rpm] [–to-tgz] [–to-slp] [opciones] archivo [...]
```

Para convertir un paquete *.rpm* en un paquete *.deb*, simplemente ejecutamos alien como usuario root o sudo:

```
# alien –to-deb /path/to/file.rpm
```

Del mismo modo, para convertir el archivo .deb a .rpm, ejecutamos:

```
# alien –to-rpm /path/to/file.deb
```

Aquí está la lista de opciones para convertir paquetes de Linux a diferentes formatos:

- -d, –to-deb - Crea paquetes Debian (este es el predeterminado)
- -r, –to-rpm: crea paquetes rpm
- -l, –to-lsb: crea un paquete LSB
- -t, –to-tgz - Crea paquetes tgz
- –to-slp - Crea paquetes slp
- -p, –to-pkg: crea paquetes pkg de Solaris

13.12.7 Windows en Linux

Uno de los problemas más comunes al pasar de Windows a Linux es no poder usar los programas a los que estamos acostumbrados. Es cierto que cada vez hay más software disponible para Linux, y que los programas más comunes (como Chrome, Spotify o VLC) tienen sus respectivas versiones en este sistema. Sin embargo, hay otros programas que no tienen versión para Linux, como puede ocurrir con Office, o con Photoshop. En ese caso, o bien tendremos que buscar alternativas (que las existen, como LibreOffice y GIMP), o recurrir a una herramienta que nos va a permitir ejecutar cualquier programa o juego de Windows en Linux: Wine.

Wine nació inicialmente como un proyecto que buscaba crear un emulador de Windows. Su acrónimo era inicialmente «WINdows Emulator», aunque viendo su evolución, y la forma de funcionar, este acrónimo fue actualizado por «Wine Is Not an Emulator«. Y es que en realidad no es un emulador, sino que este programa está formado por un cargador de programas binarios junto a un conjunto de herramientas de desarrollo que permiten portar en tiempo real el código de las aplicaciones de Windows a Unix. Además, trae por defecto una gran cantidad de bibliotecas y librerías de manera que no tengamos problemas de dependencias.

Principales Características este programa es capaz de ejecutar sin problemas cualquier programa diseñado para cualquier versión de Windows, desde la 3.x hasta Windows 10. Eso sí, solo es compatible con programas Win32 (tanto de 32 bits como de 64 bits), por lo que no vamos a poder ejecutar las apps UWP de la Microsoft Store, al menos por ahora.

Entre toda la variedad de librerías, bibliotecas y recursos, podemos encontrarnos con prácticamente todas las bibliotecas de interrupciones para programas, lo que permite hacer llamadas INT en tiempo real. De esta manera, los programas no saben que se están ejecutando en un sistema operativo que no es Windows, simplemente se ejecutan. Y lo hacen igual que en él. Si algún programa, o juego, tiene dependencias especiales (por ejemplo, una DLL concreta) podemos añadirla fácilmente a Wine. Todas las librerías se encuentran dentro del directorio «~/.wine/drive_c/windows/system32», que equivale al directorio System32 de Windows.

Por supuesto, Wine tiene soporte para una gran cantidad de recursos gráficos. Los programas se pueden dibujar tanto en una interfaz gráfica X11 (el escritorio) como desde cualquier terminal X. Es compatible con las tecnologías OpenGL, DirectX y cuenta con soporte total para GDI (y parcial para GDI32). También permite y gestiona varias ventanas a la vez (del mismo programa, o de diferentes) y es compatible con los temas msstyle de Windows.

También es compatible con los controladores de sonido de Windows, y tiene acceso a los puertos del PC, al Winsock TCP/IP y hasta a los escáneres.

¿Qué Programas y Juegos Puedo Ejecutar con Wine? por desgracia, a pesar de tener una gran compatibilidad, Wine no es capaz de ejecutar el 100% de los programas y juegos de Windows en Linux. Y algunos, aunque se pueden ejecutar, no funcionan del todo bien. Para saber si un programa se puede ejecutar, o no, en Linux, podemos buscarlo en la red del proyecto. Allí vamos a encontrar una gran base de datos que nos va a permitir saber si un programa funciona va a funcionar, si no lo hace, o qué tal lo hace.

Además de poder buscar manualmente cualquier programa o juego, también vamos a encontrar una lista con los Top-10 que mejor funcionan. Los juegos «Platino» son los que funcionan de manera idéntica en Windows que en Linux, los «Oro» los que funcionan bien, pero requieren de alguna configuración especial y los «Plata», los que funcionan, pero tienen pequeños problemas. Los programas o juegos «Bronce» o «Basura» son los que no

funcionan.

Saca Todo el Partido a Wine con Estos Programas Wine, al final, es la parte más importante para poder usar programas de Windows en Linux. Sin embargo, su configuración, sobre todo para los programas que no tienen una clasificación de platino, puede ser algo tediosa. Por suerte, existen programas que, aunque se basan igualmente en Wine, nos ayudan a configurar cada uno de estos programas de manera automática para que nosotros no tengamos que hacer nada más.

La instalación y configuración de los programas y juegos de Windows para usarlos en Linux es lo peor. Si no tenemos muchos conocimientos podemos perder mucho tiempo y, además, no conseguiremos que todo funcione del todo bien. Aquí es donde entra en juego *PlayOnLinux*. Este programa, gratuito y de código abierto, busca ayudarnos con la instalación y configuración de los programas y juegos para hacerlos funcionar en este sistema operativo.

PlayOnLinux nos ofrece una completa base de datos de programas con sus correspondientes configuraciones óptimas de manera que nosotros solo tengamos que seleccionar el programa que queremos, cargarle su instalador y dejar que este complete el proceso de puesta en marcha. Nada más. Cuando acabe la instalación ya podremos abrir el programa o juego y empezar a usarlo.

Podemos descargar esta herramienta de forma totalmente gratuita desde su página Web, o desde una termina:

```
# apt install playonlinux
```

Descargar e Instalar Wine hay muchas formas de instalar Wine en Linux. Sus desarrolladores tienen binarios específicos para cada distribución, así como unos completos repositorios desde los que vamos a poder descargar y actualizar el programa desde terminal:

```
# apt install wine
```

WINE no es una aplicación que se inicia por sí sola. Es un backend que se invoca cuando se inicia una aplicación de Windows. Lo más probable es que su primera interacción con WINE ocurra cuando inicie el instalador de una aplicación de Windows.

Ejecutar e Instalar Programas Windows una vez instalado, Wine se ejecutará al hacer doble clic sobre cualquier archivo .EXE. Además, te permitirá instalar programas, como si estuvieras en Windows y pondrá los accesos directos en el menú principal bajo la categoría «Wine».

A pesar de lo que mucha gente cree, Wine sirve no sólo para correr aplicaciones «sencillas» de Windows, sino incluso juegos complejos. Es más, está demostrado que terribles juegazos como Sim 3, Half Life 2, Command & Conquer 3, Star Wars: Jedi Knight, o importantes suites como Microsoft Office funcionan a la perfección.

Bottles Es una aplicación alternativa para la fácil ejecución de Software de Windows sobre GNU/Linux facilitando la gestión de Wine usando una especie de contenedores llamados Botellas, el paquete se puede descargar como .AppImage y FlatPak que permite manejar correctamente la instalación de dependencias y compatibilidad del Software, además mantiene las aplicaciones seguras dentro del contenedor.

13.12.8 macOS en Linux

cumple una función similar a la de Wine con los programas de Windows, solo que no tiene ningún complejo en definirse como un emulador. Lo que hace es actuar como un traductor que permite ejecutar los programas de macOS usando los recursos de Linux. El nombre Darling (Querida) es la primera parte del nombre del núcleo de macOS (Darwin) y las primeras 3 letras de Linux. Supongo que la G final es para construir una palabra fácil de memorizar.

Hay que decir que a los desarrolladores de Darling la cosa les resulta más fácil que a los de Wine. No tienen que hacer ingeniería inversa ni reinventar nada dado que se basan en las partes de Darwin que están bajo licencias abiertas. El propio Darling se distribuye bajo la licencia GPL.

Iniciando Darling el programa no tiene interfaz gráfica. Lo ponemos en marcha desde la terminal con el comando:

```
$ darling shell
```

Al escribirlo, Darling creará un directorio raíz virtual o se conectará con uno existente. Además cargará los módulos del núcleo y construirá el sistema de archivos virtual donde ejecutaremos los programas.

Desde la línea de comandos podemos acceder a dos tipos de sistemas de archivos: el tradicional de macOS que incluye los directorios de nivel superior como */Applications*, */Users* y */System* entre otros. Por otro lado, el del sistema operativo anfitrión lo hallamos en una partición denominada */Volumes/SystemRoot*

Podemos verificar el núcleo con el siguiente comando:

```
$ uname
```

y averiguar la versión de macOS con:

```
$ sw_vers
```

salimos de la terminal con

```
$ exit
```

y apagamos el contenedor con:

```
$ darling shutdown
```

Instalación de Programas Si estás usando Linux en arranque dual con macOS y quieres ejecutar alguno de los programas que tienes instalado en la partición de Mac, puedes hacerlo con el comando:

```
$ /Volumes/SystemRoot/run/media/usuario/Macintosh HD
/Applications/nombre_app.app)
```

Muchos programas para macOS se distribuyen en formato .dmg. Para instalarlos en Darling hacemos:

```
Darling [~]$ hdiutil attach Downloads/aplicación.dmg /Vol-
umes/aplicacion
Darling [~]$ cp -r /Volumes/aplicación/aplicación.app /Ap-
plications/
```

En el caso de aplicaciones almacenadas en archivos comprimidos, lo descomprimimos y copiamos en la carpeta */Applications*. Lo mismo con aplicaciones previamente descargadas de la tienda de aplicaciones.

Por último nos quedan las aplicaciones *.pkg*, el formato de paquete nativo de macOS. Este formato implica ejecutar Scripts durante la instalación. Para poder usarlos debemos hacer:

```
Darling [~]$ installer -pkg aplicación.pkg -target /
```

Podemos desinstalar los programas con:

```
Darling [~]$ uninstaller nombre_del_paquete
```

Debemos entender que si bien Darling funciona muy bien con aplicaciones para la línea de comandos, solo tiene funcionalidades muy limitadas para las que necesitan una interfaz gráfica.

Instalación de Darling Si utilizas Debian o derivados, la instalación de Darling no tiene mayor problema. Solo tienes que escribir los comandos:

```
# apt install gdebi
# gdebi darling-dkms_X.X.X.testing_amd64.deb
# gdebi darling_X.X.X.testing_amd64.deb
```

reemplaza las X por el número de versión de los paquetes que descargarás o bien se puede descargar los archivos fuentes del proyecto para su compilación e instalación.

13.12.9 Debian GNU/Linux User Repository

En el 2021 se lanzó el repositorio DUR (Debian User Repository) que se posiciona como un análogo del repositorio AUR (Arch User Repository) para GNU/Linux Debian, permitiendo a los desarrolladores de terceros distribuir sus paquetes sin incluirlos en los principales repositorios de la distribución. Al igual que con el AUR, los metadatos y las instrucciones de construcción del paquete en el DUR se definen utilizando el formato PKGBUILD.

13.12.10 Debian Janitor Paquetes Desde Repositorio Git

Debian Janitor es un sistema automatizado que permite tomar fuentes de repositorios Git y generar paquetes Debian. El objetivo general es reducir el esfuerzo manual que se necesita para mantener un paquete, lo que permite a los usuarios probar versiones posteriores más nuevas de los paquetes sin la participación del responsable de mantenimiento.

14 Apéndice A: Software Libre y Propietario

Con el constante aumento de la comercialización de equipos de cómputo y/o comunicación (teléfonos inteligentes, tabletas, computadoras portátiles y de escritorio, etc.) y su relativo bajo costo, estos equipos se han convertido en objetos omnipresentes en nuestra vida diaria, ya que estos permiten realizar un creciente número de actividades cotidianas de miles de millones de usuarios.

Dichos equipos de cómputo y/o comunicación por sí solos tienen poca utilidad, pero su uso en conjunción con el Software adecuado forman un dúo que nos ha permitido tener los avances de los que actualmente disfrutamos. El Software -sistema operativo y los programas de aplicaciones- son los que realmente generan las soluciones al interactuar uno o más paquetes informáticos con los datos del usuario. También, es común que al comprar un equipo de cómputo y/o comunicación, en el costo total, se integre el del sistema operativo, aplicaciones ofimáticas y de antivirus, sean estos usados por el usuario o no y en la mayoría de los casos no es posible solicitar que no sean incluidos en el costo del equipo.

Por otro lado, el Software comercial suele quedar obsoleto muy rápido, ya que constantemente se le agregan nuevas funcionalidades al mismo y estas en general son vendidas como versiones independientes de la adquirida originalmente. Esto obliga al usuario -si quiere hacer uso de ellas- a comprar las nuevas versiones del Software para satisfacer sus crecientes necesidades informáticas y la obsolescencia programada.

Por lo anterior y dada la creciente complejidad de los paquetes de cómputo y el alto costo de desarrollo de aplicaciones innovadoras, en muchos casos, el costo total del Software que comúnmente los usuarios instalan -y que no necesariamente usan las capacidades avanzadas del programa, por las cuales el Software tiene un alto costo comercial- en sus equipos, suele ser más caro que el propio equipo en el que se ejecutan.

Hoy en día los usuarios disponemos de dos grandes opciones para adquirir el Software necesario para que nuestros equipos funcionen, a saber:

- Por un lado, podemos emplear programas comerciales (Software propietario), de los cuales no somos dueños del Software, sólo concesionarios al adquirir una licencia de uso del Software y nos proporcionan un instalable del programa adquirido. La licencia respectiva es en la gran mayoría de los casos muy restrictiva, ya que restringe su uso a un solo

equipo y/o usuario simultáneamente.

- Por otro lado, existe el Software libre[302], desarrollado por usuarios y para usuarios que, entre otras cosas, comparten los códigos fuente, el programa ejecutable y dan libertades para estudiar, adaptar y redistribuir a quien así lo requiera el programa y todos sus derivados.

Sobre la Obsolescencia Programada Es un conjunto de estrategias deliberadas destinadas a asegurarse que la versión actual de un determinado producto quedará desfasada o inservible en un plazo de tiempo predeterminado. De esta manera, los fabricantes se aseguran que los consumidores se verán obligados a reemplazarlo aunque funcione adecuadamente.

La obsolescencia puede lograrse mediante la introducción de un modelo con características superiores o diseñando intencionadamente un producto para que deje de funcionar correctamente en un plazo determinado. En cualquiera de los dos casos, se espera que los consumidores opten por el nuevo producto de la misma marca. Muchas veces la obsolescencia no es sobre el propio producto sino aplicando restricciones al producto de un competidor con la ayuda de una tercera empresa.

Tipos de Obsolescencia Programada Podemos dividir la obsolescencia programada en 4 tipos:

1- Establecimiento artificial del plazo de duración: Los productos se fabrican con piezas cuya duración tienen una vida útil limitada cuando, si se usaran otras de calidad superior ese plazo se extendería.

2- Actualizaciones de Software: Los desarrolladores de Software sacan nuevas versiones de sus aplicaciones que en un momento determinado dejan de ser compatibles con dispositivos antiguos. En muchos casos se ha podido comprobar que esa incompatibilidad es absolutamente artificial ya que al «engañar» al Software este funcionaba sin problemas.

[302] A veces también se han usado términos como FOSS y FLOSS. Ambas cosas son similares, ya que FOSS (Free and Open Source Software) traducido como "Software de código abierto" y FLOSS (Free/Libre and Open Source Software) "Software libre y de código abierto". Según quienes adoptan estos términos, lo hacen por tener una imparcialidad entre la carga filosófica del Software libre y el aspecto técnico y/o las ventajas que brinda este modelo de desarrollo. Richard Stallman nos invita a no usarlas y no se trata de un ad hóminem. Stallman y el proyecto GNU nos aconsejan que hablemos siempre de Software libre y aquí no cabe imparcialidad.

3- Obsolescencia percibida: Esta es una táctica psicológica, se trata de convencer al consumidor mediante publicidad y el uso de influenciadores de que el producto que se tiene actualmente está viejo y que se necesita uno nuevo. Como por ejemplo: ¿cuantos megapíxeles necesitas en tu teléfono para sacar una buena foto de tu mascota?

4- Trabas a la reparación: En el caso de los teléfonos por ejemplo, lo de impedir sacar la batería (con la excusa de hacer los teléfonos más delgados) es una forma de obligar a los consumidores a recurrir a los servicios oficiales y a disuadirlos de reemplazarlas por sustitutos más económicos. Otras tácticas son la utilización de piezas no estándar o que necesitan herramientas específicas para la reparación. Muchas veces se suele restringir el acceso a estas piezas o hacer una reducida producción de las mismas para aumentar artificialmente el costo.

Ejemplos de Obsolescencia Programada

- iPhone cada vez más lentos: La Justicia francesa comprobó que actualizaciones de Software hacían cada vez más lento el rendimiento de los modelos más viejos. La empresa le echó la culpa a las baterías, pero pagó una compensación de decenas de millones de dólares. Además rebajó los precios de sus baterías de repuesto para que los teléfonos fueran más rápidos con el nuevo Software y se comprometió a hacer más en el futuro para garantizar que los teléfonos no volvieran a ser más lentos. Con la salida de un nuevo modelo de teléfono cada año, seguro que hay algo de obsolescencia planificada en alguna parte.

- Impresoras: Esto es algo que todos conocemos. Muchas veces nos encontramos con impresoras a precio rebajado, pero al momento de tener que comprar un cartucho de tinta nos encontramos con que este tiene un precio igual o superior a comprar una nueva. Además, se ponen restricciones a la recarga o al uso de cartuchos alternativos. Hubo denuncias de que algunos modelos dejaban de funcionar a partir de cierta cantidad de páginas impresas o cierto tiempo desde la primera impresión.

- Certificados de seguridad: Por ejemplo, el pasado 30 de septiembre de 2021 caduco otro certificado de autentificación (CA de DST Root CA X3 de Let's Encrypt) que ayudaba a validar la conexión en internet a

los dispositivos que no fueron actualizados a otro certificado más actual -en la mayoría de los casos por no ser del interés económico de sus creadores-. Esto ocasionó que millones de dispositivos (teléfonos inteligentes, Smart TV, tabletas, computadoras portátiles y de escritorio, etc.) con algunos años de ser creados y perfectamente funcionales dejarán de conectarse a internet de un día para otro, forzando a sus dueños a desechar el dispositivo por carecer del servicio de internet en las aplicaciones instaladas.

- Cambio de la versión del sistema operativo: En el caso del sistema operativo Windows 10 a 11, la solicitud de requisitos mínimos de Hardware es para muchos equipos excesivo, ya que se estima que dejará fuera en su actualización a casi todos los equipos con más de 4 años de antigüedad por no contar por ejemplo con el Chip TPM 2.0 o GPU compatible con DirectX 12, siendo perfectamente funcionales con la versión actual del sistema operativo. Si bien Windows 10 seguirá con soporte hasta 2025, los usuarios que deseen tener las nuevas características del sistema operativo tendrán que cambiar de equipo.

14.1 Software Propietario

No existe consenso sobre el término a utilizar para referirse al opuesto del Software libre. La expresión «Software propietario (Proprietary Software)» (véase [16]), en la lengua anglosajona, "Proprietary" significa «poseído o controlado privadamente (Privately Owned and Controlled)», que destaca la manutención de la reserva de derechos sobre el uso, modificación o redistribución del Software. Inicialmente utilizado, pero con el inconveniente de que la acepción proviene de una traducción literal del inglés, no correspondiendo su uso como adjetivo en el español, de manera que puede ser considerado como un barbarismo.

El término "propietario" en español resultaría inadecuado, pues significa que «tiene derecho de propiedad sobre una cosa», por lo que no podría calificarse de "propietario" al Software, porque éste no tiene propiedad sobre nada (es decir, no es dueño de nada) y además, no podría serlo (porque es una cosa y no una persona). Así mismo, la expresión "Software propietario" podría ser interpretada como: "Software sujeto a propiedad" (derechos o titularidad) y su opuesto, el Software libre, también está sujeto al derecho de autor. Otra interpretación es que contrariamente al uso popular del término, se puede

afirmar que "todo Software es propietario", por lo que la forma correcta de referirse al Software con restricciones de uso, estudio, copia o mejora es la de Software privativo, según esta interpretación el término "propietario" podría aplicarse tanto para Software libre como Software privativo, ya que la diferencia entre uno y otro está en que el dueño del Software privativo lo licencia como propiedad privada y el de Software libre como propiedad social.

Con la intención de corregir el defecto de la expresión "Software propietario" aparece el llamado "Software con propietario", sin embargo se argumenta contra el término "con propietario" y justamente su similitud con Proprietary en inglés, que sólo haría referencia a un aspecto del Software que no es libre, manteniendo una de las principales críticas a éste (de "Software sujeto a derechos" o "propiedad"). Adicionalmente, si "propietario" se refiere al titular de los derechos de autor -y está claro que no se puede referir al usuario, en tanto éste es simplemente un cesionario-, no resuelve la contradicción: todo el Software libre tiene también titulares de derechos de autor.

La expresión Software no libre (en inglés Non-Free Software) es usado por la FSF para agrupar todo el Software que no es libre, es decir, incluye al llamado en inglés "Semi-Free Software" (Software semilibre) y al "Propietary Software". Asimismo, es frecuentemente utilizado para referirse al Software que no cumple con las Directrices de Software libre de Debian GNU/Linux, las cuales siguen la misma idea básica de libertad en el Software, propugnada por la FSF y sobre las cuales está basada la definición de código abierto de la Open Source Initiative.

Adicionalmente el Software de código cerrado nace como antónimo de Software de código abierto y por lo tanto se centra más en el aspecto de ausencia de acceso al código que en los derechos sobre el mismo, éste se refiere sólo a la ausencia de una sola libertad por lo que su uso debe enfocarse sólo a este tipo de Software y aunque siempre signifique que es un Software que no es libre, no tiene que ser Software de código cerrado.

La expresión Software privado es usada por la relación entre los conceptos de tener y ser privado. Este término sería inadecuado debido a que, en una de sus acepciones, la palabra "privado" se entiende como antónimo de "público", es decir, que «no es de propiedad pública o estatal, sino que pertenece a particulares», provocando que esta categoría se interpretará como no referente al Estado, lo que produciría la exclusión del Software no libre generado por el aparato estatal. Además, el "Software público" se asocia generalmente con Software de dominio público.

14.2 Software Libre

La definición de Software libre (véase [21], [22], [14], [15], [13] y [17]) estipula los criterios que se tienen que cumplir para que un programa sea considerado libre. De vez en cuando se modifica esta definición para clarificarla o para resolver problemas sobre cuestiones delicadas. «Software libre» significa que el Software respeta la libertad de los usuarios y la comunidad. En términos generales, los usuarios tienen la libertad de copiar, distribuir, estudiar, modificar y mejorar el Software. Con estas libertades, los usuarios -tanto individualmente como en forma colectiva- controlan el programa y lo que hace.

Cuando los usuarios no controlan el programa, el programa controla a los usuarios. Los programadores controlan el programa y a través del programa, controlan a los usuarios. Un programa que no es libre, llamado «privativo o propietario», es considerado por muchos como un instrumento de poder injusto.

El Software libre es la denominación del Software que respeta la libertad de todos los usuarios que adquirieron el producto y por tanto, una vez obtenido el mismo puede ser usado, copiado, estudiado, modificado y redistribuido libremente de varias formas. Según la Free Software Foundation (véase [21]), el Software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir y estudiar el mismo, e incluso modificar el Software y distribuirlo modificado.

Un programa es Software libre si los usuarios tienen las cuatro libertades esenciales:

0. La libertad de usar el programa, con cualquier propósito.

1. La libertad de estudiar cómo funciona el programa y modificarlo, adaptándolo a tus necesidades.

2. La libertad de distribuir copias del programa, con lo cual puedes ayudar a tu prójimo.

3. La libertad de mejorar el programa y hacer públicas esas mejoras a los demás, de modo que toda la comunidad se beneficie.

Un programa es Software libre si los usuarios tienen todas esas libertades. Por tanto, el usuario debe ser libre de redistribuir copias, tanto con o sin modificaciones, ya sea gratuitamente o cobrando una tarifa por la distribución,

a cualquiera en cualquier parte. El ser libre de hacer estas cosas significa, entre otras cosas, que no tiene que pedir ni pagar el permiso.

También debe tener la libertad de hacer modificaciones y usarlas en privado para su propio trabajo o pasatiempo, sin siquiera mencionar que existen. Si publica sus cambios, no debe estar obligado a notificarlo a nadie en particular, ni de ninguna manera.

La libertad de ejecutar el programa significa que cualquier tipo de persona u organización es libre de usarlo en cualquier tipo de sistema de computación, para cualquier tipo de trabajo y finalidad, sin que exista obligación alguna de comunicarlo al programador ni a ninguna otra entidad específica. En esta libertad, lo que importa es el propósito de los usuarios, no el de los programadores. El usuario es libre de ejecutar el programa para alcanzar sus propósitos y si lo distribuye a otra persona, también esa persona será libre de ejecutarlo para lo que necesite; nadie tiene derecho a imponer sus propios objetivos.

La libertad de redistribuir copias debe incluir las formas binarias o ejecutables del programa, así como el código fuente, tanto para las versiones modificadas como para las que no lo estén. Distribuir programas en forma de ejecutables es necesario para que los sistemas operativos libres se puedan instalar fácilmente. Resulta aceptable si no existe un modo de producir un formato binario o ejecutable para un programa específico, dado que algunos lenguajes no incorporan esa característica, pero debe tener la libertad de redistribuir dichos formatos si encontrara o programara una forma de hacerlo.

Para que se de la libertad que se menciona en los puntos 1 y 3 de realizar cambios y publicar las versiones modificadas tenga sentido, el usuario debe tener acceso al código fuente del programa. Por consiguiente, el acceso al código fuente es una condición necesaria para el Software libre. El «código fuente» compilado no es código fuente real y no cuenta como código fuente.

La libertad 1 incluye la libertad de usar su versión modificada en lugar de la original. Si el programa se entrega con un producto diseñado para ejecutar versiones modificadas de terceros, pero rechaza ejecutar las suyas, una práctica conocida como «tivoización» o «arranque seguro» [«Lockdown»] la libertad 1 se convierte más en una ficción teórica que en una libertad práctica, esto no es suficiente, en otras palabras, estos binarios no son Software libre, incluso si se compilaron desde un código fuente que es libre.

Una manera importante de modificar el programa es agregándole subrutinas y módulos libres ya disponibles. Si la licencia del programa especifica que no se pueden añadir módulos que ya existen y que están bajo una licencia

apropiada, por ejemplo si requiere que usted sea el titular de los derechos de autor del código que desea añadir, entonces se trata de una licencia demasiado restrictiva como para considerarla libre.

La libertad 3 incluye la libertad de publicar sus versiones modificadas como Software libre. Una licencia libre también puede permitir otras formas de publicarlas; en otras palabras, no tiene que ser una licencia de Copyleft. No obstante, una licencia que requiera que las versiones modificadas no sean libres, no se puede considerar libre.

«Software libre» no significa que «no es comercial». Un programa libre debe estar disponible para el uso comercial, la programación comercial y la distribución comercial. La programación comercial de Software libre ya no es inusual; el Software libre comercial es muy importante, ejemplo de ello es la empresa RedHat (ahora propiedad de IBM). Puede haber pagado dinero para obtener copias de Software libre, o puede haber obtenido copias sin costo. Pero sin tener en cuenta cómo obtuvo sus copias, siempre tiene la libertad de copiar y modificar el Software, incluso de vender copias.

El término Software no libre se emplea para referirse al Software distribuido bajo una licencia de Software más restrictiva que no garantiza estas cuatro libertades. Las leyes de la propiedad intelectual reservan la mayoría de los derechos de modificación, duplicación y redistribución para el dueño del Copyright; el Software dispuesto bajo una licencia de Software libre rescinde específicamente la mayoría de estos derechos reservados.

Los manuales de Software deben ser libres por las mismas razones que el Software debe ser libre y porque de hecho los manuales son parte del Software. También tiene sentido aplicar los mismos argumentos a otros tipos de obras de uso práctico, es decir, obras que incorporen conocimiento útil, tal como publicaciones educativas y de referencia. Wikipedia es el ejemplo más conocido.

La lista de proyectos de este tipo es realmente impresionante, algunos han conseguido un uso y alta calidad, por ejemplo el compilador GCC, el Kernel de Linux y el sistema operativo Debian GNU/Linux y Android. Mientras que otros proyectos han caído en el olvido, pero de la gran mayoría se tiene copia del código fuente que permitiría a quienes estén interesados en dicho proyecto poder reusarlo y en su caso ampliarlo.

La característica más importante que aparece típicamente en un proyecto de este tipo, es que un conjunto de personas separadas a gran distancia, sean capaces, a través de la Web, de los E-mail y de foros de aunar sus esfuerzos para crear, mejorar y distribuir un producto, de forma que todos

ellos se benefician unos de otros. Evidentemente, la mayor parte del peso recae en los desarrolladores, pero también es necesaria una difusión para que los usuarios documenten, encuentren errores, hagan foros de discusión, etc.

Si bien, el Software libre no es más seguro (en el sentido de invulnerable) que el propietario, la diferencia estriba en que el código fuente en el Software libre está disponible para todos y cualquiera puede aportar una solución, y por lo general al poco tiempo de detectarse una vulnerabilidad (a veces en cuestión de horas) se puede disponer de una solución para la misma. Además, al tener acceso al código fuente se puede detectar fácilmente si alguien introdujo código malicioso a una determinada aplicación.

¿Por qué se Interesan los Autores, Alumnos y Profesores Universitarios en el Software Libre? La ventaja principal es porque bajo el Software libre subyace la idea de compartir conocimiento y favorecer la existencia de nuevas ideas[303]; y ¿qué es investigar y enseñar?, sino crear conocimiento y procurar que los alumnos aprendan e incluso vayan más allá de lo aprendido. Se comparte la idea, que el espíritu del Software libre es similar al que debería reinar en las instituciones educativas:

- Porque así no se condiciona a los estudiantes a usar siempre lo mismo.
- No se fomenta la piratería en los estudiantes y se evita pagar licencias que no son necesarias al existir alternativas gratuitas.
- Es mucho más seguro ya que el Software libre es público y se puede ver qué hace exactamente sin recelos.
- Se ofrece libertad de elección a los estudiantes y profesores al no limitarlos a usar una solución determinada, ampliando sus opciones y permitiendo un mayor aprendizaje.

Concretando estas ideas, profesores e investigadores necesitan herramientas para la investigación y docencia y estas deben tener una calidad mínima y ser fácilmente distribuibles entre los alumnos. En muchos casos las compañías desarrolladoras y distribuidoras de programas de cómputo no han

[303] ¿Por qué el Software creado con dinero de los impuestos no se publica como Software Libre?

¡El código pagado por los ciudadanos debería estar disponible para los ciudadanos y el mismo gobierno!

sabido ofrecer sus productos con la flexibilidad adecuada para las labores docentes o, en otros casos, los productos desarrollados no tienen la calidad esperada.

El Software libre es aún joven, pese a las decenas de miles de proyectos actuales -en los que se trabaja constantemente en mejorar la parte computacional de los algoritmos involucrados en el proyecto, haciendo y puliendo interfaces gráficas, generando ayuda en línea así como la documentación necesaria para que usuarios noveles y avanzados usen la mayor cantidad de opciones programadas- existen muchas otras necesidades profesionales y de investigación que requieren el desarrollo innovador de programas de cómputo para automatizarlas y hacerlas eficientes. Esto queda plasmado en las decenas de proyectos que a diario son registrados en las páginas especializadas en busca de difusión y apoyo para su proyecto.

En los últimos años, muchos proyectos han pasado de ser simples programas en línea de comandos a complejas aplicaciones multiplataforma -se ejecutan en distintos sistemas operativos como son Windows, Linux, Unix, Mac OS, Android- con ambientes gráficos multimedia que en muchos casos han superado a sus contrapartes comerciales -por ejemplo los navegadores Web-. Para muestra de este maravilloso avance, tomemos el proyecto del sistema operativo Android, que actualmente se ejecuta en millones de equipos -como celulares, tabletas, electrodomésticos, etc.- y en los cuales se pueden descargar miles de aplicaciones y está soportado por una gran cantidad de usuarios y empresas comerciales como Google, IBM y últimamente Microsoft -que años atrás era acérrima enemiga del Software libre-.

El Software libre ha logrado desplazar a muchos de sus competidores por sus múltiples bondades y bajo costo de desarrollo -es el caso de Windows Phone que fue reemplazado por Android de Google-, al reusar miles de aplicaciones ya existentes que usan Software libre y permitir desarrollar otro tanto de aplicaciones bajo una plataforma que se ejecuta en los más diversos procesadores. Además, el uso de Software libre y su posibilidad de ampliarlo y/o especializarlo según sea necesario, ha permitido crear de forma cada vez más rápida y confiable; para poner a disposición de un gran público programas de uso común, así como especializado que satisfagan las nuevas necesidades de los usuarios.

Software Libre en Ciencia y Educación Algunos puntos y reflexiones sobre porqué se considera que es interesante el Software libre en Ciencia y

Educación son:

- Accesible a todo el mundo aunque no sea rentable su desarrollo: El Software libre entre sus libertades permite que se pueda ejecutar por terceros, copiarlo, distribuirlo y estudiarlo/modificarlo. Eso hace que si en ciencia se usa Software libre el acceso a esos programas no suponga una barrera (se puede distribuir y ejecutar).

- Muchos de los desarrollos en el campo de la accesibilidad se realizan en universidades y son distribuidos como Software libre: Aunque no sea rentable muchas veces el desarrollo de herramientas de accesibilidad (no sea algo monetizable) en la universidad se consigue escapar a esa la lógica capitalista de solamente invertir en lo que pueda ofrecer beneficio económico.

- Transparencia: En ciencia es importante ver las costuras para comprobar si es verdad lo que se afirma, tener acceso al código fuente del Software empleado permite poder estudiarlo por si realiza algún cálculo mal.

- Propicia el espíritu crítico: Si no tienes acceso a las revistas o el acceso es privativo para los bolsillos de mucha gente no puedes comprobar la información. Se nos pide que seamos críticos con la información que se nos da del mundo científico pero no podemos entrenar el espíritu crítico sin acceso al conocimiento libre.

- Caramelos con droga en la puerta del colegio: Muchas empresas buscan introducir en los colegios, institutos y universidad su Software. Ofrecer un programa que permita trabajar y genere una dependencia quedando los datos muchas veces en las nubes (ordenadores de otras personas). Un ejemplo es Microsoft con Office 365. Dando cuentas gratuitas durante un tiempo para que se use su Software. Otro ejemplo podría ser Unity3D en vez de por ejemplo Godot.

- Especificaciones de protocolos abiertas VS cerradas: Gracias a que los protocolos TCP/IP, HTTP, POP, SNMP, DHCP, etc. son abiertos es posible construir herramientas por cualquier con conocimientos de programación. Con protocolos cerrados solamente quienes tuvieran acceso a las especificaciones podrían desarrollar y conocer cómo funcionan.

- Uso de estándares: Existiendo un estándar para documentos ofimáticos (procesador de textos, hoja de cálculo, presentaciones, etc.) algunas empresas como Microsoft se empeñan en ir con su propio formato y estándar en vez de sumarse a que sea más sencillo ir a una y que el usuario pueda optar por que herramientas usar para editar o trabajar con documentos ofimáticos.

- Software libre para la Ciencia ciudadana: Un ejemplo en el que es importante la colaboración ciudadana es el cambio climático y la defensa medioambiental del territorio. Desde imvec.tech usan herramientas de Software libre para medición y monitorización de contaminación.

Software Libre: Beneficios Más Allá de la Informática El uso de las tecnologías de código abierto supone cultivar el conocimiento y la puesta en valor de la libertad individual, lo personal y lo privado, todo ello sin menospreciar lo público y la construcción de una sociedad. El movimiento del Software libre, con Linux a la cabeza, ha capitaneado durante décadas planteamientos para un cambio en el modo de producción: el abaratamiento de costes empresariales para grandes y pequeños, el trabajo en línea, el desarrollo de Software y Hardware a pequeña escala, el replanteamiento del negocio informático, el sistema de normas éticas que rigen los grupos, la documentación abierta, etc.

Todo ello derivado de una simple idea: la libertad. Ha sido la clave que ha llevado a todo este movimiento hacia una autonomía y motivación que pocas veces se ve en otros sectores. El Open Source está lleno de alternativas con la libertad como pilar y consecuencia filosófica, de hecho muchos Forks (derivaciones) de proyectos aparecen cuando la disputa sobre la misma toma relevancia. Esta manera de hacer las cosas debería trasladarse directamente a la sociedad promoviendo esos valores para evitar caer en un mundo despótico que toma fuerza a pasos agigantados.

No estaría mal promover la comprensión de las licencias libres. Leer y entender una licencia GPL, MIT o BSD es infinitamente más sencillo y rápido que hacerlo con otras, siendo unas normas fáciles de cumplir porque encajan con un modelo ético y práctico comprensible por muchos.

Podríamos decir que GPL, BSD y MIT son las Constituciones que vertebran todo el movimiento del Software Libre, cosechando derechos de uso y logros como el ahorro o la legítima copia privada. Lo mismo podría ocurrir con las licencias Creative Commons para cierto contenido, un arma poderosa

en el sector divulgativo que está poco extendida por desconocimiento y el Status Quo de la propiedad intelectual.

La cooperación libre y voluntaria supuso un éxito hasta ahora pero está siendo amenazada por la dinámica actual de la Web, donde la centralización de las principales plataformas supone estar bajo el yugo de normas que rastrean con lupa y cambian sus términos con demasiada frecuencia. Ya ni digamos cuando eso se junta con el ansia de monetización: si quieres dinero más te vale no cabrear a los anunciantes o no tener un Strike por uso de contenido que podría ser reclamado.

Los claroscuros de este sistema hace que los creadores cada vez hagan menos de forma libre y altruista, y ello podría verse potenciado por las búsquedas con inteligencia artificial, lo que apunta a un empobrecimiento del contenido cultural fresco y renovador. Las polémicas con GitHub Copilot o ChatGPT sobre el entrenamiento de las IAs hace sobrevolar una vez más la cuestión del Copyright y el uso legítimo de los resultados brindados por éstas, lo que también condiciona la creación.

La libertad de expresión, de uso, de creación, de modificación ... esas cuestiones llevan irremediablemente a una libertad de pensamiento y acción, a una adaptación creativa que puede ser la motivación para romper moldes en todos los aspectos. El código abierto y sus licencias son, por tanto, un beneficio personal y social mucho más grande que el simple hecho de usar Linux o Software libre. El contenido despreocupado, que no prioriza el dinero y el posicionamiento/visualizaciones, se vuelve esencial para el inconformismo.

14.3 Seguridad del Software

Si bien, el Software Libre no es más seguro (en el sentido de invulnerable) que el propietario, la diferencia puede estribar en que el código fuente en el Software libre está disponible para todos y cualquiera puede aportar una solución y por lo general al poco tiempo de detectarse una vulnerabilidad (a veces en cuestión de horas) se puede disponer de una solución para la misma. Además, al tener acceso al código fuente se puede detectar si alguien introdujo código malicioso a una determinada aplicación.

Pero de todos es sabido, que los usuarios de Software de código abierto, como por ejemplo los que de manera habitual trabajan con equipos comandados por sistemas Linux, por regla general se sienten orgullosos de la seguridad que estos programas aportan con respecto a los sistemas cerrados propios de otras firmas, dígase Microsoft Windows o Mac de Apple.

¿Es Seguro el Software Libre? En primer lugar definiremos el concepto de "seguridad" como salvaguarda de las propiedades básicas de la información. Entre las características que debe cumplir para ser seguro, encontramos la integridad, es decir, que sólo los usuarios autorizados pueden crear y modificar los componentes del sistema, la confidencialidad, sólo estos usuarios pueden acceder a esos componentes, la disponibilidad, que todos los componentes estén a disposición de los usuarios siempre que lo deseen y el "no repudio", o lo que es lo mismo, la aceptación de un protocolo de comunicación entre el servidor y un cliente, por ejemplo, mediante certificados digitales.

Entre las diferencias de seguridad entre un Software Libre y el Software Propietario, podemos destacar:

- Seguridad en el Software Propietario: En el caso de tener "agujeros de seguridad", puede que no nos demos cuenta y que no podamos repararlos. Existe una dependencia del fabricante, retrasándose así cualquier reparación y la falsa creencia de que es más seguro por ser oscuro (la seguridad por oscuridad determina los fallos de seguridad no parcheados en cada producto).

- Seguridad en el Software Libre: Por su carácter público y su crecimiento progresivo, se van añadiendo funciones y se nos permite detectar más fácilmente los agujeros de seguridad para poder corregirlos. Los problemas tardan mucho menos en ser resueltos por el apoyo que tiene de los Hackers y una gran comunidad de desarrolladores y al ser un Software de código libre, cualquier empresa puede aportar soporte técnico.

Sin embargo esta es una pregunta sobre la que los expertos al día de hoy, tras muchos años de discusiones, siguen sin ponerse de acuerdo. ¿Es más seguro el Software de código abierto que los programas cerrados, o viceversa? Lo cierto es que, en términos generales, ambos bandos tienen sus razones con las que defender sus argumentos. Por un lado, los usuarios de las aplicaciones y sistemas de código abierto, defienden que, al estar el código fuente disponible a los ojos de todo el mundo, es mucho más fácil localizar posibles agujeros de seguridad y vulnerabilidades que pongan en peligro los datos de los usuarios.

Por otro lado, aquellos que consideran que los sistemas cerrados son más seguros en este sentido, afirman que al tener acceso tan solo los expertos al código fuente de sus aplicaciones, es más complicado que se produzcan

filtraciones o inserciones de Software malicioso en este tipo de sistemas. Hay que tener en cuenta que, por ejemplo, Google premia a las personas que descubren fallos de seguridad en su Software como Chrome, aunque no es el único gigante de la tecnología en utilizar estas tácticas.

De hecho muchas empresas están gastando miles de millones de dólares y/o euros en hacer que sus propuestas sean lo más seguras posible, argumentando que la seguridad de sus proyectos es una de sus prioridades, todo con el fin de intentar frenar que los atacantes vulneren sus sistemas. Por otro lado, otros aseguran que cuando el código fuente es público, más ojos están disponibles para detectar posibles vulnerabilidades o errores en dicho código, por lo que siempre será más rápido y sencillo poner soluciones con el fin de ganar en seguridad.

Sea como sea, en cualquiera de los dos casos, lo que ha quedado más que demostrado es que la seguridad no está garantizada en ningún momento, ya sean propuestas de código abierto, o no. Pero también es cierto que lo que se procura es que los riesgos de ser atacados se reduzcan en medida de lo posible. Los sistemas Linux son considerados desde hace mucho tiempo como un sistema operativo seguro, en buena parte debido a las ventajas que ofrece su diseño. Dado que su código está abierto, son muchas las personas que incorporan mejoras de las que el resto de usuarios de Linux se benefician, a diferencia de las propuestas de Windows o MacOS, donde estas correcciones generalmente se limitan a las que detectan Microsoft y Apple.

No obstante, en nuestra defensa del Software libre, diremos que su código abierto permite que los errores sean encontrados y solucionados con mayor rapidez, por lo que determinamos que es el Software más recomendable.

En general, puede afirmarse que el Software libre es más seguro, ya que debido a su carácter abierto y distribuido, un gran número de programadores y personas expertas pueden estar atentas al código fuente -especialmente en los grandes proyectos-, lo cual permite hacer auditorías con objeto de detectar errores y puertas traseras (Backdoor, en inglés) que pongan en riesgo nuestros datos.

Así, los grandes programas y proyectos de Software libre, con una extensa comunidad de desarrollo y usuarios que lo respalden, presentan niveles muy altos de seguridad, un alto grado de protección y una rápida respuesta a posibles vulnerabilidades.

14.4 Tipos de Licencias

Tanto la Open Source Initiative como la Free Software Foundation mantienen en sus páginas Web (véase [21], [22], y [17]) listados oficiales de las licencias de Software libre que aprueban.

Una licencia es aquella autorización formal con carácter contractual que un autor de un Software da a un interesado para ejercer "actos de explotación legales". Pueden existir tantas licencias como acuerdos concretos se den entre el autor y el licenciatario. Desde el punto de vista del Software libre, existen distintas variantes del concepto o grupos de licencias:

Licencias GPL Una de las más utilizadas es la Licencia Pública General de GNU (GNU GPL). El autor conserva los derechos de autoría (Copyright), y permite la redistribución y modificación bajo términos diseñados para asegurarse de que todas las versiones modificadas del Software permanecen bajo los términos más restrictivos de la propia GNU GPL. Esto hace que sea imposible crear un producto con partes no licenciadas GPL, el conjunto tiene que ser GPL.

En la práctica, esto hace que las licencias de Software libre se dividan en dos grandes grupos, aquellas que pueden ser mezcladas con código licenciado bajo GNU GPL (y que inevitablemente desaparecerán en el proceso, al ser el código resultante licenciado bajo GNU GPL) y las que no lo permiten al incluir mayores u otros requisitos que no contemplan ni admiten la GNU GPL y que por lo tanto no pueden ser enlazadas ni mezcladas con código gobernado por la licencia GNU GPL.

GPL Versión 1 la versión 1 de GNU GPL, fue presentada el 25 de febrero de 1989, impidió lo que eran las dos principales formas con las que los distribuidores de Software restringían las libertades definidas por el Software libre. El primer problema fue que los distribuidores publicaban únicamente los archivos binarios, funcionales y ejecutables, pero no entendibles o modificables por humanos. Para prevenir esto, la GPLv1 estableció que cualquier proveedor de Software libre además de distribuir el archivo binario debía liberar a su vez código fuente entendible y que pudiera ser modificado por el ser humano bajo la misma licencia (secciones 3a y 3b de la licencia).

El segundo problema era que los distribuidores podían añadir restricciones adicionales, añadiendo restricciones a la licencia o mediante la combinación del Software con otro que tuviera otras restricciones en su distribución. Si

esto se hacía, entonces la unión de los dos conjuntos de restricciones sería aplicada al trabajo combinado, entonces podrían añadirse restricciones inaceptables. Para prevenir esto, GPLv1 obligaba a que las versiones modificadas en su conjunto, tuvieran que ser distribuidas bajo los términos GPLv1 (secciones 2b y 4 de la licencia). Por lo tanto, el Software distribuido bajo GPLv1 puede ser combinado con Software bajo términos más permisivos y no con Software con licencias más restrictivas, lo que entraría en conflicto con el requisito de que todo Software tiene que ser distribuido bajo los términos de la GPLv1.

GPL Versión 2 según Richard Stallman, el mayor cambio en GPLv2 fue la cláusula "Liberty or Death" («libertad o muerte»). Esta sección dice que si alguien impone restricciones que le prohíben distribuir código GPL de tal forma que influya en las libertades de los usuarios (por ejemplo, si una ley impone que esa persona únicamente pueda distribuir el Software en binario), esa persona no puede distribuir Software GPL. La esperanza es que esto hará que sea menos tentador para las empresas el recurrir a las amenazas de patentes para exigir una remuneración de los desarrolladores de Software libre.

En 1991 se hizo evidente que una licencia menos restrictiva sería estratégicamente útil para la biblioteca C y para las bibliotecas de Software que esencialmente hacían el trabajo que llevaban a cabo otras bibliotecas comerciales ya existentes. Cuando la versión 2 de GPL fue liberada en junio de 1991, una segunda licencia Library General Public License fue introducida al mismo tiempo y numerada con la versión 2 para denotar que ambas son complementarias. Los números de versiones divergieron en 1999 cuando la versión 2.1 de LGPL fue liberada, esta fue renombrada como GNU Lesser General Public License para reflejar su lugar en esta filosofía.

GPL Versión 3 A finales de 2005, la Free Software Foundation (FSF) anunció estar trabajando en la versión 3 de la GPL (GPLv3). El 16 de enero de 2006, el primer borrador de GPLv3 fue publicado y se inició la consulta pública. La consulta pública se planeó originalmente para durar de nueve a quince meses, pero finalmente se extendió a dieciocho meses, durante los cuales se publicaron cuatro borradores. La GPLv3 oficial fue liberada por la FSF el 29 de junio de 2007.

Según Stallman los cambios más importantes se produjeron en el campo

de las patentes de Software, la compatibilidad de licencias de Software libre, la definición de código fuente y restricciones a las modificaciones de Hardware. Otros cambios están relacionados con la internacionalización, cómo son manejadas las violaciones de licencias y cómo los permisos adicionales pueden ser concedidos por el titular de los derechos de autor. También añade disposiciones para quitar al DRM su valor legal, por lo que es posible romper el DRM (Digital Rights Management) en el Software de GPL sin romper leyes como la DMCA (Digital Millennium Copyright Act).

GPLv2 vs GPL v3 GPLv3 contiene la intención básica de GPLv2 y es una licencia de código abierto con un Copyleft estricto. Sin embargo, el idioma del texto de la licencia fue fuertemente modificado y es mucho más completo en respuesta a los cambios técnicos, legales y al intercambio internacional de licencias.

La nueva versión de la licencia contiene una serie de cláusulas que abordan preguntas que no fueron o fueron cubiertas de manera insuficiente en la versión 2 de la GPL. Las nuevas regulaciones más importantes son las siguientes:

- GPLv3 contiene normas de compatibilidad que hacen que sea más fácil combinar el código GPL con el código que se publicó bajo diferentes licencias. Esto se refiere en particular al código bajo la licencia de Apache v. 2.0.

- Se insertaron normas sobre gestión de derechos digitales para evitar que el Software GPL se modifique a voluntad, ya que los usuarios recurrieron a las disposiciones legales para protegerse mediante medidas técnicas de protección (como la DMCA o la directiva sobre derechos de autor).

- La licencia GPLv3 contiene una licencia de patente explícita, según la cual las personas que licencian un programa bajo licencia GPL otorgan derechos de autor y patentes, en la medida en que esto sea necesario para utilizar el código que ellos otorgan. Por lo tanto, no se concede una licencia de patente completa. Además, la nueva cláusula de patente intenta proteger al usuario de las consecuencias de los acuerdos entre los titulares de patentes y los licenciatarios de la licencia pública general que solo benefician a algunos de los licenciatarios (correspondientes al

acuerdo Microsoft / Novell). Los licenciatarios deben garantizar que todos los usuarios disfrutan de tales ventajas (licencia de patente o liberación de reclamos) o que nadie puede beneficiarse de ellos.

- A diferencia de la GPLv2, la GPLv3 establece claramente que no es necesario divulgar el código fuente en un uso ASP (Application Service Provider) de los programas GPL, siempre que no se envíe una copia del Software al cliente. Si el efecto Copyleft debe extenderse al uso de ASP, debe aplicarse la Licencia pública general de Affero, versión 3 (AGPL) que solo difiere de la GPLv3 en esta consideración.

Licencias Estilo BSD Llamadas así porque se utilizan en gran cantidad de Software distribuido junto a los sistemas operativos BSD. El autor, bajo tales licencias, mantiene la protección de Copyright únicamente para la renuncia de garantía y para requerir la adecuada atribución de la autoría en trabajos derivados, pero permite la libre redistribución y modificación, incluso si dichos trabajos tienen propietario. Son muy permisivas, tanto que son fácilmente absorbidas al ser mezcladas con la licencia GNU GPL con quienes son compatibles. Puede argumentarse que esta licencia asegura "verdadero" Software libre, en el sentido que el usuario tiene libertad ilimitada con respecto al Software y que puede decidir incluso redistribuirlo como no libre.

Licencia Copyleft Hay que hacer constar que el titular de los derechos de autor (Copyright) de un Software bajo licencia Copyleft puede también realizar una versión modificada bajo su Copyright original y venderla bajo cualquier licencia que desee, además de distribuir la versión original como Software libre. Esta técnica ha sido usada como un modelo de negocio por una serie de empresas que realizan Software libre (por ejemplo MySQL); esta práctica no restringe ninguno de los derechos otorgados a los usuarios de la versión Copyleft.

Licencia estilo MIT Las licencias MIT son de las más permisivas, casi se consideran Software de dominio público. Lo único que requieren es incluir la licencia MIT para indicar que el Software incluye código con licencia MIT.

Licencia Apache License La licencia Apache trata de preservar los derechos de autor, incluir la licencia en el Software distribuido y una lista de

los cambios realizados. En modificaciones extensivas del Software original permite licenciar el Software bajo otra licencia sin incluir esas modificaciones en el código fuente.

Licencia Mozilla Public License MPL Esta licencia requiere que los archivos al ser distribuidos conserven la misma licencia original pero pueden ser usados junto con archivos con otra licencia, al contrario de la licencia GPL que requiere que todo el código usado junto con código GPL sea licenciado como código GPL. También en caso de hacer modificaciones extensivas permite distribuirlas bajo diferentes términos y sin incluir el código fuente en las modificaciones.

Licencia Código de Dominio Público Es un código que no está sujeto a derechos de autor que puede utilizarse sin restricciones.

Licencia Creative Commons Las licencias de Creative Commons son más utilizadas para cualquier creación digital que para el Software, entendiendo como creación digital desde fotos, artículos en blogs, música, vídeos, este trabajo, etc. Hay varios tipos de licencias de Creative Commons diferenciando entre permitir modificaciones a la obra original, solicitando crédito de la creación o permitiendo un uso comercial de la obra.

Licencias de Código Abierto Las licencias de código abierto son un intermedio entre las licencias privativas y las licencias de Software libre. Las licencias de código abierto permiten el acceso al código fuente pero no todas se consideran licencias de Software libre al no otorgar otros derechos que se requieren para considerar un Software como Software libre como el derecho al uso o con cualquier propósito, modificación y distribución.

Dado el éxito del Software libre como modelo de desarrollo de Software algunas empresas cuyo Software era privativo pueden decidir hacerlo de código abierto con la intención de suplir algunas carencias de Software privativo pero sin perder ciertos derechos que son la fuente de sus ingresos como la venta de licencias.

Las expresiones «Software libre» y «código abierto» se refieren casi al mismo conjunto de programas. No obstante, dicen cosas muy diferentes acerca de dichos programas, basándose en valores diferentes. El movimiento del Software libre defiende la libertad de los usuarios de ordenadores, en un

movimiento en pro de la libertad y la justicia. Por contra, la idea del código abierto valora principalmente las ventajas prácticas y no defiende principios. Esta es la razón por la que gran parte de la comunidad de Software libre está en desacuerdo con el movimiento del código abierto y nosotros no empleamos esta expresión en este texto.

Licencia Microsoft Public License La Microsoft Public License es una licencia de código abierto que permite la distribución del Software bajo la misma licencia y la modificación para un uso privado. Tiene restricciones en cuanto a las marcas registradas.

En caso de distribuir el Software de forma compilada o en forma de objeto binario no se exige proporcionar los derechos de acceso al código fuente del Software compilado o en forma de objeto binario. En este caso esta licencia no otorga más derechos de los que se reciben, pero si permite otorgar menos derechos al distribuir el Software (compilado o en forma de objeto binario).

Modelo de Desarrollo de Software Bazar y Catedral El tipo de licencia no determina qué Software es mejor o peor, si el privativo o el Software libre, la diferencia entre las licencias está en sus características éticas y legales. Aunque el modelo de desarrollo con una licencia de código abierto a la larga suele tener un mejor desarrollo y éxito que el Software privativo, más aún con un medio como internet que permite colaborar a cualquier persona independiente de donde esté ubicada en el mundo.

Comparación con el Software de Código Abierto Aunque en la práctica el Software de código abierto y el Software libre comparten muchas de sus licencias, la Free Software Foundation opina que el movimiento del Software de código abierto es filosóficamente diferente del movimiento del Software libre. Los defensores del término «código abierto (Open Source)», afirman que éste evita la ambigüedad del término en ese idioma que es «Free» en «Free Software».

Mucha gente reconoce el beneficio cualitativo del proceso de desarrollo de Software cuando los desarrolladores pueden usar, modificar y redistribuir el código fuente de un programa. El movimiento del Software libre hace especial énfasis en los aspectos morales o éticos del Software, viendo la excelencia técnica como un producto secundario de su estándar ético. El movimiento de código abierto ve la excelencia técnica como el objetivo prioritario, siendo

el compartir el código fuente un medio para dicho fin. Por dicho motivo, la FSF se distancia tanto del movimiento de código abierto como del término «código abierto (Open Source)».

Puesto que la «iniciativa de Software libre Open Source Initiative (OSI)» sólo aprueba las licencias que se ajustan a la «definición de código abierto (Open Source Definition)», la mayoría de la gente lo interpreta como un esquema de distribución, e intercambia libremente "código abierto" con "Software libre". Aún cuando existen importantes diferencias filosóficas entre ambos términos, especialmente en términos de las motivaciones para el desarrollo y el uso de tal Software, raramente suelen tener impacto en el proceso de colaboración.

Aunque el término "código abierto" elimina la ambigüedad de libertad frente a precio (en el caso del inglés), introduce una nueva: entre los programas que se ajustan a la definición de código abierto, que dan a los usuarios la libertad de mejorarlos y los programas que simplemente tienen el código fuente disponible, posiblemente con fuertes restricciones sobre el uso de dicho código fuente. Mucha gente cree que cualquier Software que tenga el código fuente disponible es de código abierto, puesto que lo pueden manipular, sin embargo, mucho de este Software no da a sus usuarios la libertad de distribuir sus modificaciones, restringe el uso comercial, o en general restringe los derechos de los usuarios.

14.4.1 Licencias Creative Commons

Las Licencias[304] Creative Commons (CC) de forma general no tienen una definición oficial, sin embargo, entre las muchas definiciones aceptadas están la de la UNESCO, la cual expresa la siguiente descripción:

Las Licencias Creative Commons (CC) son modelos de contratos que sirven para otorgar públicamente el derecho de utilizar una publicación protegida por los derechos de autor. Entre menos restricciones implique una licencia, mayores serán las posibilidades de utilizar y distribuir un contenido. Las Licencias CC permiten a cualquier usuario descargar, copiar, distribuir, traducir, reutilizar, adaptar y desarrollar su contenido sin costo alguno.

Sin embargo, en la Web oficial de la Organización Creative Commons se nos dice sobre las mismas lo siguiente:

[304] Las licencias de Creative Commons son más utilizadas para cualquier creación digital que para el Software, entendiendo como creación digital desde fotos, artículos en blogs, música, vídeos, este trabajo, etc.

Las Licencias Creative Commons (CC) brindan a todos, desde creadores individuales hasta grandes instituciones, una forma estandarizada de otorgar permiso al público para usar su trabajo creativo bajo la ley de derechos de autor. Desde la perspectiva del reutilizador, la presencia de una licencia Creative Commons sobre una obra protegida por derechos de autor responde a la pregunta: ¿Qué puedo hacer con esta obra?.

Las «Licencias Creative Commons» que hoy en día pertenecen a la organización mundial Creative Commons[305], y buscan regularizar y mantener, de forma equilibrada y satisfactoria, todo lo relacionado con el derecho de utilizar una publicación protegida por los derechos de autor a nivel mundial, han logrado un buen trabajo, sin duda alguna. Y seguramente en el tiempo, se irán adaptando a las nuevas realidades sociales y tecnológicas para poder seguir manteniendo de forma armónica las posibilidades de utilizar y distribuir cualquier contenido libre y abierto sobre la Internet, y más allá.

¿Cuáles son y cómo funcionan o para qué se usan? Las 7 distintas Licencias Creative Commons son las siguientes:

CC BY Esta licencia permite a los reutilizadores distribuir, remezclar, adaptar y desarrollar el material en cualquier medio o formato, siempre que se otorgue la atribución al creador. La licencia permite el uso comercial.

CC BY-SA Esta licencia permite a los reutilizadores distribuir, remezclar, adaptar y desarrollar el material en cualquier medio o formato, siempre que se otorgue la atribución al creador. La licencia permite el uso comercial. Si remezcla, adapta o construye sobre el material, debe licenciar el material modificado bajo términos idénticos.

CC BY-NC Esta licencia permite a los reutilizadores distribuir, remezclar, adaptar y desarrollar el material en cualquier medio o formato, únicamente con fines no comerciales y siempre que se otorgue la atribución al creador.

[305] Una organización mundial sin ánimo de lucro que permite compartir y reutilizar la creatividad y el conocimiento mediante el suministro de herramientas legales gratuitas. Y cuyas herramientas legales (licencias) ayudan a quienes quieren fomentar la reutilización de sus obras ofreciéndolas para su uso bajo términos generosos y estandarizados; a quienes quieren hacer usos creativos de las obras; y a quienes quieren beneficiarse de esta simbiosis.

CC BY-NC-SA Esta licencia permite a los reutilizadores distribuir, remezclar, adaptar y desarrollar el material en cualquier medio o formato, únicamente con fines no comerciales y siempre que se otorgue la atribución al creador. Si remezcla, adapta o construye sobre el material, debe licenciar el material modificado bajo términos idénticos.

CC BY-ND Esta licencia permite a los reutilizadores copiar y distribuir el material en cualquier medio o formato únicamente en forma no adaptada, y siempre y cuando se otorgue la atribución al creador. La licencia permite el uso comercial.

CC BY-NC-ND Esta licencia permite a los reutilizadores copiar y distribuir el material en cualquier medio o formato únicamente en forma no adaptada, únicamente con fines no comerciales y siempre que se otorgue la atribución al creador.

CC0 (CC Cero) Esta licencia es una herramienta de dedicación pública que permite a los creadores renunciar a sus derechos de autor y poner sus obras en el dominio público mundial. CC0 permite a los reutilizadores distribuir, remezclar, adaptar y desarrollar el material en cualquier medio o formato, sin condiciones.

14.4.2 Nuevas Licencias para Responder a Nuevas Necesidades

El mundo de la tecnología avanza mucho más rápido que las leyes y estas tienen que esforzarse para alcanzarlo. En el caso del Software libre y de código abierto, tanto la Free Software Foundation como la Open Source Initiative (los organismos encargados de regular las diferentes licencias) enfrentan periódicamente el problema de cómo mantener sus principios y al mismo tiempo evitar que alguien se aproveche indebidamente.

En el último tiempo, la Open Source Initiative le dio el sello de aprobación a otras nuevas licencias para propósitos específicos.

Nuevas Licencias de Código Abierto

- Cryptographic Autonomy License version 1.0 (CAL-1.0)

Fue creada en el 2019 por el equipo del proyecto de código abierto Holochain, esta licencia fue desarrollada para ser utilizada con aplicaciones criptográficas distribuidas. El inconveniente con las licencias tradicionales es que no obligaba a compartir los datos. Esto podría perjudicar el funcionamiento de toda la red. Por eso la CAL también incluye la obligación de proporcionar a terceros los permisos y materiales necesarios para utilizar y modificar el Software de forma independiente sin que ese tercero tenga una pérdida de datos o capacidad.

- Open Hardware Licence (OHL)

De la mano de la Organización Europea para la Investigación Nuclear (CERN) llegó esta licencia con tres variantes enfocadas en la posibilidad de compartir libremente tanto Hardware como Software.

Hay que hacer una aclaración. La OSI fue creada en principio pensando en el Software por lo que no tiene mecanismos para la aprobación de licencias de Hardware. Pero, como la propuesta del CERN se refiere a ambos rubros, esto posibilitó la aprobación:

- CERN-OHL-S es una licencia fuertemente recíproca: El que utilice un diseño bajo esta licencia deberá poner a disposición las fuentes de sus modificaciones y agregados bajo la misma licencia.

- CERN-OHL-W es una licencia débilmente recíproca: Sólo obliga a distribuir las fuentes de la parte del diseño que fue puesta originalmente bajo ella. No así los agregados y modificaciones.

- CERN-OHL-P es una licencia permisiva: Permite a la gente tomar un proyecto, relicenciarlo y utilizarlo sin ninguna obligación de distribuir las fuentes.

Hay que decir que la gente del CERN parece haber encontrado la solución a un problema que viene afectando a algunos proyectos de código abierto. Una gran empresa utiliza ese proyecto para comercializar servicios y no solo hace ningún aporte al proyecto original (ya sea con código o dando apoyo financiero) si no que también compite en el mismo mercado.

Post Open Zero-Cost en el año 2024 se desarrolla la licencia «Post Open Zero-Cost» con la cual busca abordar los desafíos surgidos en la interacción entre desarrolladores de código abierto y empresas comerciales, especialmente en lo que respecta a la compensación justa por el uso comercial del código.

La característica distintiva de la licencia «Post-Open» en comparación con las licencias abiertas existentes, como la GPL, es la introducción de un componente contractual que puede ser rescindido en caso de violación de los términos. Esta licencia ofrece dos tipos de acuerdos contractuales: gratuitos y de pago. El acuerdo de pago permite negociar derechos adicionales para la distribución comercial de productos o modificaciones sin requerir su divulgación pública.

Las situaciones que podrían llevar a la terminación del acuerdo contractual incluyen: violación de los términos de la licencia; reclamaciones por infracción de patentes; imposición de condiciones adicionales (como sanciones en contratos con clientes por divulgación de información sensible); cambios sujetos a leyes de control de exportaciones; ocultamiento de información sobre vulnerabilidades; y uso del código para entrenamiento de modelos de aprendizaje automático bajo términos no permitidos. Las relaciones contractuales no se rescinden de inmediato, sino que se notifica la infracción y se otorgan 60 días para corregirla antes de la rescisión efectiva del acuerdo.

Uno de los problemas que la nueva licencia busca abordar está relacionado con las limitaciones de la GPL, la cual se centra en otorgar derechos sin la capacidad de revocarlos, lo que permite a las empresas encontrar formas de eludir sus requisitos, especialmente en lo que respecta al acceso al código fuente. Estas lagunas son utilizadas para restringir la disponibilidad del código subyacente en productos comerciales mediante la imposición de términos contractuales adicionales con los usuarios finales.

Un claro ejemplo es el de RHEL, la cual los clientes firman un acuerdo con Red Hat que limita la redistribución del código fuente al imponer condiciones sobre la coincidencia de las copias instaladas y compradas de RHEL. Esto coloca a los usuarios en la disyuntiva entre su libertad para disponer del software y mantener su estatus de cliente de Red Hat. Aunque la GPL permite la distribución de parches que solucionan vulnerabilidades en el código de RHEL, esto podría interpretarse como una violación del acuerdo con Red Hat y podría resultar en la terminación de los servicios por parte de la empresa.

14.5 Implicaciones Económico-Políticas del Software Libre

Una vez que un producto de Software libre ha empezado a circular, rápidamente está disponible a un costo muy bajo. Al mismo tiempo, su utilidad no decrece. El Software, en general, podría ser considerado un bien de uso inagotable, tomando en cuenta que su costo marginal es pequeñísimo y que no es un bien sujeto a rivalidad -la posesión del bien por un agente económico no impide que otro lo posea-.

14.5.1 Software Libre y la Piratería

Puesto que el Software libre permite el libre uso, modificación y redistribución, a menudo encuentra un hogar entre usuarios para los cuales el coste del Software no libre es a veces prohibitivo, o como alternativa a la piratería. También es sencillo modificarlo localmente, lo que permite que sean posibles los esfuerzos de traducción a idiomas que no son necesariamente rentables comercialmente, además:

- Porque así no se condiciona a los usuarios a usar siempre lo mismo.
- Porque así no se fomenta la piratería en los usuarios al no pagar licencias.
- Porque así no se obliga a usar una solución concreta y se ofrece libertad de elección a los usuarios.
- Porque es mucho más seguro ya que el Software libre es público y se puede ver qué hace exactamente sin recelos.

La mayoría del Software libre se produce por equipos internacionales que cooperan a través de la libre asociación. Los equipos están típicamente compuestos por individuos con una amplia variedad de motivaciones y pueden provenir tanto del sector privado, del sector voluntario o del sector público. En los últimos años se ha visto un incremento notable de grandes corporativos (como IBM, Microsoft, Intel, Google, Samsung, Red Hat, etc.) que han dedicado una creciente cantidad de recursos humanos y computacionales para desarrollar Software libre, ya que esto apoya a sus propios negocios.

14.5.2 ¿Cuánto Cuesta el Software Libre?

En esta sección intentaremos dar una idea de cuál es el costo del desarrollo del Software Libre, por supuesto que no se tratará más que de una conjetura aproximada basada en las cifras proporcionadas por desarrolladores de Software comercial (al año 2020).

Gratis no Significa Gratuito Supongamos que todos los recursos humanos participantes en el desarrollo de un proyecto de Software libre lo hagan de forma voluntaria. De todas formas tenemos lo que los contables llaman «Costo de oportunidad» esto es, los ingresos que podrían haber generado esas personas si hubieran dedicado el tiempo y los conocimientos invertidos en el proyecto a uno en el que les pagaran. Así, el calcular el costo promedio por hora que cobra un programador, por la cantidad de horas invertidas al proyecto, nos da un razonable costo mínimo. Lo mismo puede hacerse con los voluntarios dedicados a la difusión en las redes. El costo de una campaña de marketing digital puede estimarse fácilmente.

Muchos proyectos de código abierto como una distribución Linux, son construidos a partir de la integración de otros proyectos, por los que sus costos de desarrollo también deberían sumarse.

Por otra parte, necesitamos recursos físicos. Aún cuando los voluntarios trabajen desde su casa, siguen teniendo que comprar y mantener sus equipos, además de pagar la electricidad que los hace funcionar.

Bases para el Cálculo Hay muchos factores que determinan el costo de desarrollar una pieza de Software. En un extremo tenemos una aplicación simple que requiere muy poca interacción del usuario o procesamiento del lado del servidor. Tal es el caso de un cliente de escritorio para redes sociales. Por el otro sistemas operativos que deben operar en múltiples plataformas realizando múltiples tareas (por ejemplo Debían que aspira a ser el sistema operativo universal). Sin embargo, el costo de una aplicación simple puede elevarse debido a que tiene múltiples pantallas diferentes. Por ejemplo un juego desarrollado con HTML5 y Javascript.

Los dos aspectos claves son la cantidad de horas de trabajo necesarias y las tecnologías involucradas. Para una aplicación de escritorio como un procesador de textos con las prestaciones habituales, optimizado para un determinado escritorio Linux, se estima que se tendría que contar con al menos el equivalente a 42,000 euros en trabajo voluntario. Un gestor de contenidos

para comercio electrónico con seguimiento de pedidos e integración con las principales plataformas de pago implicaría desembolsar unos 210,000 euros o su equivalente en trabajo voluntario.

Tomando en cuenta que este cálculo incluye lo que costó el desarrollo de las bibliotecas y otros proyectos libres y de código abierto incluidos, pero no los gastos que efectivamente deben desembolsarse en efectivo como la compra de equipos, Software de seguridad y desarrollo y el pago de electricidad e internet.

Por otro lado, el proceso de medición de costes del Software es un factor realmente importante en el análisis de un proyecto. Hay distintos métodos de estimación de costes de desarrollo de Software (también conocido como métrica del Software). La gran mayoría de estos métodos se basan en la medición del número de Líneas de Código que contiene el desarrollo (se excluyen comentarios y líneas en blanco de los fuentes).

Desarrollo de Fedora 9 La Linux Foundation ha calculado que costaría desarrollar el código de la distribución Fedora 9 que fue puesta a disposición del público el 13 de mayo de 2008, en el informe citado "Estimating the Total Development Cost of a Linux Distribution" se calcula que Fedora 9 tiene un valor de 10.8 mil millones de dólares y que el coste únicamente del Kernel (2.6.25 con 8,396,250 líneas de código) tendría un valor de 1.4 mil millones de dólares.

Esta distribución tiene unas 205 millones de líneas de código, el proyecto debería ser desarrollado por 1000 - 5000 desarrolladores (el trabajo invertido por una única persona desarrollándolo se alargaría durante unos 60.000 años) y esa estimación no va muy desencaminada ya que en los 2 últimos años del desarrollo de esa versión contribuyeron unos 3,200 desarrolladores aunque el número de trabajadores en la historia de la distribución es mucho mayor.

¿Qué pasa con GNU/Linux? en el año 2015 (las estadísticas más actuales que conseguimos) la Linux Foundation analizó el costo de desarrollo del núcleo. Combinando el aporte de los recursos humanos (voluntarios y de pago) y los desembolsos necesarios, la cuenta sumó 476,767,860,000.13 euros.

Todos sabemos que el hecho de tener desarrolladores asalariados no garantiza necesariamente Software de calidad. Pero, tener desarrolladores que pueden dedicar toda su atención a un proyecto en lugar de hacerlo en sus horas libres si lo hace. Lamentablemente, por el momento el único modo de

lograr eso es obtener el apoyo de corporaciones (Intel, Google, IBM, AMD, Sun Microsystems, Dell, Lenovo, Asus, HP, SGI, Oracle, RedHat, etc.) que solo lo hacen con los proyectos que son de su interés como el Kernel de Linux, hay que notar que para el Kernel de Linux un porcentaje importante (más del 10 %) lo hacen programadores independientes.

Costes Recordemos que la segunda de las cuatro libertades de un programa para ser Software libre es:

- Libre redistribución

y esta puede ser a través de un pago o sin costo. Es por ello que existen distintas empresas, organizaciones y usuarios que pueden apoyar a los usuarios finales en el desarrollo y soporte de algún programa de Software libre o una distribución personalizada de Linux por un costo determinado.

Mucha gente, en especial ejecutivos de empresas, se acercan a Linux bajo la promesa de que es una solución de bajo costo -muchos piensan que incluso es gratis-. Pero la realidad es que detrás de Linux y los programas de Software libre (y aquí la traducción correcta de la palabra inglesa, Free es libre, no gratis) pueden llevar una serie de costos ocultos que deben ser considerados al momento de decidir si se implementa una solución propietaria o una libre.

Los costos ocultos aparecen cuando se intenta instalar y capacitar en el uso de algún Software y se necesita la ayuda de un informático, al que se tiene que pagar, o alguna empresa quiere personalizar la interfaz de un programa y necesita la ayuda de un programador, que también tienen un coste, por lo que finalmente el comentario suele ser "el Software libre no es barato".

El primer punto a considerar al evaluar ambas alternativas es el costo de la licencia. Los productos de Software libre no suelen tener un costo de licencia asociado, que sí existe en los programas propietarios. De hecho es allí en donde los fabricantes de Software recargan sus costos de investigación y desarrollo, de producción e incluso sus ganancias. En este primer punto el ganador claro es el Software libre y es lo que los adeptos de este esquema publicitan: "su compañía puede ahorrar miles de dólares al año usando Software libre".

El segundo punto a considerar es el costo de instalación, configuración y capacitación. Dependiendo de su complejidad, algunos productos comerciales no contemplan costos extra por este concepto y otros -como Windows, por ejemplo- son tan populares que se puede encontrar numerosas opciones

de instalación -a través de empresas o profesionales- donde escoger en el mercado. A veces en el Software libre la configuración puede implicar recompilar el producto con algunas opciones particulares, algo que sólo pueden realizar técnicos con un nivel adecuado de conocimientos y que puede que no sean tan fáciles de encontrar. Aquí por lo general la ventaja va hacia el Software comercial.

Una vez instalado el Software, toca realizar actualizaciones de mantenimiento. Si bien es cierto lo que algunos de los fanáticos del Software libre dicen, que nadie lo obliga a mejorar el Software con que cuenta, la realidad -especialmente en lo que a seguridad se refiere- obliga a las empresas a mantener su Software actualizado. Aún así, los costos de actualizar Software libre suelen ser significativamente más bajos que los de productos comerciales y suelen ser menos exigentes con el Hardware necesario para ejecutarlos. La mayoría de las veces la ventaja es para el Software libre, pero hay que evaluar ya que varía dependiendo de cada caso.

Por último hay algunas casas que desarrollan productos de Software libre -dan el código y permiten que cualquiera lo modifique o reutilice- pero fijan contratos con cargos mensuales o anuales para mantenimiento del mismo, algo que se parece mucho a un cobro por licencia, por lo que hay que estar seguro de conocer todas las condiciones cuando una empresa nos ofrece una solución propia y la califica como Software libre.

Además de estas consideraciones hay una que debe sumarse eventualmente a esta evaluación: el costo de migrar a otra solución. En Software libre suelen usarse estándares abiertos para almacenar los datos, lo que facilita las migraciones. En cambio muchas soluciones propietarias suelen tener formatos propietarios que pueden dejar "amarrados" los datos de la empresa a una aplicación específica.

Sólo después de evaluar estos aspectos del Software, que pueden tener implicaciones importantes en el presupuesto, es que un CIO (Chief Information Officer) puede decir si una solución de Software libre le conviene más a una empresa o no, algo que va más allá de que la aplicación sea gratis o no.

14.5.3 La Nube y el Código Abierto

Desde hace años se han creado nuevos desafíos para el código abierto que plantea la nube, un término que para el usuario promedio puede significar cosas diferentes, pero que para la empresa se resume en servicios. Y es que los beneficios económicos que genera el mero Software de código abierto no

son comparables a los que se obtienen cuando se ofrece ese mismo Software a través de servicios, más allá -pero incluyendo- del soporte.

Este hecho diferencial lleva tiempo provocando fricciones entre desarrolladores y proveedores y hay quien adelantó incluso el fin del modelo de desarrollo del código abierto tal y como lo conocemos. ¿Quién tiene la razón?, ¿es para tanto la situación?.

En sus exposiciones, representantes de compañías y proyectos de código abierto muy populares en el ámbito empresarial, explican el supuesto perjuicio que les ocasiona el uso que los grandes proveedores de servicios en la nube hacen del Software que ellos desarrollan y cómo algunos han considerado y aplicado un enfoque más cerrado para sus productos con el fin de evitar lo que denominan como expolio. Hay declaraciones que merecen ser rescatadas para dotar de contexto a la discusión:

- El papel que juega el código abierto en la creación de oportunidades comerciales ha cambiado, durante muchos años les permitimos que las empresas de servicios tomaran lo que se ha desarrollado y ganasen dinero con ello.

- Empresas como Amazon Web Services, Azure de Microsoft, etc. Han ganado cientos de millones de dólares ofreciendo a sus clientes servicios basados en Software libre sin contribuir tanto a la comunidad de código abierto que construye y mantiene ese proyecto. Es imposible saber exactamente de cuánto dinero estamos hablando, pero es cierto que los proveedores de la nube se benefician del trabajo de los desarrolladores de código abierto que no emplean.

- Hay un mito ampliamente instalado en el mundo de código abierto que dice que los proyectos son impulsados por una comunidad de contribuyentes, pero en realidad, los desarrolladores pagados contribuyen con la mayor parte del código en la mayoría de los proyectos de código abierto modernos.

En resumen, todas estas voces se quejan de dos cosas: los grandes beneficios que obtienen los proveedores de servicios en la nube con su Software sin retribuirles en consecuencia, y la falta de colaboración manteniendo los productos con los que lucran. Sin embargo, no nos engañemos, el quid de la cuestión está principalmente en el dinero: la opinión generalizada de la

comunidad es que el Software de código abierto nunca fue pensado para que las empresas de servicios en la nube lo tomasen y lo vendieran.

Por otro lado, si es posible bifurcar un proyecto libre que se cierra, ¿no hubiese sido mejor colaborar con él antes y haber evitado el cierre? Si no se invierte y se mantiene con salud aquello que da beneficios, puede terminar por desaparecer. A medio camino entre el depredador y el parásito: así es como ven muchas desarrolladoras de código abierto a los proveedores de servicios en la nube.

Vale la pena retomar ahora la frase «el Software de código abierto nunca fue pensado para que las empresas de servicios en la nube lo tomasen y lo vendieran». ¿Para qué fue pensado el código abierto entonces? No hay ninguna licencia de código abierto o Software libre reconocida por la Free Software Foundation o la Open Source Initiative que prohíba hacer negocio con el Software. Lo que prohíben es la discriminación en la capacidad y alcance de su uso en función de la parte, se trate de un individuo o de la mayor multinacional imaginable.

¿Cuál es la solución a un embrollo de tamaña envergadura? Lo único claro es que no es una cuestión de blancos y negros y las consideraciones son demasiadas como para seguir ahondando: empresas que cotizan en bolsa quieren más dinero de otras compañías -que también cotizan en bolsa y por mucho más-, que además del Software ponen la infraestructura sobre la que distribuyen sus ofertas y que tienen la capacidad de clonar tu producto en un abrir y cerrar de ojos, no solo porque tienen el capital, sino porque tienen la experiencia necesaria tras contribuir técnica, pero también en muchos casos, económicamente, durante largo tiempo.

Pese a ello, esta situación está alterando el paradigma actual, en el que el modelo de desarrollo del código abierto se ha impuesto como impulsor de la innovación en el sector empresarial y ya hay quien habla de que nos acercamos al fin, o al principio del fin de la Era del Open Source, cuya preponderancia estaría sentenciada por la revolución de la nube, a la postre el mayor estímulo que haya tenido el código abierto hasta la fecha.

El futuro, pues, pasaría por el Shared Source Software, bajo el cual diferentes compañías con intereses alineados colaborarán en el desarrollo de proyectos concretos, pero limitando su explotación comercial a sí mismas. Todavía no estamos ahí, no obstante, y no parece tampoco que el relevo se vaya a dar en breve. De suceder, será muy llamativo: la muerte del código abierto por un éxito mal entendido.

14.5.4 El Código Abierto como Base de la Competitividad

En septiembre del 2021, se publicó un amplio y detallado informe llevado a cabo por el Fraunhofer ISI y por OpenForum Europe para la Comisión Europea, «The impact of open source software and hardware on technological independence, competitiveness and innovation in the EU economy», cuantifica la importancia económica del código abierto aplicado tanto al Software como al Hardware, su efecto en la contribución al producto interior bruto generado, la reducción en aspectos como el coste total de propiedad, dependencia del proveedor y autonomía digital; lanza una serie de recomendaciones específicas de políticas públicas destinadas a lograr un sector público digitalmente autónomo, una investigación y desarrollo abierto que fomente el crecimiento europeo y una industria más digitalizada y competitiva.

En las estimaciones del informe se apunta que las empresas europeas invirtieron alrededor de mil millones de euros en Software de código abierto en 2018, lo que resultó en un impacto en la economía europea de entre 65,000 y 95,000 millones de euros. El análisis estima una relación costo-beneficio superior a 1:4 y predice que un aumento del 10% de las contribuciones de a repositorios de código abierto sería susceptible de generar anualmente entre un 0.4% y un 0.6% adicional en el PIB, así como más de seiscientas nuevas empresas tecnológicas en la Unión Europea.

El análisis de las contribuciones a repositorios de Software de código abierto en la Unión Europea revela que el ecosistema tiene una naturaleza diferente frente al norteamericano, con un volumen de contribuciones que provienen sobre todo de empleados de compañías pequeñas o muy pequeñas, frente a un escenario en los Estados Unidos en el que predominan grandes compañías tecnológicas que se benefician en sus modelos de negocio de la gran cantidad y de la rápida mejora del Software disponible. En Europa, los contribuyentes individuales ascendieron a más de 260,000, lo que representa el 8% de los casi 3.1 millones de empleados de la UE en el sector del desarrollo de Software en 2018. En total, los más de 30 millones de desarrollos consolidados en repositorios en los estados miembros de la Unión Europea representan una inversión de personal equivalente a casi mil millones de euros, que han pasado a estar disponibles en el dominio público y que, por lo tanto, no tienen que ser desarrollados por otros actores.

Según el análisis, cuanto más pequeña es la empresa, mayor es la inversión relativa en Software de código abierto (las empresas con 50 empleados o menos asumieron casi la mitad de los desarrollos en la muestra de las com-

pañías más activas). Aunque más del 50% de los contribuyentes pertenecen a la industria tecnológica (el 8% del total de sus empleados participaron en estos desarrollos), también hubo participación significativa de empresas de consultoría, científicas, técnicas y, en menor medida, de distribuidores, minoristas y empresas del ámbito financiero.

¿Puede una filosofía de desarrollo como el código abierto, disponible para todo el mundo, llegar a convertirse en una fuente de ventajas diferenciales para el resto de los países, que se ha visto tradicionalmente muy superado en su relevancia en el entorno tecnológico por los gigantes tecnológicos de Estados Unidos o de China? El informe afirma que su uso puede llegar a incidir en gran medida en el desarrollo de una independencia tecnológica superior, de una mayor competitividad y de más innovación. Veremos si llegamos a ver en el resto del mundo políticas que incentiven el uso del código abierto como una variable estratégica clave para ello. La idea, capitalizar la tecnología de una forma más orientada al procomún y al desarrollo colaborativo, suena sin duda atractiva e interesante.

14.5.5 Software Libre en Empresas y Corporaciones

En esta sección exploraremos algunas de las claves por las cuales el Software libre está hoy en el punto de mira de todo tipo de empresas y grandes corporaciones (algunas de las cuales ayer eran sus acérrimos enemigos). Pero hay que empezar destacando que corporativos como Google, Amazon Web Services, Azure de Microsoft, Microsoft, IBM, entre otras, en los últimos años han ganado miles de millones de dólares ofreciendo a sus clientes servicios y/o productos basados en Software libre y con una queja recurrente por su pobre o nula contribución a la comunidad de código abierto que construyó y mantiene esos proyectos.

Si bien es imposible saber exactamente de cuánto dinero estamos hablando, pero es cierto que empresas y corporaciones se benefician diariamente del trabajo de los desarrolladores de código abierto que no emplean.

Grandes Equipos de Programadores GNU/Linux ha demostrado que los equipos de desarrolladores grandes, distribuidos, aunque desorganizados pueden crear Software viable.

Antes de la llegada de GNU/Linux, la mayoría del Software era desarrollado por pequeños equipos de programadores que trabajaban en estrecha coordinación entre sí. Ese era el enfoque recomendado por informáticos de hace

unos decenios, que advertían que añadir más programadores a un proyecto tendía a disminuir su eficiencia. Y estaban muy equivocados.

Desde el principio, el Kernel de Linux se desarrolló con un enfoque diferente, en el que programadores de todo el mundo, que en la mayoría de los casos no se conocían, escribieron e integraron el código de forma rápida y poco organizada. Gracias a la publicación temprana y frecuente, consiguieron que funcionara y hoy día es el Kernel más usado en informática en supercomputadoras y dispositivos móviles.

Pero actualmente hay un mito ampliamente instalado en el mundo de código abierto, que dice que los proyectos son impulsados por una comunidad de contribuyentes gratuitos, pero en realidad, los desarrolladores pagados contribuyen con la mayor parte del código en la mayoría de los proyectos de código abierto modernos de los cuales las corporaciones pueden sacar provecho. Claro ejemplo es el propio Kernel de Linux, en el cual una gran cantidad de desarrolladores actuales pertenecen o son subvencionados por empresas, fundaciones o corporaciones (actualmente cientos de ellas), como en el caso de Linus Torvalds que trabaja bajo los auspicios de la Fundación de Linux.

Reutilización de Software Parte de la razón por la que Linux se hizo muy popular entre los ingenieros de Software con relativa rapidez fue que Linux -y el Software libre en general- facilita la reutilización del código escrito por otras personas.

Hoy en día, la reutilización de Software de terceros es habitual, incluso entre los equipos de desarrollo cuyos productos no son de código libre. Es difícil imaginar la construcción de una aplicación hoy en día sin hacer uso de las bibliotecas de Software de origen, las API de terceros u otros recursos externos a su propio proyecto.

Es cierto que proyectos como GNU, que precedió al Kernel de Linux en siete años, promovían la reutilización de código antes de que apareciera el núcleo. Pero, podría decirse que Linux fue el proyecto que trajo las prácticas de codificación libre a la corriente que tanto parece interesar a ciertas grandes empresas, ayudando a crear el modelo de ingeniería de Software de componentes de Software modulares y reutilizables.

Gestión Actual del Código Fuente Linus Torvalds, que creó el núcleo de Linux cuando era estudiante en Helsinki, es el más famoso por ese trabajo.

Pero un hecho a menudo olvidado es que Torvalds es también el padre de Git, el masivamente popular gestor de código fuente libre.

Torvalds creó Git para ayudar a gestionar el código fuente de Linux. Si Linux no existiera, tampoco existiría Git. Tampoco existiría GitHub, ni GitLab, ni GitOps. Y, lo que es más importante, sin la idea de Software libre y colaborativo de Richard Stallman, tampoco existiría la cultura de intercambio y colaboración abierta que sostienen estas tecnologías.

Estrategias de Despliegue de Software "App Store" Apple puede atribuirse el mérito de haber lanzado la primera App Store, un lugar donde los desarrolladores pueden compartir aplicaciones y los usuarios pueden instalarlas fácilmente, utilizando un catálogo Online centralizado.

Pero al igual que con muchas cosas que ha hecho Apple, el concepto de App Store (que ahora es una estrategia de despliegue de Software apilado como servicio, especialmente pero no sólo en el ecosistema móvil) se parece mucho a lo que los desarrolladores de GNU/Linux estaban haciendo a través de los repositorios de Software mucho antes de que las tiendas de aplicaciones se convirtieran en algo común en el mundo del Software propietario, como también lo es la Tienda de Windows.

Los repositorios de Software en GNU/Linux hacen más o menos lo mismo que las tiendas de aplicaciones: Permiten a los usuarios seleccionar las aplicaciones que quieren de una lista centralizada y en línea, y luego instalarlas con unos pocos clics o bien órdenes de terminal (como el famoso *apt* de Debian).

Es cierto que empresas como Apple parecen tener el mérito de crear tiendas de aplicaciones muy fáciles de usar de hacer clic e ir, pero no es un invento de ellos. Y la historia del concepto de tienda de aplicaciones en general implica a más actores que sólo la comunidad de Apple. Aún así, creo que se podría argumentar con fuerza que, sin GNU/Linux y los repositorios de Software de GNU/Linux, las tiendas de aplicaciones tal y como las conocemos hoy no existirían.

Formatos Abiertos para Intercambio de Información Hay una gran variedad de tecnologías disponibles para producir y almacenar datos. Como son: hojas de cálculo, bases de datos, Software estadístico más específico y más. Esto genera una enorme diversidad de formatos, a veces esto es por decir lo menos caótico.

La ventaja de los archivos de formatos abiertos, es que permiten a los de-

sarrolladores producir varios paquetes de Software y servicios utilizando esos formatos. Esto entonces reduce al mínimo los obstáculos para la reutilización de la información que contienen.

El advenimiento del Software libre ha generado algunos de los formatos abiertos más usados para el intercambio de información, pero los entes generadores de información no siempre se adecuan a los niveles de apertura deseados, algunos de estos formatos abiertos son: XML, JSON, YAML, RDF, REBOL, PDF, CSV, ODF, OOXML, TXT, HTML, HDF.

Pero, incluso si la información se proporciona en formato electrónico, formato legible por máquina y en detalle, puede existir problemas relacionados con el formato del archivo en sí (principalmente el generado por los diversos sistemas operativos). Los formatos en los cuales la información es publicada -en otras palabras, la base digital en la cual la información es almacenada- puede ser "abierta" o "cerrada".

Un formato abierto es aquel donde las especificaciones del Software están disponibles para cualquier persona, de forma gratuita, así cualquiera puede usar dichas especificaciones en su propio Software sin ninguna limitación en su reutilización que fuere impuesta por derechos de propiedad intelectual.

Si el formato del archivo es "cerrado", esto puede ser debido a que el formato es propietario y sus especificaciones no están disponibles públicamente, o porque el formato es propietario y aunque las especificaciones se han hecho públicas, su reutilización es limitada. Si la información es liberada en un formato de archivo cerrado, esto puede causar grandes obstáculos para reutilizar la información codificada en él, forzando a aquellos que deseen usar la información a comprar Software innecesario.

El uso de formatos de archivo con propiedad, para el que la especificación no está disponible públicamente, puede crear dependencia de Software de terceros o de los titulares de licencias de los formatos de archivos. En el peor de los casos, esto puede significar que la información sólo se puede leer con cierto Software específico, que puede ser caro, o que puede quedar obsoleto.

La preferencia del término Gobierno de Datos Abiertos, es que la información se publicará en formatos de archivo abiertos, los cuales son de lectura mecánica y esto es una aportación más del Software libre.

Ciencia Abierta La ciencia abierta (Open Science) es el movimiento creciente para hacer que la ciencia sea abierta. La ciencia en sí misma se utilizó como un ejemplo principal de la eficacia del movimiento de código abierto, ci-

tando prácticas como la difusión abierta de información, métodos y revisión por pares de la literatura científica. Podría decirse que la ciencia abierta comenzó en el siglo XVII con el advenimiento de la revista científica y la práctica de repetir los experimentos presentados en los artículos académicos. Estas revistas se imprimirían y distribuirían en todo el mundo, a menudo supervisadas por sociedades científicas como la Royal Society.

¿Qué impulsó la necesidad de un movimiento de ciencia abierta? La Royal Society tenía el famoso lema "Nullius in verba", traducido de forma aproximada como "no tome la palabra de nadie". Esto encarnaba un principio general en la ciencia de que todas las teorías están abiertas a ser cuestionadas y los resultados declarados deben ser repetibles. De hecho, es una práctica generalizada que fue realizada por la sociedad en esos primeros años. En los últimos años esta práctica no ha sido tan común, con más y más ciencia confiando en elementos cerrados, lo que en última instancia conduce a errores que son más difíciles de detectar sin un intercambio completo de información: datos, métodos y publicaciones.

El movimiento de ciencia abierta afirma en términos generales que la ciencia debe realizarse de manera abierta y reproducible donde todos los componentes de la investigación estén abiertos. Muchas revistas permanecen estancadas en un formato en el que se imprimían físicamente, a pesar de que en la actualidad se distribuyen en gran medida en línea. A menudo, todavía utilizan archivos PDF como una forma de "papel electrónico" con publicaciones fijas, procesos cerrados de revisión por pares y poco o ningún acceso a los datos. Este fue sin duda el modo más eficiente de difundir el conocimiento científico antes de los albores de internet, pero ahora un número cada vez mayor lo considera lejos de ser el óptimo.

La ciencia abierta encarna una serie de aspectos, en el núcleo esto incluye acceso abierto, datos abiertos, código abierto y estándares abiertos que ofrecen una diseminación sin restricciones del discurso científico. Estas cosas permiten una ciencia reproducible al brindar acceso completo a los componentes principales de la investigación científica. Hay una serie de componentes adicionales que también se están explorando, como la revisión por pares abierta, donde los revisores de publicaciones científicas publican revisiones abiertamente con su nombre adjunto y la ciencia de libreta abierta donde las libretas (tradicionalmente cerradas) se publican abiertamente en línea a medida que se realiza la investigación.

¿Por qué la ciencia abierta es tan importante en la era digital? También existe una creciente comprensión de que, dado que la investigación científica

depende cada vez más del código informático para simulaciones, cálculos, análisis, visualización y procesamiento de datos en general, es importante tener acceso a este código tal como tradicionalmente ha sido importante mostrar (y derivar) cualquier nueva técnica matemática introducida para el análisis. Hay revistas como PLOS ONE y F1000 que exploran el significado de las publicaciones, ya sea que se deben congelar en el tiempo o se pueden actualizar. Los repositorios de datos también están ganando importancia a medida que las agencias de financiación requieren la publicación y preservación de los datos generados por la investigación financiada.

En esencia, la ciencia abierta se trata de volver a esos valores fundamentales inculcados por algunos de los primeros científicos de que no debemos confiar en la palabra de nadie, que es esencial que todos los elementos pertinentes a un descubrimiento pretendido se publiquen para que los resultados puedan repetirse y validarse. El movimiento de la ciencia abierta varía en el grado en que lo requiere, pero están surgiendo patrones. Se están estableciendo recomendaciones sobre licencias, como CC0 para datos, CC-BY para publicaciones, licencias compatibles con OSI para código fuente y formatos abiertos para datos. En última instancia, se trata de empoderar a todos para que participen en la ciencia, con internet como vehículo principal para la amplia difusión de este conocimiento.

Este movimiento está cambiando la forma en que se hace la ciencia, está recibiendo el respaldo de muchas agencias de financiamiento, ya que requieren planes de gestión de datos, planes de distribución de código fuente y una mayor validación de los resultados a través del acceso abierto a estos resultados para todos. Esto también mejora la transferencia de conocimientos de la academia a la industria, ya que se brinda acceso completo en el momento de la publicación o después de un período de embargo. El movimiento de la ciencia abierta se limita en gran medida a la investigación que está financiada por las agencias de financiación nacionales de todo el mundo y exige que todos los que financian la investigación tengan acceso total e igualitario a ella.

Open Hardware El concepto de Software libre también se permeó al Hardware. El término Open Hardware u Open Source Hardware, se refiere al Hardware cuyo diseño se hace públicamente disponible para que cualquiera pueda estudiarlo, modificarlo y distribuirlo, además de poder producir y vender Hardware basado en ese diseño. Tanto el Hardware como el Software

que lo habilita, siguen la filosofía del Software libre. Hoy en día, el término "hágalo usted mismo" (DIY por sus siglas en inglés) se está popularizando en el Hardware gracias a proyectos como Arduino que es una fuente abierta de prototipos electrónicos, una plataforma basada en Hardware flexible y fácil de utilizar que nació en Italia en el año 2005.

El movimiento de Hardware abierto o libre, busca crear una gran librería accesible para todo el mundo, lo que ayudaría a las compañías a reducir en millones de dólares en trabajos de diseño redundantes. Ya que es más fácil tener una lluvia de ideas propuesta por miles o millones de personas, que por solo una compañía propietaria del Hardware, tratando así de que la gente interesada entienda cómo funciona un dispositivo electrónico, pueda fabricarlo, programarlo y poner en práctica esas ideas en alianza con las empresas fa-bricantes, además se reduciría considerablemente la obsolescencia programada y en consecuencia evitaríamos tanta basura electrónica que contamina el medio ambiente. Al hablar de Open Hardware hay que especificar de qué tipo de Hardware se está hablando, ya que está clasificado en dos tipos:

- Hardware estático. Se refiere al conjunto de elementos materiales de los sistemas electrónicos (tarjetas de circuito impreso, resistencias, capacitores, LEDs, sensores, etcétera).

- Hardware reconfigurable. Es aquél que es descrito mediante un HDL (Hardware Description Language). Se desarrolla de manera similar a como se hace Software. Los diseños son archivos de texto que contienen el código fuente.

Para tener Hardware reconfigurable debemos usar algún lenguaje de programación con licencia GPL (General Public License). La licencia GPL, al ser un documento que cede ciertos derechos al usuario, asume la forma de un contrato, por lo que usualmente se le denomina contrato de licencia o acuerdo de licencia. La Organización Europea para la investigación Nuclear (CERN) publicó el 8 de julio de 2011 la versión 1.1 de la Licencia de Hardware Abierto.

Existen programas para diseñar circuitos electrónicos y aprender de la electrónica como EDA (Electronic Design Automation) y GEDA (GPL Electronic Design Automation), son aplicaciones de Software libre que permiten poner en práctica las ideas basadas en electrónica.

Es posible realizar el ciclo completo de diseño de Hardware reconfigurable desde una máquina con GNU/Linux, realizándose la compilación, simulación,

síntesis y descarga en una FPGA (Field Programmable Gate Arrays). Para la compilación y simulación se puede usar GHDL (https://ghdl.free.fr) junto con GTKWave (https://gtkwave.sourceforge.net) y para la síntesis el entorno ISE de Xilinx. Este último es Software comercial pero existe una versión gratuita con algunas restricciones.

Sabemos que tanto en el caso del Software como el Hardware, libre no es lo mismo que gratis. Específicamente, en el caso del Hardware, como estamos hablando de componentes físicos que son fabricados, la adquisición de componentes electrónicos puede ser costosa. Aun así, es un campo que no solo es apasionante sino que también tiene mucho futuro y representa grandes oportunidades.

Entusiasmo de la Comunidad Por último, pero no menos importante, probablemente el mayor impacto duradero de GNU/Linux en el modelo de ingeniería de Software se reduce a lo que podría llamarse entusiasmo de la comunidad. Me refiero a la forma en que GNU/Linux en particular, y el Software libre en general, ha animado a los desarrolladores de todo tipo a considerar las contribuciones a la comunidad como uno de sus objetivos finales y esto ahora es notorio no solo en Software, sino en Hardware abierto, obras literarias, escritos técnicos (como este trabajo), imágenes, vídeo, música y un largo etc.

En un mundo de código libre en el que las contribuciones a los proyectos de código pueden ser aceleradores de carrera y el código de licencia libre se reutiliza ampliamente, los desarrolladores entienden que hay un valor real en la construcción de Software que puede beneficiar a tantos usuarios como sea posible.

Tal vez los desarrolladores valorarían a la comunidad en su conjunto si GNU/Linux y el código libre nunca hubieran aparecido. Pero me cuesta imaginar un mundo en el que corporaciones como Microsoft y Google trabajarán juntos en la construcción de Software para GNU/Linux si GNU/Linux no hubiera popularizado el concepto de proyectos de Software impulsados por la comunidad que nadie posee realmente, pero que todos pueden utilizar.

Si bien, es innegable que todo lo anterior puso en la mira de las empresas de todos los tamaños y de las grandes corporaciones el Software libre, la principal razón es el poder utilizar una gran cantidad Software funcional, depurado y ampliamente usado para ofrecer servicios y/o productos basados en Software libre y así beneficiarse económicamente de ello.

Por otro lado, se ha visto a través de múltiples estudios, el impacto y la cuantificación de la importancia económica del código abierto aplicado tanto al Software como al Hardware, su efecto en la contribución al producto interior bruto generado, la reducción en aspectos como el coste total de propiedad, dependencia del proveedor y autonomía digital. Además de generar políticas públicas destinadas a lograr un sector público digitalmente autónomo, una investigación y desarrollo abierto que fomente el crecimiento de los países y una industria más digitalizada y competitiva.

Retomando la frase «el Software de código abierto nunca fue pensado para que las empresas de servicios lo tomasen y lo vendieran». ¿Para qué fue pensado el código abierto, entonces? No hay ninguna licencia de código abierto o Software libre reconocida por la Free Software Foundation o la Open Source Initiative que prohíba hacer negocio con el Software. Lo que prohíben es la discriminación en la capacidad y alcance de su uso en función de la parte, se trate de un individuo o de la mayor multinacional imaginable.

Pese a ello, esta situación está alterando el paradigma actual, en el que el modelo de desarrollo del código abierto se ha impuesto como impulsor de la innovación en el sector empresarial, a la postre el mayor estímulo que haya tenido el código abierto hasta la fecha.

¿Puede una filosofía de desarrollo como el código abierto, disponible para todo el mundo, llegar a convertirse en una fuente de ventajas diferenciales para el resto de los países, que se ha visto tradicionalmente muy superado en su relevancia en el entorno tecnológico por los gigantes tecnológicos de Estados Unidos o de China? Se afirma que su uso puede llegar a incidir en gran medida en el desarrollo de una independencia tecnológica superior, de una mayor competitividad y de más innovación. La idea, capitalizar la tecnología de una forma más orientada al procomún y al desarrollo colaborativo, suena sin duda atractiva e interesante pero no libre de inconvenientes para algunos sectores de desarrolladores de Software libre.

14.6 Código Abierto y las Organizaciones Internacionales

Aunque la Organización de las Naciones Unidas (ONU) ha hablado previamente bien del desarrollo del código abierto, varios eventos recientes muestran que la ONU está tomando medidas definitivas para presentar al mundo entero el camino del código abierto. En julio del 2021, el Consejo Económico y Social de la ONU (ECOSOC) adoptó un proyecto de resolución presentado por el representante de Pakistán titulado: Tecnologías de fuente abierta para

el desarrollo sostenible.

14.6.1 Las Naciones Unidas y el Código Abierto

El ECOSOC destacó la disponibilidad de tecnologías de código abierto que pueden contribuir a los Objetivos de Desarrollo Sostenible (ODS). El consejo invitó al Secretario General a "desarrollar propuestas específicas sobre formas de aprovechar mejor las tecnologías de código abierto para el desarrollo sostenible basadas en las aportaciones de los Estados Miembros interesados y otras partes interesadas".

El desarrollo de tecnología de código abierto puede ser una herramienta rápida y eficaz para la innovación. Aplicarlo a tecnologías apropiadas para ayudar a alcanzar los ODS es extremadamente prometedor. Las "tecnologías apropiadas" abarcan opciones y aplicaciones tecnológicas que son a pequeña escala, económicamente asequibles, descentralizadas, energéticamente eficientes, ambientalmente racionales y fácilmente utilizadas por las comunidades locales para satisfacer sus necesidades.

Existe un caso particularmente fuerte para las tecnologías apropiadas de código abierto OSAT (Outsourced Semiconductor Assembly and Test). OSAT podría ayudar a todos a salir de la pobreza y alcanzar un estado sostenible aprovechando el mismo tipo de desarrollo que hace que el Software de código abierto sea un éxito rotundo.

La Declaración Ministerial del Foro político de alto nivel sobre desarrollo sostenible también destacó la importancia de "tecnologías no patentadas que pueden contribuir a los Objetivos de Desarrollo Sostenible, a través de diversas fuentes de acceso abierto". Pidió "el desarrollo y la puesta en funcionamiento de una plataforma en línea en el marco del Mecanismo de facilitación de la tecnología para establecer un mapeo integral y servir como puerta de entrada a la información sobre iniciativas, mecanismos y programas de ciencia, tecnología e innovación existentes, dentro y fuera de las Naciones Unidas".

Es un pequeño paso, pero muy emocionante, porque las Naciones Unidas no se demoran una vez que ven formas de ayudar a sus Estados Miembros y a las personas que los integran. Ahora, el Departamento de Asuntos Económicos y Sociales de las Naciones Unidas (DESA) está trabajando para que esto suceda. DESA está utilizando una Nota sobre una base de datos centralizada propuesta de las Naciones Unidas de tecnologías apropiadas de código abierto publicada por la Conferencia de las Naciones Unidas sobre Comercio

y Desarrollo (UNCTAD) para hacerlo.

La Nota de la UNCTAD aboga por una base de datos centralizada de OSAT para acelerar el descubrimiento y la innovación en todos los sectores asociados con los ODS al tiempo que se minimizan los obstáculos legales o financieros. Esto es importante para la difusión del acervo mundial de conocimientos, especialmente en los países en desarrollo.

Actualmente, no existe un repositorio completo o una base de datos central de OSAT y Appropedia.org, quizás sea el mejor ejemplo. Sin embargo, la Nota de la UNCTAD dice: "Muchas organizaciones, organizaciones sin fines de lucro y empresas con fines de lucro están desarrollando OSAT y manteniendo bases de datos existentes a pequeña escala. Si bien hay muchos OSAT disponibles, se encuentran dispersos en varias bases de datos para tecnologías particulares. Mientras tanto, sigue existiendo una clara necesidad de aumentar la tasa de uso de OSAT.

Por lo tanto, existe una necesidad urgente de una base de datos de código abierto centralizada global (COSD) confiable. Al tener un alcance global, un repositorio de COSD proporcionaría una ventanilla única a la que todos pueden acceder para resolver los desafíos locales".

Concluye: "La ONU está bien posicionada para liderar el establecimiento de un COSD dado su papel bien establecido en la promoción de la tecnología de código abierto a través de varios foros y publicaciones intergubernamentales. En particular, 2030 Connect es una plataforma tecnológica en línea de la ONU que se desarrolló como parte del trabajo del Equipo de Trabajo Interinstitucional de la ONU. El COSD podría mejorarlo".

Con el liderazgo de la ONU, quizás no estemos demasiado lejos de cuándo, sí tiene un problema local (sin importar en qué parte del mundo se encuentre), pueda descargar una solución de código abierto examinada y probada. Quizás, esta es la potencia de fuego que necesitamos para alcanzar los ambiciosos Objetivos de Desarrollo Sostenible.

14.6.2 La Comisión Europea se Compromete a Liberar Todo el Software que Pueda Beneficiar a la Sociedad

A finales del 2021, la Unión Europea (UE) y su órgano legislativo, la Comisión Europea siguen avanzando en su estrategia digital con el Software de código abierto como uno de los pilares fundamentales. En esta ocasión ha sido esta última la que anuncia novedades para con la distribución del Software desarrollado para cubrir necesidades internas de la organización.

De acuerdo a la información publicada, la Comisión Europea ha aprobado una nueva regulación que favorece el libre acceso al Software que producen siempre y cuando existan beneficios potenciales para «los ciudadanos, las empresas u otros servicios públicos», lo que de la teoría a la práctica bien puede abarcar todo lo que se desarrolle bajo su tejado.

Esta nueva disposición se apoya a su vez en un reciente estudio realizado también por la Comisión sobre el impacto del Software de código abierto en áreas como la independencia tecnológica, la competitividad y la innovación en la economía de la Unión Europea. El objetivo, hallar evidencias sólidas con las que conformar las políticas europeas de código abierto para los próximos años.

En términos económicos, de hecho, los cálculos son de lo más optimistas y apuntan un impacto económico contundente, de miles de millones de euros de ahorro al año -a modo de ejemplo, se estimó entre 65 y 95 mil millones de euros solo en 2018- y con un incremento mínimo en la apuesta, se podría dar un crecimiento del PIB de la UE de en torno a los 100,000 millones de euros.

Con semejante escenario, no es de extrañar que la misma Comisión Europea esté interesada en promover las soluciones de código abierto dentro y fuera de las instituciones y no solo se basan en el beneficio económico directo: son muchas otras las ventajas del modelo también recogidas en el informe, tal y como se ha mencionado: independencia, competitividad, innovación... y en el caso de las administraciones públicas, colaboración, reutilización y transparencia.

En palabras de Johannes Hahn, comisario de Presupuesto y Administración: «El código abierto ofrece grandes ventajas en un ámbito en el que la UE puede desempeñar un papel de liderazgo. Las nuevas normas aumentarán la transparencia y ayudarán a la Comisión, así como a los ciudadanos, las empresas y los servicios públicos de toda Europa, a beneficiarse del desarrollo de Software de código abierto. Poner en común los esfuerzos para mejorar el Software y la creación conjunta de nuevas funciones reduce los costes para la sociedad, ya que también nos beneficiamos de las mejoras realizadas por otros desarrolladores. Esto también puede mejorar la seguridad, ya que especialistas externos e independientes comprueban los fallos y las deficiencias de seguridad de los programas informáticos».

La comisaría de Innovación, Investigación, Cultura, Educación y Juventud, Mariya Gabriel, ha declarado: «La Comisión pretende, con su ejemplo, estar al frente de la transición digital en Europa. Con las nuevas normas, la

Comisión aportará un valor significativo a las empresas, también las emergentes, a los innovadores, a los ciudadanos y las administraciones públicas, poniendo a su disposición el código abierto de sus soluciones informáticas. Esta decisión también ayudará a estimular la innovación, gracias al código de la Comisión disponible públicamente».

Como muestra del Software desarrollado bajo el amparo de la Comisión Europea que va a ser liberado se incluyen proyectos como eSignature, «un conjunto de normas, herramientas y servicios gratuitos que ayudan a las administraciones públicas y a las empresas a acelerar la creación y verificación de firmas electrónicas jurídicamente válidas en todos los Estados miembros de la UE»; o LEOS (Legislation Editing Open Software), «el Software utilizado en toda la Comisión para elaborar textos jurídicos. LEOS, escrito originalmente para la Comisión, se está desarrollando en estrecha colaboración con Alemania, España y Grecia».

Esta nueva iniciativa de la Comisión Europa contempla asimismo la creación de un repositorio centralizado para facilitar el descubrimiento, el acceso y la reutilización del Software incluido, el cual se sumará a todos los proyectos realizados por las diferentes administraciones públicas comunitarias en base al mismo modelo de desarrollo. Y viene de lejos este impulso, aun cuando comienza a unificarse ahora.

Sin ir más lejos, hace años que la propia Comisión Europea puso en marcha el programa Interoperable Delivery of European eGovernment Services to Public Administrations, Businesses and Citizens que dio origen al observatorio JoinUp (https://joinup.ec.europa.eu/), en cuyas páginas se recogen casi 3,000 soluciones de Software abierto, 133 colecciones de recursos y cuantiosa información relacionada.

Más tarde, de 2014 a 2017 se inició la «primera fase» en la estrategia de código abierto de la Unión Europea, especialmente dentro de la propia Comisión, estableciendo determinados requisitos en materia de Software de código abierto; actualmente se está desarrollando la nueva «estrategia de código abierto 2020-2023», con la que la Comisión Europea pretende ampliar y afianzar los objetivos de la estrategia digital y la contribución al programa Europa Digital.

15 Apéndice B: Seguridad, Privacidad y Vigilancia

Ante el constante aumento -explosivo- en el uso de dispositivos conectados a internet como computadoras personales, Laptops, tabletas y teléfonos celulares, expertos en ciberseguridad advierten un entorno propicio para que prosperen los cibercriminales y que, tanto individuos como empresas, se encuentren expuestos a múltiples amenazas de ciberseguridad.

En la actualidad, aproximadamente la mitad de la población mundial accede de algún modo a internet. Con tantos accesos concurrentes a la red de redes, la posible amenaza de seguridad a los sistemas informáticos crece y se complejiza, a pesar de las diversas y especializadas maneras de contrarrestarlas. Por su propia naturaleza, una conexión a internet hace que tu equipo de cómputo quede expuesto a ataques y pueda ser accesible por otro equipo, llegando a tener acceso a tu información.

¿Por qué?

- En muchas ocasiones, los usuarios no cuentan con la suficiente sensibilización sobre su exposición al riesgo, o bien, no están familiarizados con las herramientas y/o capacitación de sus organizaciones para prevenir y enfrentar amenazas de ciberseguridad.

- Los usuarios de internet a menudo utilizan redes Wi-Fi no seguras y usan dispositivos que no están configurados con los controles de políticas de seguridad básicos, lo cual los vuelve excepcionalmente vulnerables a ataques cibernéticos.

- Es común que los usuarios compartan dispositivos (computadoras, tabletas, teléfonos) para educación, trabajo y esparcimiento aumentando con ello su exposición a amenazas de ciberseguridad.

- Los piratas informáticos eligen como blanco la dependencia, cada vez mayor, de las personas con respecto a las herramientas digitales, además de que más tiempo en línea incrementa la potencial exposición de los usuarios a amenazas de ciberseguridad.

- Un error común es creer que los ciberatacantes utilizan únicamente herramientas muy avanzadas y técnicas para Hackear las computadoras

o cuentas de las personas. Los atacantes han aprendido que la forma más sencilla de robar la información de los usuarios, comprometer sus cuentas o infectar los sistemas es simplemente engañar al usuario para que lo hagan por ellos con una técnica denominada ingeniería social.

- Los piratas informáticos son extremadamente creativos al idear formas de aprovecharse de los usuarios y de la tecnología para acceder a contraseñas, redes y datos, a menudo sirviéndose de herramientas de ingeniería social y de temas y tendencias populares para tentar a los usuarios a tener comportamientos inseguros en línea.

Todo lo anterior, ha creado una enorme superficie de exposición a ataques cibernéticos dirigidos a los usuarios, la red, la computadora portátil, el teléfono inteligente, la tableta, etc. con la intención de cometer delitos informáticos. Por ello estos lineamientos generales de ciberseguridad son sugeridos para el uso seguro de redes y dispositivos de telecomunicaciones en apoyo al usuario, contiene recomendaciones sencillas y prácticas para fomentar una buena salud cibernética de los dispositivos utilizados para el trabajo a distancia, educación y diversión; y promover la ciberseguridad entre las personas y sus organizaciones.

La seguridad de la información en la red es más que un problema de protección de datos, y debe estar básicamente orientada en asegurar la información importante de las personas, de las organizaciones y la propiedad intelectual. Los riesgos de la información están presentes cuando confluyen fundamentalmente dos elementos: las amenazas de ataques, y las vulnerabilidades de la tecnología; conceptos íntimamente relacionados donde no es posible ninguna consecuencia sin la presencia conjunta de estos.

15.1 ¿Qué es la Privacidad y por qué es Importante?

Consideramos que la privacidad debería ser un derecho básico y una protección necesaria en la era digital para evitar la victimización y la manipulación. Una sociedad no puede tener libertad sin privacidad. Puede parecer un lujo, pero es importante para el bienestar de una sociedad libre y justa. Muchos gobiernos y empleadores hacen vigilancia[306] activa a sus ciudadanos o empleados para monitorear ideas, discusiones o disensiones no deseadas. Los

[306] ¿Qué es vigilancia? Vigilancia es el seguimiento de las comunicaciones, acciones o movimientos de una persona por un gobierno, empresa, grupo o persona.

infractores son luego procesados o reeducados para alinearse con lo que las autoridades consideran apropiado. Sin el beneficio del anonimato, el deseo de los ciudadanos de expresar sus pensamientos se reprime efectivamente.

En la era digital, la privacidad va más allá del anonimato, al proteger a las personas de la victimización y la manipulación. La sociedad ha adoptado la tecnología para educarse, comunicarse, realizar negocios y formar relaciones. Nuestros puntos de vista y opiniones están fuertemente influenciados por lo que aprendemos de las fuentes de noticias locales, nacionales e internacionales.

Contribuimos en gran medida al panorama digital a través de nuestras acciones y decisiones. Nuestras huellas digitales están en todas partes. Cuentan una historia de a dónde vamos, qué hacemos, quién nos gusta o no, y qué pensamos. Se crean con cada Clic que hacemos y con cada archivo, aplicación y dispositivo que usamos. Cuando esos datos se agregan, pueden proporcionar información tremendamente poderosa sobre una persona o comunidad, lo suficiente como para construir personas complejas y precisas.

Esta información se usa comúnmente para manipular las creencias y comportamientos de las personas. Las compras en línea son un ejemplo perfecto: el Marketing dirigido y la publicidad basada en datos es un gran negocio porque logra que la gente gaste dinero. Todo se reduce a saber lo que las personas hacen, piensan, dicen, consumen y miran[307]. Tener acceso a grandes

¿Cuándo es legal la vigilancia? En general, cuando es selectiva, se basa en indicios suficientes de conducta delictiva y está autorizada por una autoridad estrictamente independiente, como un juez.

¿Qué es la vigilancia masiva? La vigilancia masiva indiscriminada es el control de las comunicaciones por internet y telefónicas de un gran número de personas -a veces de países enteros- sin que existan indicios suficientes de conducta delictiva. Este tipo de vigilancia no es legal.

¿Qué datos recogen? Algunos gobiernos almacenan y analizan historiales de navegación, búsquedas en internet, mensajes de correo electrónico, mensajes instantáneos, conversaciones por Webcam y llamadas telefónicas. También reúnen metadatos -datos sobre datos-, como destinatarios de correo electrónico, horas de llamadas y registros de ubicación.

¿Qué hacen con mis datos? Se guardan en grandes centros de datos donde unos algoritmos informáticos pueden hacer búsquedas en ellos y analizarlos. También están a disposición de las autoridades de las agencias de seguridad a través de potentes bases de datos como XKeyscore.

[307] Has notado que si buscas algo en un buscador comercial -como Google, Yahoo, Bing, Ask, Baidu, etc.-, minutos después en tus distintas redes sociales aparecerán mensajes relacionados a tu búsqueda. Esto ocurre porque los buscadores comerciales en cada búsqueda que haces, vídeo o los anuncios que ves o en los que haces Clic comparten tu ubicación,

cantidades de datos privados les brinda a los anunciantes la capacidad de crear mensajes oportunos y significativos que atraigan a las personas a los comportamientos deseados.

Pero si los minoristas pueden hacer que las personas compren cosas que no necesitan, ¿para qué más se pueden usar los datos privados? ¿Qué tal cambiar lo que piensa la gente, a quiénes apoyan, sus opiniones políticas, qué debería convertirse en ley y en qué creer? El uso de información privada se ha aprovechado durante mucho tiempo para promover, vilipendiar o perseguir a diversas religiones, partidos políticos y líderes.

En las últimas décadas, ha cambiado la forma en que los ciudadanos del mundo reciben sus noticias. Los segmentos de noticias y entretenimiento han comenzado a mezclarse, a menudo informando hechos con adornos e historias obstinadas para influir en las opiniones públicas. Cuanta más información privada se conozca, más fácil será influir, convencer, engatusar o amenazar a las personas.

Según informes de Oxford Internet Institute, a pesar de los esfuerzos para combatir la propaganda computacional, el problema ha crecido a gran escala. El mayor crecimiento proviene de propaganda política que es difundida junto con desinformación y noticias basura alrededor de los períodos electorales. Se incrementa el número de campañas políticas a nivel mundial que usan Bots, noticias basura y desinformación para polarizar y manipular a los votantes.

Un velo de privacidad puede proteger tanto los beneficios como los abusos. La tendencia actual es establecer y ampliar los derechos de privacidad en beneficio de los ciudadanos. Esto reduce la victimización, manipulación y explotación digitales al proteger los datos confidenciales y permite actividades que promueven la libertad y la libertad de expresión.

Sin leyes, los gobiernos y las empresas han desarrollado prácticas que aprovechan el poder de recopilar información confidencial y utilizarla en su propio beneficio. Las nuevas leyes de privacidad están cambiando el panorama y muchas empresas éticas reducen sus esfuerzos de cobranza para ser más conservadoras y respetuosas. También muestran flexibilidad en la forma en que tratan, protegen y comparten dichos datos.

Algunos gobiernos y agencias también están reduciendo la recolección, limitando la retención o terminando los programas domésticos que los ciu-

los sitios que visitas, los dispositivos, navegadores y aplicaciones que usas para acceder a los servicios del buscador con fines comerciales.

Una opción para nuestra privacidad es usar buscadores que no dejan rastro de nuestras búsquedas como puede ser: DuckDuckGo.

dadanos consideran invasivos. Al mismo tiempo, los organismos encargados de hacer cumplir la ley quieren conservar la capacidad para detectar e investigar delitos, para proteger la seguridad de los ciudadanos.

También se hace un mal uso de la privacidad. Es la herramienta preferida por quienes cometen delitos y permite que los actos atroces contra otros no se detecten. Puede ocultar actos terribles y permitir una coordinación generalizada entre el fraude, abuso y terror.

Se argumenta que las puertas traseras digitales, las claves maestras y los algoritmos de cifrado que obtienen acceso a los sistemas y la información privada ayudarían en la detección legal de actividades delictivas y en las investigaciones para identificar a los terroristas. Aunque suena como una gran herramienta contra los delincuentes, es una caja de Pandora.

Las puertas traseras y las llaves maestras no limitan el acceso para una investigación específica donde existe una causa probable, sino que permiten una vigilancia generalizada[308] y la recolección de datos de toda una población, incluidos los ciudadanos respetuosos de la ley. Esto viola el derecho de las personas a la privacidad y abre la puerta a la manipulación y el enjuiciamiento político. La capacidad de leer cada texto, correo electrónico, mensaje y conversación en línea para "monitorear" a la población crea un camino claro hacia el abuso. El riesgo de control y explotación es real.

Incluso para aquellos que no tienen ninguna objeción a que su gobierno

[308] El 5 de junio de 2013, el exanalista de la CIA y de la NSA Edward Snowden decidió revelar la existencia de programas de vigilancia sobre las comunicaciones de millones de ciudadanos de todo el mundo. A través de The Guardian y The Washington Post, supimos que, en nombre de la seguridad y sin ningún control judicial, la NSA y el gobierno británico habían rastreado e-mails, llamadas telefónicas y mensajes encriptados. Empresas como Facebook, Google y Microsoft habían sido obligadas a entregar información de sus clientes por órdenes secretas de la NSA. Esta misma agencia grabó, almacenó y analizó los 'metadatos' de las llamadas y de los mensajes de texto enviados en México, Kenia y Filipinas. Incluso llegaron a espiar el móvil de la canciller alemana Angela Merkel.

Snowden hizo las filtraciones a la prensa desde Hong Kong y actualmente vive en Rusia, donde se le concedió asilo. No puede volver a su país porque está acusado de revelar información clasificada a personas no autorizadas y de robo de propiedad del gobierno federal. Para unos es un héroe y para otros un traidor, gracias a las revelaciones de Snowden la opinión pública es más consciente de su derecho a la privacidad y ha reaccionado oponiéndose al espionaje masivo.

Aunque queda un largo camino para asegurar que los Gobiernos no invadan la vida privada de las personas, los tribunales han declarado ilegales algunos aspectos de estos programas y las empresas tecnológicas han tenido que posicionarse ante un escándalo capaz de dañar seriamente su reputación.

tenga acceso, debemos considerar el hecho de que tales puertas traseras y claves maestras serían buscadas por ciberdelincuentes y otros actores del estado-nación. Ningún sistema es infalible. Con el tiempo, los delincuentes encontrarían y utilizarían estas herramientas en detrimento de la comunidad digital mundial. Algunas puertas traseras podrían valer decenas de miles de millones de dólares para el comprador adecuado, ya que podrían desbloquear un poder inimaginable para apoderarse de la riqueza, afectar a la gente, dañar naciones, socavar la independencia y reprimir el pensamiento libre.

Proteger la privacidad no se trata de ocultar información. Se trata de la capacidad de liberarse de influencias no deseadas, de la tiranía y de comunicarse con los demás de formas que desafíen el statu quo. La privacidad protege a las personas, pero también los cimientos de una sociedad libre. La privacidad no es un tema fácil y no existe una solución perfecta. Es una situación dinámica y seguirá cambiando con el sentimiento público.

Todos quieren cierto nivel de discreción, confidencialidad y espacio. Nadie quiere que se expongan sus contraseñas, finanzas familiares, detalles de relaciones personales, historial médico, ubicación, compras y discusiones privadas. Las personas tampoco disfrutan de verse inundadas de Spam, Phishing y llamadas de ventas incesantes. La privacidad no se trata necesariamente de ocultar algo, sino de limitar la información a quienes tienen derecho a saber.

Muy poca privacidad puede socavar la libertad de expresión, la libertad y la denuncia de victimizaciones. También empodera a entidades poderosas para manipular el mundo digital de las personas para coaccionarlas, manipularlas y victimizarlas. Demasiada privacidad puede permitir que los actores criminales prosperen y se escondan de las autoridades. Debe lograrse un equilibrio[309].

[309] Los gobiernos nos enfrentan a un dilema falso: seguridad o libertad. En un estado de derecho, donde las leyes equilibran ambos conceptos, las personas son inocentes hasta que se demuestra lo contrario y tienen derecho a que se respete su vida privada. Por tanto, antes de violar estos derechos, los gobiernos deben de tener indicios de que se está cometiendo un delito. No pueden buscar pruebas aleatoriamente en nuestras comunicaciones privadas antes de que se cometa ese delito.

Varios países dan "carta blanca" a la vigilancia indiscriminada en sus leyes. En Francia se permite interceptar masivamente comunicaciones, retener información durante largos períodos de tiempo y se ha eliminado la autorización judicial previa. También Reino Unido ha introducido en su legislación mayores poderes de espionaje. Polonia ha otorgado poderes de vigilancia incompatibles con respecto a la privacidad a la policía y o otras agencias. Otros países como China o Rusia también vigilan internet con total desprecio a

Algunos gobiernos y empresas tecnológicas quieren que aceptemos que no tenemos derechos cuando estamos en internet. Que cuando usamos el teléfono o entramos en nuestra cuenta de correo electrónico, todo lo que hacemos o decimos les pertenece. No permitiríamos este grado de intrusión en nuestra vida fuera de internet, así que no debemos permitirlo dentro.

Cuando los gobiernos no protegen los derechos humanos, sólo la acción de la gente común y corriente puede hacer que las cosas cambien y que los responsables de los abusos rindan cuentas. Muchas organizaciones trabajan para que los gobiernos prohíban la vigilancia masiva y el intercambio ilegal de información confidencial. Esta será una larga lucha no exenta de dificultades, pero cuyo impulso va creciendo con el tiempo.

¿Qué son los datos biométricos? en general entenderemos por datos biométricos a las características físicas, fisiológicas, morfológicas, de comportamiento y rasgos de personalidad distintivas de cada persona que permite distinguir ciertas singularidades e identificar al individuo en cuestión.

El uso de datos biométricos no es nuevo e incluso es algo en lo que constantemente se ven envueltos las redes sociales[310], aunque los fines de uso son diversos. Recientemente Texas demandó a Meta, dueña de Facebook, por almacenar millones de estos datos y comercializarla con terceros.

Las Amenazas a los Usuarios en un entorno dinámico de interconectividad, pueden venir de cualquier parte, sea interna o externa, e íntimamente relacionada con el entorno de las organizaciones. Las vulnerabilidades son una debilidad en la tecnología o en los procesos asociados con la información, y como tal, se consideran características propias de los sistemas o de la infraestructura que los soporta.

Las personas o usuarios de internet que emplean las tecnologías para vulnerar los sistemas en la red, robar información y/o infectarlos con comportamientos dudosos, son comúnmente conocidos como *Crackers*.

la intimidad de las personas.

[310] Por ejemplo en el caso de X antes Twitter, podrá hacerse de tu foto de perfil, así como otros datos que incluyen historial de empleo, educación, preferencias de empleo, capacidades y habilidades, así como búsqueda de trabajo, datos que utilizaría para ofrecer servicios similares a los de Linkedln, "recomendarte potenciales trabajos, compartir con potenciales empleadores cuando solicitas un trabajo, permitir que los empleadores encuentren potenciales candidatos y mostrarte publicidad más relevante".

El término Hacker[311] en el sentido más filosófico tiende a promover una conciencia colectiva de la libertad del conocimiento y la justicia social, por lo que muchas veces se los encuentra en situaciones de activismo (llamado en este caso Hacktivismo) en pos de dicha ideología. Su forma de actuar, por lo general, determina su clasificación en:

- *Hacker* de Sombrero Blanco (White Hat): éticos, expertos en seguridad informática, especializados en realizar test de intrusión y evaluaciones de seguridad.
- *Hacker* de Sombrero Negro (Black Hat): también conocidos como *Crackers*, vulneran los sistemas de información con fines maliciosos.
- *Hacker* de Sombrero Gris (Grey Hat): en ocasiones vulneran la ley, y de forma general no atacan malintencionadamente o con intereses personales, sino que sus motivaciones se relacionan a protestas o desafíos personales.
- *Hacker* de Sombrero Verde (Green Hat): son novatos que pueden evolucionar y buscan convertirse en expertos.
- *Hacker* de Sombrero Azul (Blue Hat): son personas expertas que se les paga para probar Software en busca de errores antes de que éste salga al mercado.

Para una entidad, la fuga de información provocada por el actuar de algunos de estos usuarios de la red, puede ocurrir deliberadamente como resultado de una acción intencional de algún empleado descontento, como consecuencia de un ciberataque, o inadvertidamente, por un colaborador desprevenido víctima de un Software malicioso.

Una de las principales amenazas para los dispositivos tecnológicos utilizados para el trabajo y estudio a distancia es el Malware, también conocido como Software o código malicioso. Éste se define como cualquier programa informático que se coloca de forma oculta en un dispositivo, con la intención de comprometer la confidencialidad, integridad o disponibilidad de los datos, las aplicaciones o el sistema operativo.

[311] Existen algunos otros términos en el ciberespacio, por ejemplo: Newbie, que significa principiante; Lammer, persona que presume tener conocimientos que realmente no posee; Phreaker, Hacker orientado a los sistemas telefónicos; Script Kiddie, quien utiliza programas creados por terceros sin conocer su funcionamiento.

Los tipos más comunes de amenazas de Malware incluyen Virus, gusanos, troyanos, Rootkits y Spyware. Las amenazas de Malware pueden infectar cualquier dispositivo por medio del correo electrónico, los sitios Web, las descargas y el uso compartido de archivos, el Software punto a punto y la mensajería instantánea.

Además, existen amenazas relacionadas con la ingeniería social como el Phishing, Smishing y Vishing, por medio de los cuales los atacantes intentan engañar a las personas para que revelen información confidencial o realicen ciertas acciones, como descargar y ejecutar archivos que parecen ser benignos, pero que en realidad son maliciosos.

15.2 Las Vulnerabilidades y Exposiciones Comunes

De manera global, la última década ha sido testigo del cambio de paradigma en que los atacantes buscan explotar vulnerabilidades dentro de las organizaciones y las infraestructuras de redes. Con el fin de contrarrestar estos ataques, las políticas de seguridad persiguen constantemente aprender de ellos para estar preparados lo mejor posible, y en este sentido, intentar garantizar confianza y tranquilidad a los usuarios de la red, sobre el empleo de sus datos, finanzas y propiedades intelectuales.

Los ataques de seguridad vienen en muchas formas y usan varios puntos de entrada. Cada tipo de ataque viene en varios tipos, ya que generalmente hay más de una forma en que se pueden configurar o camuflar en función de la experiencia, los recursos y la determinación del pirata informático.

Las Vulnerabilidades y Exposiciones Comunes (Common Vulnerabilities and Exposures, CVE[312]) que tienen los distintos sistemas operativos (productos), es una lista de información registrada sobre vulnerabilidades de seguridad conocidas, en la que cada referencia tiene un número de identificación llamada CVE-ID, descripción de la vulnerabilidad, que versiones del Software están afectadas, posible solución al fallo (si existe) o como confi-

[312] https://www.cvedetails.com/
https://www.cvedetails.com/top-50-products.php
https://thebestvpn.com/vulnerability-alerts/
Hay que notar que en la base de datos se diferencian distintos productos de Windows (como Windows 7.0 u 8.1) pero no se hace lo mismo con los demás productos (por ejemplo para Debian GNU/Linux están los errores desde 1999 a la fecha) con lo que es engañosa la comparación del número vulnerabilidades para un sistema operativo con tantas versiones generadas en dicho tiempo con otro con un tiempo de vida corto.

gurar para mitigar la vulnerabilidad y referencias a publicaciones o entradas de foros o Blogs donde se ha hecho pública la vulnerabilidad o se demuestra su explotación. Además suele también mostrarse un enlace directo a la información de la base de datos de vulnerabilidades (https://nvd.nist.gov, https://openssf.org y https://docs.aws.amazon.com/security), en la que pueden conseguirse más detalles de la vulnerabilidad y su valoración.

El mundo está cada vez más interconectado y, como resultado de esto, la exposición a las vulnerabilidades de seguridad también ha aumentado dramáticamente. Las complejidades de mantener las plataformas de cómputo actuales hacen que sea muy difícil para los desarrolladores cubrir cada punto de entrada potencial. En 2019 hubo un promedio de más de 45 vulnerabilidades y exposiciones comunes registradas por día y estas siguen en aumento año con año.

El CVE-ID ofrece una nomenclatura estándar para identificación de la vulnerabilidad de forma inequívoca que es usada en la mayoría de repositorios de vulnerabilidades.

Es definido y es mantenido por The MITRE Corporation (por eso a veces a la lista se la conoce por el nombre MITRE CVE List) con fondos de la National Cyber Security Division del gobierno de los Estados Unidos de América. Forma parte del llamado Security Content Automation Protocol.

Por otro lado, como parte de las amplias revelaciones sobre vigilancia masiva filtradas en 2013, 2014 y años posteriores, se descubrió que las agencias de inteligencia estadounidenses y británicas, la Agencia de Seguridad Nacional (NSA, por sus siglas en inglés) y el Cuartel General de Comunicaciones del Gobierno (GCHQ, por sus siglas en inglés), respectivamente, tienen acceso a los datos de los usuarios de dispositivos Android. Estas agencias son capaces de leer casi toda la información del teléfono como los SMS, geolocalización, correos, notas o mensajes.

Documentos filtrados en enero de 2014, revelaron que las agencias interceptan información personal a través de internet, redes sociales y aplicaciones populares, como Angry Birds, que recopilan información para temas comerciales y de publicidad. Además, según The Guardian, el GCHQ tiene una wiki con guías de las diferentes aplicaciones y redes de publicidad para saber los diferentes datos que pueden ser interceptados. Una semana después de salir esta información a la luz, el desarrollador finlandés Rovio, anunció que estaba reconsiderando sus relaciones con las distintas plataformas publicitarias y exhortó a la industria en general a hacer lo mismo.

Las informaciones revelaron que las agencias realizan un esfuerzo adi-

cional para interceptar búsquedas en Google Maps desde Android y otros teléfonos inteligentes para recopilar ubicaciones de forma masiva. La NSA y el GCHQ insistieron en que estas actividades cumplen con las leyes nacionales e internacionales, aunque The Guardian afirmó que «las últimas revelaciones podrían sumarse a la creciente preocupación pública acerca de cómo se acumula y utiliza la información, especialmente para aquellos fuera de Estados Unidos de Norte América, que gozan de menos protección en temas de privacidad que los estadounidenses».

15.3 Alfabetismo Digital

Según la organización Common Sense Media, el alfabetismo digital es la capacidad de encontrar, identificar, evaluar y usar la información encontrada en medios digitales de manera efectiva. Básicamente, es la misma definición tradicional de alfabetismo, pero adaptada a la era digital y a fuentes no tradicionales de información.

El anuario del 2016 de la UNESCO en "Alfabetización mediática e informacional para los objetivos de desarrollo sostenible" hace referencia a las "Cinco leyes de la alfabetización mediática e informacional":

1. La información, la comunicación, las bibliotecas, los medios de comunicación, la tecnología, el internet y otras fuentes de información se encuentran en la misma categoría. Ninguna es más relevante que la otra ni debe ser tratada como tal.

2. Cada ciudadano es un creador de información o conocimiento y tiene un mensaje. Todas las personas deben estar facultadas para acceder a nueva información y expresarse.

3. La información, el conocimiento y los mensajes no siempre están exentos de valores o prejuicios. Cualquier conceptualización, uso y aplicación de alfabetismo digital debe presentar este hecho de manera transparente y comprensible para todos los ciudadanos.

4. Todo ciudadano desea conocer y comprender información, conocimientos y mensajes nuevos, así como comunicarse; y sus derechos nunca deben ser comprometidos.

5. El alfabetismo digital es un proceso dinámico de experiencias vividas. Se considera completa cuando incluye conocimientos, habilidades y actitudes, cuando abarca el acceso, la evaluación, el uso, la producción y la comunicación de información, de contenido mediático y tecnológico.

De igual manera, en el anuario de la UNESCO se exponen 10 habilidades que deben desarrollarse para lograr la alfabetización digital, o como lo define el anuario, alfabetización mediática e informacional. Estas habilidades son:

- Interactuar con información referente a los medios y la tecnología.
- Ser capaz de aplicar habilidades técnicas de comunicación de información para procesar la información y producir contenido mediático.
- Utilizar, de manera ética y responsable la información y comunicar su comprensión o conocimiento adquirido a una audiencia o lectores en una forma y medio apropiados.
- Extraer y organizar información y contenidos.
- Evaluar de forma crítica la información y el contenido presentado en los medios y otras fuentes de información, incluyendo medios en línea, en términos de autoridad, credibilidad, propósito y posibles riesgos.
- Localizar y acceder a información de contenido relevante.
- Sintetizar las ideas extraídas del contenido.
- Comprender las condiciones bajo las cuales se pueden cumplir esas ideas o funciones.
- Comprender el papel y las funciones de los medios, incluyendo medios en línea, en la sociedad y su desarrollo.
- Reconocer y articular la necesidad de información y de los medios.

Internet ha sido una herramienta que ha transformado y definido la comunicación en el siglo XXI. A través de sus múltiples interfaces, internet ha tenido éxito, para que tanto individuos como organizaciones se conecten, se comuniquen e intercambien información. Las plataformas tecnológicas y las redes sociales han acelerado la velocidad a través de la cual los usuarios

pueden acceder y recuperar información, simplificando el proceso en el que las noticias se difunden, actualizan e incluso se comunican. Hoy en día, es prácticamente instantáneo darse cuenta de un evento de noticias sin que sea necesariamente comunicado por medios tradicionales como los periódicos o la radio.

La facilidad a través de la cual las personas ahora pueden comunicarse ha traído una sensación de democratización a la libertad de expresión. Transformar la libertad de expresión ha posibilitado nuevas capacidades para crear y editar contenido, generando nuevas oportunidades para el periodismo alternativo; nuevas capacidades de organización y movilización (que apoyan en gran medida otros derechos, como la libertad de asociación); y nuevas posibilidades para innovar y generar desarrollo económico (apoyando los derechos sociales y económicos).

Sin embargo, esta facilidad en el intercambio y la creación de información también presenta desafíos tanto para las organizaciones como para las personas que son usuarias de dichas redes, tanto como fuente, como usuario final. Aunque estos desafíos varían en escala, todos son igualmente significativos. Algunos de los más destacados incluyen el desafío a la calidad superficial de la información, la susceptibilidad a la información errónea y la exposición a ataques cibernéticos. Por lo tanto, la necesidad de mitigar y mantener la integridad de esta información se ha convertido en un área de trabajo creciente para muchas organizaciones públicas y privadas.

En las siguientes secciones se ofrece una variedad de técnicas y mejores prácticas para mitigar y contrarrestar los desafíos mencionados anteriormente. Sin embargo, es importante tener en cuenta, al leer estas recomendaciones, la posición que representas o con la que te asocias. Algunas de las recomendaciones pueden no ser aplicables a una figura pública, como políticos, activistas u otros actores cuyas mejores prácticas en las redes sociales están sujetas a un mayor escrutinio. En este sentido, el ejercicio de los derechos de expresión, reunión y protesta se debe respetar, y debe ser respetado, en el ámbito digital al tiempo que se garanticen prácticas más seguras de internet.

15.4 Amenazas a la Ciberseguridad

La aparición de vulnerabilidades en los sistemas operativos y los métodos de encubrimiento de los atacantes, lo convierten en una práctica en aumento. Algunos de los principales ataques en la red, hacia donde dirigen su mirada

los *Hackers* para vulnerar la seguridad, pueden definirse como:

- Shoulder Surfing: es una técnica mediante la que el ciberdelincuente consigue nuestra información mirando "por encima del hombro" desde una posición cercana, mientras utilizamos los dispositivos sin darnos cuenta.

- Dumpster Diving: se le conoce como el proceso de buscar en nuestra basura para obtener información útil sobre nuestra persona o empresa que luego puede utilizarse contra nosotros para otro tipo de ataques.

- Malware: el término se refiere de forma genérica a cualquier Software malicioso que tiene por objetivo infiltrarse en un sistema para dañarlo. Comúnmente se asocian como tipos de Malware a los virus, gusanos y troyanos.

- Virus: es un código que infecta los archivos del sistema mediante un programa maligno, pero para ello necesita que el usuario lo ejecute directamente. Una vez activo, se disemina por todo el sistema a donde el equipo o cuenta de usuario tenga acceso, desde dispositivos de Hardware hasta unidades virtuales o ubicaciones remotas en una red.

- Gusanos: es un programa que, una vez infectado el equipo, realiza copias de sí mismo y las difunde por la red. A diferencia del virus, no necesita la intervención del usuario, ya que pueden transmitirse utilizando las redes o el correo electrónico. Son difíciles de detectar, pues su objetivo es difundirse e infectar a otros equipos, y no afectan inicialmente el funcionamiento normal del sistema. Su uso principal es el de la creación de redes zombies (Botnets), utilizadas para ejecutar acciones de forma remota como ataque de denegación de servicio (DoS) a otro sistema.

- Troyanos: similares a los virus, sin embargo, mientras que este último es destructivo por sí mismo, el troyano lo que busca es abrir una puerta trasera (Backdoor) para favorecer la entrada de otros programas maliciosos. Su misión es precisamente pasar desapercibido e ingresar a los sistemas sin que sea detectado como una amenaza potencial. No se propagan a sí mismos y suelen estar integrados en archivos ejecutables aparentemente inofensivos.

- Spyware: es un programa espía, cuyo objetivo es recopilar información de un equipo y transmitirla a una entidad externa sin el consentimiento del propietario. Su trabajo suele ser silencioso, sin dar muestras de su funcionamiento, llegando incluso a instalar otros programas sin que se perciban. Las consecuencias de su infección incluyen, además, pérdida considerable del rendimiento del sistema y dificultad para conectarse a internet.

- AdWare: su función principal es la de mostrar publicidad. Aunque su intención no es la de dañar equipos, es considerado por algunos una clase de Spyware, ya que puede llegar a recopilar y transmitir datos para estudiar el comportamiento de los usuarios y orientar mejor el tipo de publicidad.

- Ransomware[313]: este es uno de los más sofisticados y modernos ataques, ya que lo que hace es secuestrar datos (encriptándolos) y pedir un

[313] El Ransomware se ha convertido en la principal amenaza para la ciberseguridad mundial. Desde que el troyano WannaCry afectó en la primavera de 2017 al menos a 200.000 equipos y servidores de 150 países, poniendo 'contra las cuerdas' a importantes empresas, el uso de este tipo de ataques informáticos no ha dejado de aumentar y son cada vez más numerosos, sofisticados, peligrosos y masivos.

Si en sus inicios los atacantes se conformaban infectando ordenadores de consumo a cambio de unos pocos dólares, todos los informes apuntan que los ciberdelincuentes están enfocando su ámbito de actuación preferente al segmento empresarial, organizaciones, administraciones e infraestructuras públicas con Malware tan peligroso como 'CryptoWall', 'Babuk', 'Black Kingdom', 'Ryuk' o 'CryptoLocker' que destacan por su alto nivel de código y su capacidad de control de ordenadores y redes mediante el cifrado de archivos.

El número de víctimas por Ransomware es ya interminable. La última conocida (abril 2021) ha sido la cadena de tiendas Phone House, pero la lista de los últimos meses es amplísima: la multinacional Acer; el estudio CD Projekt Red; la asociación deportiva NBA; el líder en cámaras Canon; el desarrollador japonés Capcom; la firma de seguros Mapfre; municipios y hospitales estadounidenses o la infraestructura del SEPE, el Servicio Público de Empleo Estatal de España que gestiona subsidios y maneja datos personales de millones de desempleados.

Solo es un ejemplo de afectados y, además, se sospecha que otras empresas han recibido ataques, aunque han preferido no divulgarlos públicamente ante la pérdida reputacional que suponen estos incidentes que son una lacra que parece imparable. Realmente, no hay sistema operativo, plataforma, dispositivo o red informática que esté a salvo, porque el Ransomware emplea cualquier tipo de vulnerabilidad, tipo de Malware o de ataque para secuestrar los equipos. Y no en todas las ocasiones es detectado a tiempo por los sistemas de seguridad y Software antimalware.

Teniendo en cuenta que un Ransomware típico puede infectar dispositivo móviles, ordenadores personales, servidores o redes, bloqueando su funcionamiento y/o acceso a una

rescate por ellos. Normalmente, se solicita una transferencia en dinero electrónico (Bitcoins), para evitar el rastreo y localización. Este tipo de ciberataque va en aumento y es uno de los más temidos en la actualidad.

- Stalkerware: se trata de un programa que permite el seguimiento y monitorización de la actividad del usuario en un dispositivo móvil como teléfono, tableta o computadora. El problema de este tipo de programas es que no han sido creados puntualmente para el espionaje y el acoso, sino para el intercambio de datos entre dispositivos de manera más sencilla. Y qué sí se instalan sin el consentimiento de la persona, permite espiar sus comunicaciones y toda su actividad gracias a las funcionalidades y sensores incorporados en el dispositivo.

- Escaneo de Puertos: técnica empleada para auditar dispositivos y redes con el fin de conocer qué puertos están abiertos o cerrados, los servicios que son ofrecidos, así como comprobar la existencia de algún cortafuegos (Firewall), la arquitectura de la red, o el sistema operativo, entre otros aspectos. Su empleo permite al atacante realizar un análisis preliminar del sistema y sus vulnerabilidades, con miras a algún otro tipo de ataque, pues cada puerto abierto en un dispositivo, es una potencial puerta de entrada al mismo.

- Phishing: no es un Software, se trata más bien de diversas técnicas de suplantación de identidad para obtener datos privados de las víctimas, como contraseñas o datos bancarios. Los medios más utilizados son el correo electrónico, mensajería o llamadas telefónicas, y se hacen pasar por alguna entidad u organización conocida, solicitando datos confidenciales, para posteriormente ser utilizado por terceros en su beneficio.

- Whaling: es un método para simular ocupar cargos de nivel superior en una organización con el objeto de conseguir información confidencial u obtener acceso a sistemas informáticos con fines delictivos.

- El Smishing: ocurre cuando se recibe un mensaje de texto corto (SMS)

parte o a todo el equipo apoderándose de los archivos con un cifrado fuerte y exigiendo una cantidad de dinero como "rescate" para liberarlos, el mejor (y casi único) de los consejos en ciberseguridad es la prevención con las copias de seguridad como máximo exponente. La opción de pagar el rescate para recuperar el acceso a los archivos es muy negativa para la industria ya que retroalimenta aún más esta amenaza.

en el teléfono celular, por medio del cual se solicita al usuario llamar a un número de teléfono o ir a un sitio Web.

- El Vishing: es la estafa que se produce mediante una llamada telefónica que busca engañar, suplantando la identidad de una persona o entidad para solicitar información privada o realizar alguna acción en contra de la víctima.

- Juice-jacking o Video-jacking: son los nombres que se han puesto a los procesos por los cuales, a través de un puerto (generalmente USB) nos conectamos a un puerto Hackeado, de esta forma el ciberdelincuente puede instalar, grabar datos, tomar video o sacar datos de nuestros dispositivos móviles. Esto se ha vuelto común en los puertos de recarga de energía públicos a través del puerto USB. Si se hará uso de este tipo de servicios, es recomendable adquirir un dispositivo del tipo PortaPow de carga rápida con adaptador USB que inhabilitan los pines de datos permitiendo sólo la carga del dispositivo.

- Keylogger (registrador de teclas): es un Software malicioso o dispositivo en Hardware -generalmente conectado al teclado- que se encarga de registrar las pulsaciones que se realizan en el teclado, para posteriormente usarlas para robar información privada.

- Criptomining es un Malware diseñado para la extracción de Criptomonedas en nuestros dispositivos. Si el usuario accesa a un sitio Web que esté infectado, el Malware se puede descargar de forma inadvertida a través de una descarga automática y nuestro dispositivo comenzará a desenterrar una moneda criptográfica seleccionada para los Hackers; la infección será notoria por un uso intensivo de nuestro dispositivo.

- Botnets (Redes de robots): Son computadoras o dispositivos conectados a la red (teléfonos inteligentes, tabletas, etc.) infectados y controlados remotamente, que se comportan como robots (Bots) o zombies, quedando incorporados a redes distribuidas, las cuales envían de forma masiva mensajes de correo Spam o código malicioso, con el objetivo de atacar otros sistemas o dejarlos fuera de servicio.

- Denegación de Servicios: tiene como objetivo inhabilitar el uso de un sistema o computadora, con el fin de bloquear el servicio para el que está destinado. Los servidores Web poseen la capacidad de resolver un

número determinado de peticiones o conexiones de usuarios de forma simultánea, en caso de superar ese número, comienzan a ralentizarse o incluso bloquearse y desconectarse de la red. Existen dos técnicas para este ataque: la denegación de servicio o DoS (Denial of Service) y la denegación de servicio distribuido o DDoS (Distributed Denial of Service); la diferencia entre ambos es el número de equipos de cómputo que realizan el ataque. En el primero, las peticiones masivas al servicio se realizan desde una misma máquina o dirección IP, consumiendo así los recursos que ofrece el servicio hasta que no tiene capacidad de respuesta y comienza a rechazar peticiones (denegar el servicio); en el segundo, las peticiones o conexiones se realizan empleando un gran número de computadoras o direcciones IP, todas al mismo tiempo y hacia el mismo servicio objeto del ataque, de forma general, las computadoras que lo realizan se encuentran infestadas, formando parte de una Botnet, y comportándose como zombis.

- Ataque MITM (Man In The Middle): conocido como "hombre en el medio", ocurre cuando una comunicación entre dos sistemas es interceptada por una entidad externa simulando una falsa identidad. En este sentido, el atacante tiene control total de la información que se intercambia, pudiendo manipularla a voluntad, sin que el emisor y el receptor lo perciban rápidamente. Es común que se realice empleando redes WI-FI públicas y abiertas, y es muy peligroso ya que se puede obtener información sensible de las víctimas, y es difícil identificarlo si no se poseen los mínimos conocimientos sobre el tema.

- Rootkit: es un tipo de Malware diseñado para infectar una PC, el cual permite instalar diferentes herramientas que dan acceso remoto al equipo de cómputo. Este Malware se oculta en la máquina, dentro del sistema operativo y sortea obstáculos como aplicaciones antimalware o algunos productos de seguridad. El Rootkit contiene diferentes herramientas maliciosas como un módulo para robar los números de tarjeta o cuentas bancarias, un Bot para ataques y otras funciones que pueden desactivar el Software de seguridad.

- SIM Swapping: es la duplicación del SIM de un número telefónico que permite a un atacante usurpe nuestra identidad, pudiendo autentificarse por medio de SMS en diversos servicios que usen la autentificación de dos pasos, incluyendo los servicios bancarios.

- SIM Jacker: es el envió de un mensaje malicioso al dispositivo destino, que si se abre el enlace adjunto, el dispositivo inteligente queda comprometido y se puede extraer toda la información del mismo incluyendo la ubicación.
- 0-Day (día cero): es una nueva vulnerabilidad para la cual no se han creado parches o revisiones, y que se emplea para llevar a cabo un ataque. El nombre se debe a que no existe ninguna revisión para mitigar el aprovechamiento de la(s) vulnerabilidad(es), estas pueden ser utilizadas para qué troyanos, Rootkits, virus, gusanos y otros Malwares se propaguen e infecten más equipos.
- IP Spoofing: el ciberdelincuente consigue falsear su dirección IP y hacerla pasar por una dirección válida en la cual confiamos, de este modo, consigue saltarse las restricciones y puede hacernos conectar a una Web maliciosa.
- Web Spoofing: consiste en la suplantación de una página Web real por otra falsa. La Web falsa es una copia del diseño original, llegando incluso a utilizar una URL muy similar para que demos nuestras credenciales de acceso.
- Email Spoofing: consiste en suplantar la dirección de correo de una persona o entidad de confianza para solicitarnos datos o que descarguemos archivos maliciosos.
- DNS Spoofing: a través de programas maliciosos específicos y aprovechándose de las vulnerabilidades en las medidas de protección, los atacantes consiguen infectar y acceder a nuestro Router suplantando la DNS (Domain Name System). Así cuando tratamos de acceder a una determinada Web desde el navegador, este nos lleva a otra Web elegida por el atacante.
- Sniffing: se trata de una técnica utilizada para escuchar todo lo que se transmite dentro de una red, de esta forma se monitorea el tráfico y se puede capturar información que viaja de forma no cifrada para analizarla y hacerse de nuestros datos.
- Ataque de inyección de SQL (Structured Query Language): es aprovechar una o más vulnerabilidades de servidores SQL que hace que se divulgue información confidencial de la base de datos que de otra manera

no haría al inyectar código SQL malicioso. Esto representa un riesgo enorme si la base de datos almacena información de identificación personal, como números de tarjetas de crédito, información personal y contraseñas.

- Baiting o Gancho: se sirve de algún medio físico como una unidad USB infectada que el atacante deja al alcance de los usuarios, para que cuando este lo introduzca en su equipo se infecte. También puede usar anuncios en Webs con la que promociona cursos o premios que nos inciten a compartir datos o descargar Software malicioso.

En general, para poder llevar a cabo alguno de estos ataques, los intrusos deben disponer de los medios técnicos, los conocimientos y las herramientas adecuadas, deben contar con una determinada motivación o finalidad, y se tiene que dar además una oportunidad que facilite el desarrollo del ataque, como podría ser un fallo en la seguridad del sistema informático elegido.

15.5 Dicen que si no Pagas por un Producto, Entonces el Producto Eres Tú.

¿Cuál es el modelo de negocio de empresas tecnológicas como Facebook? usar Google, Facebook, Messenger, Instagram y WhatsApp es completamente gratis. Y pese a ello, por ejemplo Facebook saca cada vez más dinero por usuario. Esto es posible gracias al uso que Facebook hace de nuestros datos. Teóricamente, y escándalos por fallos de seguridad al margen, Facebook no vende nuestros datos a terceros, sino que vende a terceros el acceso a nosotros gracias al uso de nuestros datos.

Así, las empresas tecnológicas (no importa si pagamos o no por un servicio o producto) van detectando nuestros gustos e intereses en base al dispositivo desde el que accedemos, las páginas que seguimos, nuestro historial de navegación y otros factores, y crea un perfil sobre cada uno de nosotros. Luego vende espacios de nuestro Feed a las empresas que busquen gente como nosotros (franja de edad determinada, localización, aficiones...). Y ya se está planteando formas de ir más allá mediante acuerdos con terceros, como uno con la banca para poder saber el dinero que tenemos en nuestra cuenta.

Esta red publicitaria que tan optimizada está merece el aplauso por su logro técnico, pero quizás no sea tan plausible si tenemos en cuenta ese rastreo

tan agresivo y la escasa preocupación de las empresas por la privacidad de sus usuarios.

Por otro lado, el uso generalizado de las redes sociales entraña algunos riesgos que, siguiendo recomendaciones básicas, se pueden evitar. Como cualquier comunidad frecuentada por miles de usuarios (o, como sucede a veces con las redes sociales, por miles de millones), se deben conocer los mecanismos de control y de seguridad para poder utilizarlos con fiabilidad para mantener nuestra privacidad y es por eso que el usuario tiene que ser especialmente cuidadoso con el uso que hace de la red social, a continuación daremos algunas recomendaciones:

- No necesitamos informar de todo y a todos, y menos con información sensible. Por ello, cuanta menos información pongamos, mejor.
- A menudo publicamos notas o mensajes en el muro de un amigo. Esto también es visible para otros usuarios de Facebook o de la red social que sea.
- Se dice que una foto vale más que mil palabras. Estamos dando información sobre nuestros hábitos, nuestros movimientos, a posibles atacantes.
- Hay que asegurarse de que nuestro contenido en las redes sociales sea visible solo para amigos y familiares.
- Google, Twitter, Facebook e Instagram usan el geoetiquetado mediante el GPS para ayudar a los usuarios a marcar la ubicación donde se hizo una foto, vídeo y ayudar a que el perfil sea más «social». Cualquier usuario puede interpretar fácilmente información como nuestro estado económico, estilo de vida, lugares frecuentes y la rutina diaria a través de los medios con etiquetas geográficas.
- En una red social lo normal es que cada usuario se identifique con su nombre y apellido real y que aporte datos personales, como si estudia o trabaja, con quién se relaciona o en qué ciudad vive.Esto hace que su exposición pública sea mucho mayor que antes, esto implica la pérdida del anonimato algo que antes era común y habitual en internet.
- En las redes sociales, una vez que se pulsa el botón de "publicar", esa información es enviada a los contactos del usuario. Eso significa que si

más adelante el usuario se arrepiente de lo dicho, publicado o mostrado y trata de borrarlo, solo conseguirá eliminarlo de su propio perfil pero no de internet.

- Quizás lo más importante de estos consejos para mantener la seguridad en redes sociales es el sentido común en lo que publicamos y comentamos en ellas.

Acuerdos de Privacidad la lectura de los acuerdos de privacidad[314] orientará al usuario sobre qué datos se comparten o no, y también se ofrece la opción de seleccionar o anular las opciones de privacidad, seguridad o administrativas escogidas para proteger la cuenta y el dispositivo.

Al registrarte en una cuenta de redes sociales, por defecto, toda la información anotada en un perfil se hace pública, lo que significa que cualquier persona puede acceder al contenido que hayas registrado en una cuenta. Sin embargo, las necesidades y preferencias de privacidad varían de persona a persona. Mientras que algunos usuarios prefieren tener una mayor exposición y así poder promocionar su contenido en redes sociales, otros prefieren incluir muy poca o ninguna información.

Para lograr una mayor protección del usuario y su información, es importante evaluar en qué medida la persona está dispuesta a incluir información personal en su perfil. Por consiguiente, ten en cuenta lo siguiente al:

- Seleccionar un nombre de usuario: el nombre de usuario es el "nombre digital" que una persona se asigna a sí misma o a su organización para ser identificada en línea. Si existe la preferencia de no ser fácilmente identificada en ninguna plataforma, pero poder continuar usando estas redes, la persona puede asignar y usar un seudónimo que puede estar relacionado o no con esa persona. Además, la persona puede cambiar su nombre de usuario en cualquier momento simplemente ingresando

[314] Si de verdad leyéramos los términos y condiciones de uso de plataformas Online seleccionadas (datos de abril 2020), tardaríamos (con una velocidad de 240 palabras por minuto) aproximadamente:

Microsoft 1:03:30 s (15,260 palabras), Spotify 35:48 s (8,600 palabras), Tik Tok 31:24 s (7,459 palabras), Apple 30:30 s (7,314 palabras), Zoom 30:12 s (7,243 palabras), Tinder 25:54 s (6,215 palabras), Uber 25:36 s (5,658 palabras), Twitter 25:30 s (5,633 palabras), LinkedIn 18:06 s (4,346 palabras), Facebook 17:12 s (4,132 palabras), Amazon 14:12 s (3,416 palabras), YouTube 13:42 s (3,308 palabras, Netflix 11:00 s (2,628 palabras), Instagram 9:42 s (2,451 palabras).

a la configuración de su (s) cuenta (s). El nombre de usuario no tiene que ser coherente en todas las redes sociales; estas pueden variar según las preferencias en cada una.

- Incluir una imagen en la cuenta: el usuario tiene la opción de personalizar una cuenta con la inclusión de una foto del perfil. Cuando un usuario prefiere no ser identificado, se sugiere elegir una imagen en la que no pueda ser identificado y cambiarla cuando sea necesario. Ten en cuenta que cuando se usa la misma imagen en todas las redes sociales, la simple búsqueda de imágenes puede llevar a otras cuentas.

- Incluir una ubicación: cuando se activan los servicios de ubicación en la plataforma de redes sociales, estos permiten a los usuarios rastrear el origen de cualquier actividad de medios en línea. Es importante tener en cuenta que una vez que se activa esta función, permanecerá activa hasta que se elija deshabilitarla en la configuración de privacidad. A pesar de que se permitía que esta característica estuviera activa en el pasado, las plataformas tienen la funcionalidad de deshabilitar la ubicación de cualquier contenido que se haya publicado en sus cuentas.

Sin embargo, aunque un usuario active o desactive la función de compartir la ubicación, potencialmente, la ubicación de un usuario podrá ser descubierta por el contenido que comparta o las imágenes que haya elegido para compartir.

15.6 Datos que Recopila Google

Desde el escándalo de Cambridge Analytica[315] en 2018, poco a poco todos nos hemos ido haciendo más conscientes de nuestra privacidad, y cómo las grandes empresas recopilan nuestros datos casi sin que nos demos cuenta. Ya no solo Facebook y las redes sociales, ya que hay otras empresas como Google que pueden recopilar muchos más datos sobre lo que hacemos cada día.

En parte, esto se debe a que es una empresa que ofrece una gran cantidad de servicios gratuitos que utilizamos casi sin pensar, y en otra, porque es la dueña de todo lo que buscamos en la red (si usamos sus productos y servicios).

[315] Se le acusa de tratar de influenciar los resultados de las elecciones presidenciales de 2016 en los Estados Unidos de Norteamérica.

Para hacerme una mejor idea de hasta dónde llega, podemos descargar todos nuestros datos y sí los revisamos es seguro que tendremos una desagradable sorpresa, que nos llevan a la conclusión de que Google sabe mucho más sobre nosotros que la propia Facebook, como por ejemplo, por dónde he viajado y qué aplicaciones utilizo y cuándo.

Algunos de estos datos los puedes encontrar fácilmente en algunas páginas de datos. Pero para otros, hemos utilizado la herramienta Google Takeout para descargar una copia de seguridad de todo lo que tienen sobre nosotros. Ya es bastante significativo que esta copia ocupe varios Gigas. Algo que debes saber es que puedes configurarlo para que muchos de los datos que Google tiene sobre ti se autodestruyan cuando haya pasado determinado tiempo.

Una cosa que hay que tener en cuenta es que Google no tiene por qué estar vendiendo o revisando minuciosamente estos datos, aunque tampoco hay manera de asegurarse de que no lo hagan con alguno. Sin embargo, en algunos casos nos parece excesivo incluso el simple hecho de que recopilen algunos de ellos. Para que sepas de lo que estamos hablando, aquí tienes una lista del tipo de datos personales que he visto que Google recopila.

Lo primero que llama la atención al mirar los datos de Google, es que sabe por dónde te has estado moviendo. Por ejemplo, tiene una función de cronología en la que puedes ver todos los movimientos que ha ido registrando sobre ti cuando tienes habilitado el historial de ubicaciones de Google Maps. Este suele estar activado por defecto en tus dispositivos, lo que quiere decir que si no cambias la configuración deliberadamente seguirá todos tus pasos a través de tu móvil, tableta o cualquier otro dispositivo con el que estés identificado con tu cuenta de Google.

Todos estos datos Google te los va presentando de forma ordenada y cronológica, mostrándote a qué hora has salido de un sitio, el medio de transporte por el que has ido y qué paradas has hecho. Sorprende la minuciosidad con la que Google lo registra todo. Además, en el caso de que te hayas ido de viaje sacando muchas fotos, Google también va a saber dónde y cuándo sacaste las fotografías con tu móvil, y todo te lo va a mostrar de forma ordenada viendo cómo hiciste el camino y dónde hiciste las fotos. Incluso distingue los trayectos que hiciste en coche y los que has hecho a pie. Esto lo hace utilizando los metadatos de tus fotos y cruzándolos con la información de ubicaciones que recopila.

Y más allá de esta herramienta, en la copia de seguridad de Google también puedes encontrar otras sorpresas. Por una parte, hay una carpeta con un historial de ubicaciones, en la que puedes ver las coordenadas por las que

te has movido en los últimos días y la manera en la que lo has hecho. Es como lo de Google Maps, pero con el código en bruto.

También guarda las actividades que hayas registrado utilizando Google Fit. Aquí puedes encontrar dos carpetas diferentes, una con todos los ejercicios que has hecho utilizando esta aplicación, con horas y coordenadas, y otra con un archivo diario en el que se puede ver todos los movimientos agregados en cada momento. Da igual que lleves años sin utilizar la aplicación guarda los datos obtenidos de aplicaciones de terceros al vincularlas con el servicio.

Google también tiene una sección Mi Actividad en la que recoge y almacena todas las búsquedas que haces en Internet, así como el contenido que consumes dentro de portales pertenecientes a Google. Aquí no solo vas a encontrar las búsquedas hechas a través del buscador, sino las búsquedas realizadas en Chrome y tu Android.

La verdad es que al mirar mi cronología no he podido evitar sentirme un poco incómodo, sobre todo porque todos los datos están perfectamente organizados y van acompañados de enlaces. De esta manera, puedes saber que has buscado determinados términos y volver a pulsar sobre ellos para ver los resultados de búsqueda, pero también puedes saber qué perfiles de Twitter o Facebook has visitado o las páginas a las que has entrado.

Además de esto, Google también recopila cuando utilizas otras plataformas como Stadia. Incluso si simplemente pulsas en el botón de Discover de la aplicación de Google para ver los temas que la propia Google considera importantes para ti, va recopilando cuáles son los temas sobre los que te ha mostrado información cada vez.

Es más, dentro de algunas páginas también te dice en qué secciones has navegado. Esto se vuelve un poco más preocupante con el tema de los foros, ya que te dice el título de cada hilo que has visitado dentro de un mismo foro, e incluso te ofrece enlaces para volver a él.

La Web también registra cuántas veces utilizas cada aplicación móvil y lo que es peor, todos estos datos siguen apareciendo aunque borres el historial de Chrome. He probado borrar las páginas a las que he entrado hoy con el navegador, y estas seguían apareciendo en la página de My Activity.

¿Y qué implicaciones tiene esto? Pues no sólo que Google sabe exactamente todo lo que haces en Internet si utilizas sus productos, sino que cualquiera que se ponga frente a tu computadora, tableta o teléfono inteligente (si tiene acceso a tu cuenta) va a poder saberlo, ya que al entrar a My Activity Google no pide que confirme mi identidad. Así que cualquiera puede con relativa facilidad tener acceso a esta información.

En resumen Google tiene acceso a:

- Tu nombre, tu dirección, tu edad, tu correo electrónico. Tu modelo de teléfono, tu proveedor de telefonía celular, tu plan y tu consumo telefónico y de internet.
- Las palabras que usas con más frecuencia dentro de tus correos electrónicos. Todos los correos que hayas escrito o recibido, incluido Spam. Los nombres de tus contactos y sus direcciones y teléfonos.
- Las fotografías que tomas con tu teléfono Android, aunque las hayas borrado y aunque no las subas nunca a ninguna red social. Los sitios a los que vas, dentro y fuera del país; la fecha en la que fuiste y la ruta que tomaste. Qué tan rápido llegaste. La tarjeta de crédito o débito que usas para pagar.
- Todos los sitios de internet que has visitado en Google, con qué frecuencia y lo que viste dentro de cada uno. En qué idioma buscas. A qué hora navegas. Con quién has hablado vía Hangouts. Qué videos te gustan. Qué música oyes, etc.

Para tratar de minimizar nuestra huella digital, si soy usuario de los servicios de Google puedo desactivar todos los servicios de rastreo a los que me da opción en su página de "Administrar tu cuenta de Google", no es notoria la degradación del servicio por dichas desactivaciones. Pero esto no impide que Google y servicios de terceros recolecten y transmitan nuestros datos, solo no se almacenarán en nuestra cuenta, ni tendremos acceso a los mismos.

También puedo prescindir de los servicios de Google usando otras alternativas no tan invasivas como describimos más adelante en este texto.

15.7 ¿Cómo me Protejo?

Algunas normas básicas de ciberseguridad para nuestra seguridad, tener privacidad y evitar la vigilancia son:

- Usar contraseñas largas, robustas (incorporar mayúsculas, minúsculas, números y símbolos) y diferentes para cada servicio que lo solicite, para facilitar el manejo de las contraseñas es recomendable el uso de gestores de contraseñas.

- Cifrar Dispositivos, Discos y Unidades de Respaldo para mantener la confidencialidad de nuestra información.
- Mantener actualizado el sistema operativo y las aplicaciones (sólo instalar las necesarias) de nuestros dispositivos además de usar mecanismos que coadyuven en la seguridad como cortafuegos, antivirus, entre otras aplicaciones de seguridad.
- Generar respaldos periódicos -compactados y cifrados- de nuestra información y garantizar que es posible restituir nuestros datos de dichos respaldos (una buena estrategia de respaldo es la siguiente: mantener tres copias de cualquier fichero importante -una principal y dos respaldos-, mantener los ficheros en dos tipos distintos de almacenamiento para protegerlos ante distintos riesgos y almacenar una copia de seguridad fuera de nuestra casa u oficina).
- Para ciertas actividades en la red o el uso de programas poco confiables es recomendable el uso de máquinas virtuales que limitan los posibles daños por programas maliciosos.
- Al navegar en internet es necesario hacerlo de forma segura para no exponernos de forma innecesaria, mediante un uso seguro y aceptable de las herramientas en la Nube.
- Conocer las medidas mínimas de protección para una navegación segura en internet.
- Conocer las técnicas usadas en ataques de inteligencia social, para no ser víctimas de la ciberdelincuencia.
- Al hacer uso de videoconferencias conocer las políticas de privacidad y las medidas de seguridad para no exponernos a ciberacoso y pérdida de datos.
- Hacer un uso correcto de los dispositivos personales cuando estos son usados para el trabajo.
- Uso de escritorios remotos y virtuales como una forma de mitigar los riesgos cuando se trabaja de forma remota.

Entre otras tantas cosas que el usuario actual de las Tecnologías de la Información y la Comunicación (TIC) debe dominar. Además, debemos conocer algunas recomendaciones útiles para reducir nuestra huella en internet, como:

- No compartir nuestro correo electrónico y número telefónico, pese a que es habitual utilizarlos para registrarnos en sitios Web, si los compartimos libremente por internet nos exponemos a ser víctimas de Spam, Phishing y todo tipo de ciberataques basados en ingeniería social.

- El uso de cuentas de correo alternativas nos permite registrarnos en diferentes sitios Web sin utilizar datos y/o cuentas personales o de trabajo.

- Usar cuentas de correo seguro, anónimo y bidireccional como el servicio de Tutanota, Mailfence o ProtonMail. O usar el modo confidencial -en el cual podemos establecer fecha de vencimiento y contraseña para cada correo- al enviar correos desde cuentas Google.

- Acceder a las opciones del navegador para eliminar cada cierto tiempo la información almacenada en formas de Cookies, Caché o el historial de navegación.

- El uso de modo privado o incógnito[316] en el navegador nos permite no almacenar información sobre las páginas Web visitadas ni se guardan las Cookies, ya que se eliminarán al salir del navegador.

- Evitar revelar información sensible en redes sociales sobre nosotros, familia y conocidos, en medida de lo posible debemos vigilar lo que publicamos, quién puede verlo y configurar las opciones de privacidad.

[316] En Chrome y otros navegadores Web, el modo incógnito -del cual Google sostiene que este modo- está diseñado para evitar el almacenamiento local de datos, y realmente no protege nuestra privacidad, aunque otros usuarios en el mismo dispositivo no verán la actividad en modo incógnito, los sitios Web y servicios, incluidos los de Google, aún pueden recopilar datos. La actividad, como descargas, favoritos y elementos de la lista de reproducción, se almacenará.

Cabe mencionar que ningún navegador ofrece el 100% de privacidad ni anonimato al usuario, lo que si, es que entre los diferentes navegadores existentes, podremos encontrar navegador web (como por ejemplo Brave) que ofrecen capas adicionales de protección de los datos del usuario, pero esto no los vuelve 100% eficientes en ello, pues incluso navegadores como Tor (para la Dark Web) tiene sus fallos.

- El uso de Webs seguras (https y certificados digitales) aseguran que la información que intercambiamos con ellas, como datos bancarios o contraseñas viajen de forma cifrada.
- Usar navegadores especializados (Tor) y lugares de búsqueda de información (DuckDuckGo) que nos permitan una navegación segura al no guardar lo que buscamos, ni los sitios donde accedemos.
- El uso del GPS en nuestros dispositivos móviles es una gran herramienta en nuestra vida digital, pero debemos tenerlo activado sólo cuando sea necesario, ya que muchas aplicaciones comparte nuestra ubicación en tiempo real sin nuestro conocimiento y alguien puede conocer donde vivimos, los lugares que frecuentamos o cuando no estamos en casa.
- Al tomar fotografías o vídeos desactivar el guardado de datos *Exif* (Exchangeable Image File) también conocidos como metadatos: datos sobre la cámara, datos sobre la fotografía y datos sobre el origen como ubicación por GPS, etc.
- Nunca tomar vídeos o fotos comprometedoras, de carácter íntimo o sexual y menos publicarlas, ya que suponen una gran amenaza a nuestra seguridad y pueden tener consecuencias muy graves, como la exposición pública, extorsión o el ciberacoso.
- No compartir fotos, vídeos o audios de menores.
- No debemos compartir vídeos, fotos o audios de terceros sin su aprobación, en especial los que contienen conversaciones privadas que difundan datos personales o información que podría considerarse como revelación de secretos y que la otra persona preferiría no difundir.
- Al emitir opiniones, quejas o comentarios subidos de tono u ofensivos en medios electrónicos nos expone a ser atacados o censurados y en ciertos casos podrían ameritar acciones legales.
- No publicar documentos personales en forma de fotos, vídeos o archivos (*.Docx*, *.PDF*, etc) ya que con su revelación nos expone a una suplantación de identidad y al uso de nuestros datos de forma fraudulenta.

- Cumplir las normas mínimas de protección de dispositivos personales en redes corporativas.
- Hacer uso correcto de escritorios remotos y virtuales en equipos personales para el trabajo remoto.

16 Apéndice C: Distribuciones Seguras, Penetración, Inmutables y IOT

Para muchos, Linux y Mac OS son dos sistemas operativos más seguros que Windows de Microsoft, pero con todo, hay algunas distribuciones especializadas de Linux que satisfacen las necesidades de temas relacionados con la seguridad, pruebas de penetración, análisis forense, auditorías de seguridad, etc.

Las distribuciones seguras intentan preservar la privacidad y el anonimato, ayudan a utilizar internet de forma anónima y evitar la censura en prácticamente cualquier lugar y cualquier equipo de cómputo, pero sin dejar rastro a menos que lo solicites explícitamente.

Las distribuciones para pruebas de penetración ofrecen herramientas para penetración, análisis forense y auditorías de seguridad en donde se realizan distintos tipos de tareas que identifican, en una infraestructura objetivo, las vulnerabilidades que podrían explotarse y los daños que podría causar un atacante. En otras palabras, se realiza un proceso de Hacking ético para identificar qué incidentes podrían ocurrir antes de que sucedan y, posteriormente, reparar o mejorar el sistema, de tal forma que se eviten estos ataques.

Además, las distribuciones anónimas ofrecen niveles adicionales de privacidad y seguridad y se complementan con las distribuciones para pruebas de penetración que ofrecen herramientas para penetración y auditorías de seguridad (mediante el uso de tecnologías como TOR[317], Sandbox[318], Fire-

[317] Tor es una red abierta y distribuida que te ayuda a defenderte de una forma de vigilancia en la red que amenaza tu libertad y privacidad, tus actividades comerciales confidenciales y relaciones, además de la seguridad gubernamental. Además, te protege redirigiendo tus comunicaciones alrededor de una red distribuida de retransmisores realizados por voluntarios alrededor del mundo: lo cual previene que alguien observe tus comunicaciones a partir de los sitios que visitas, también evita que los sitios en que navegas obtengan tu ubicación física.

[318] Sandbox es un mecanismo para ejecutar programas con seguridad y de manera separada. A menudo se utiliza para ejecutar código nuevo, o Software de dudosa confiabilidad proveniente de terceros. Ese entorno aislado permite controlar de cerca los recursos proporcionados a los programas cliente a ejecutarse, tales como espacio temporal en discos y memoria. Habitualmente se restringen las capacidades de acceso a redes, la habilidad de inspeccionar la máquina anfitrión y dispositivos de entrada entre otros. En este sentido, el aislamiento de procesos es un ejemplo específico de virtualización.

wall[319], herramientas criptográficas[320], etc.). De ambos tipos de Softwares hay varias alternativas diferentes, tanto comerciales como de Software libre, por lo que decidirse por una u otra, en ocasiones puede ser una tarea un tanto complicada. Es por ello que aquí listamos algunas de las distribuciones de Linux más usadas en la actualidad, apartados con los que cada vez debemos prestar más atención.

Las distribuciones inmutables garantizan que el núcleo del sistema operativo permanezca sin cambios. El sistema de archivos raíz de una distribución inmutable sigue siendo de solo lectura, lo que permite permanecer igual en varias instancias (por supuesto, puedes cambiar las cosas si así lo deseas, sin embargo, la capacidad permanece desactivada de forma predeterminada).

Por otro lado, existen versiones de sistemas operativos para el Internet de las cosas (IOT), estos sistemas operativos están diseñado específicamente para funcionar dentro de las limitaciones particulares de los dispositivos de IoT, que generalmente están limitados en tamaño de memoria, potencia de procesamiento y capacidad, y están diseñados para permitir una transferencia rápida de datos a través de Internet. Hay varios sistemas operativos (principalmente basados en Linux) que puedes usar para IoT, pero no te permitirán aprovechar al máximo tu configuración y esa es la razón por la que existen distribuciones centradas en IoT.

Algunas de las distribuciones seguras, para penetración e inmutables son sistemas operativos Live[321] diseñados para ser usados desde un CD, DVD,

[319] Un cortafuegos es una parte de un sistema o una red que está diseñada para bloquear el acceso no autorizado, permitiendo al mismo tiempo comunicaciones autorizadas. Se trata de un dispositivo o conjunto de dispositivos configurados para permitir, limitar, cifrar o descifrar el tráfico entre los diferentes ámbitos sobre la base de un conjunto de normas y otros criterios. Los cortafuegos se utilizan con frecuencia para evitar que los usuarios de internet no autorizados tengan acceso a redes privadas, especialmente intranets.

[320] El surgimiento de redes de comunicación, en particular de internet, ha abierto nuevas posibilidades para el intercambio de información. Al mismo tiempo, son cada vez mayores las amenazas a la seguridad de la información que se transmite. Es necesario entonces, crear diferentes mecanismos, dirigidos a garantizar la confidencialidad y autenticidad de los documentos electrónicos, todo ello es parte de la Criptografía.

[321] Un Live CD/DVD o USB, más genéricamente Live Distro, es un sistema operativo almacenado en un medio extraíble, tradicionalmente un CD/DVD o USB (de ahí sus nombres), que puede ejecutarse directamente en una computadora.

En la historia más reciente de Linux, las llamadas distribuciones Live Distro se han vuelto muy populares porque le permiten probar una distribución de Linux sin siquiera instalarla en el equipo. Esto es excelente porque no tiene todas las molestias de volver a particionar el disco o instalarlo sobre su sistema operativo (Windows/Mac OS). Sim-

memoria USB o máquina virtual independientemente del sistema operativo original de la computadora.

16.1 Distribuciones de GNU/Linux «Seguras»

Tails (https://tails.boum.org/)

Para muchos esta es la primera opción a la hora de buscar una solución de seguridad en Linux. También conocida como «The Amnesic Incognito Live System», esta es una distribución basada en Debian GNU/Linux . Es un proyecto de código abierto que se publicó por primera vez hace 8 años y que redirige todo el tráfico Web a través de Tor logrando la privacidad a través del anonimato. Además, almacena todo en la RAM y evita el uso del disco duro, por lo que borra todo una vez se apaga. La imagen tiene un tamaño menor de 1.2 GB y necesita al menos 2 GB de RAM en un equipo de 64 bits, se puede usar en formato Live, como máquina Virtual o bien instalarse en una USB, DVD o en el disco duro del equipo.

Septor (https://septor.sourceforge.io)

Es un sistema operativo que proporciona a los usuarios un entorno informático perfecto para navegar por internet de forma anónima. Septor pro-

plemente puede colocar el CD/DVD o USB para una distribución en vivo e iniciar la computadora desde ahí. Por lo general, obtiene la mayor parte de la funcionalidad principal de la distribución, por lo que realmente puede evaluar si la distribución es para usted antes de elegir instalarla de verdad.

Normalmente, una versión Live viene acompañado de un par de aplicaciones. Algunos Live CD/DVD o USB incluyen una herramienta que permite instalarlos en el disco duro. Otra característica es que por lo general no se efectúan cambios en la computadora utilizada.

Para usar una versión Live es necesario obtener uno (muchos de ellos distribuyen libremente una imagen ISO que puede bajarse de Internet y grabarse en disco/USB) y configurar la computadora para que arranque desde la unidad lectora, reiniciando luego la computadora con el disco en la lectora o USB, con lo que el que el sistema Live se iniciará manualmente.

Uno de los mayores inconvenientes de este sistema es el mal uso de una gran cantidad de memoria RAM, una parte para su uso habitual y otra para funcionar como el disco virtual del sistema. En el arranque, se le pueden dar distintos parámetros para adaptar el sistema a la computadora, como la resolución de pantalla o para activar o desactivar la búsqueda automática de determinado Hardware.

Otro inconveniente es el rendimiento de la Live Distro, pues la velocidad de transferencia de las unidades lectoras CD/DVD o USB es muy inferior a la de los discos duros. Una vez instalada en la computadora se apreciará la velocidad real de la distribución.

porciona a los usuarios una distribución estable y confiable que se basa en Debian GNU/Linux y funciona en una amplia gama de computadoras, usa un escritorio KDE Plasma personalizado y tecnologías Tor. Esta distribución es similar a Tails y se puede usar en vivo directamente desde una unidad Flash USB, máquina virtual o instalarla localmente.

Whonix (https://www.whonix.org/)

Es una distribución que se basa en Debian GNU/Linux y consta de dos máquinas virtuales, una es Tor Gateway que se ejecuta en Debian GNU/Linux, mientras que la otra es una Workstation. Whonix se instala en un sistema operativo Host proporcionado por el usuario que puede ser Linux, Windows, MacOS o Qubes OS. Así al utilizar la red abierta y distribuida de transmisión de Tor, Whonix echa abajo las posibilidades de vigilancia de la Red. Además, y por motivos de seguridad, hace todo lo posible para ocultar nuestra dirección IP real.

Qubes OS (https://www.qubes-os.org/)

Se conoce como uno de los sistemas operativos más seguros del mundo y se basa en realizar la virtualización mediante el «hipervisor Xen» -un hipervisor imita el Hardware y permite ejecutar varias máquinas virtuales simultáneamente-. El entorno de usuario de Qubes OS podría ser Fedora, Debian, Whoix o Windows y, al igual que Tails. Así mismo utiliza diferentes dominios para separar los niveles de confianza, por ejemplo, un dominio de trabajo, un dominio para el ocio, etc.; los cuales se ejecutan en diferentes máquinas virtuales, esta versión requiere un mínimo de 16 GB de RAM.

Subgraph OS (https://subgraph.com/)

Nos encontramos con un sistema operativo seguro basado en Debian GNU/Linux que promete proporcionar una experiencia digital anónima. Ha sido diseñado para evitar diferentes ataques de Malware, es capaz de ser una plataforma de comunicación segura además de proporcionar una interfaz de usuario muy sencilla.

Discreete Linux (https://www.privacy-cd.org/)

En este caso nos encontramos con un proyecto de Software libre que puede ser utilizado por cualquier persona que desee llevar una vida digital anónima también basado en Debian GNU/Linux. Además, protege a sus usuarios

contra los ataques de vigilancia accionados por troyanos. Es una de las alternativas más adecuadas para los usuarios que no tienen un conocimiento muy profundo de estos sistemas pero que consideran la seguridad en internet como su principal preocupación. Hace uso de cifrados y entornos aislados para proporcionar un entorno de trabajo seguro. Así mismo no utiliza los discos duros internos del equipo, ya que almacena todos sus datos en la memoria RAM o en una unidad de disco USB externa.

Kodachi (https://www.digi77.com/linux-kodachi/)
Es un sistema operativo centrado en la seguridad y basado en Debian GNU/Linux cuyo objetivo es proporcionar una experiencia informática segura. Ponerlo en marcha es muy sencillo y no necesita demasiado tiempo, ya que permite la opción de arrancar desde el Hardware del PC, o desde una unidad USB externa para mayor seguridad. Hace uso de elementos tales como una conexión VPN activa, el servicio TOR y DNScrypt con el que garantiza un buen nivel de privacidad. Además, todo el sistema operativo se ejecuta desde la memoria RAM del equipo, por lo que después de apagado no queda ningún rastro de actividad.

Tens (https://www.spi.dod.mil/lipose.htm)
También conocido como Trusted End Node Security, este sistema es distribuido y desarrollado por el Departamento de Defensa de los Estados Unidos. Se basa en Arch Linux y puede ejecutarse en cualquier equipo con tecnología Intel. Sólo arranca desde la RAM y viene cargado con un asistente de cifrado, un Software de cifrado simple y potente para la protección de nuestra información confidencial.

Tin Hat (https://sourceforge.net/projects/tinhat/)
Esta propuesta es una derivación de Gentoo y es un sistema operativo seguro que se ejecuta en su totalidad en la RAM del equipo, por lo que no monta ningún sistema de archivos directamente en el dispositivo de arranque, evitando así la posibilidad de dejar expuesto cualquier dato. Como era de esperar, podremos arrancarlo desde un CD o desde una unidad flash USB. Puede ejecutarse tanto en arquitecturas de Hardware de 32 como de 64 bits y es recomendable tener conocimientos previos de Gentoo Linux.

IprediaOS (https://www.ipredia.org/os/)

Para empezar diremos que I2P es una capa de comunicación P2P anónima que se crea utilizando herramientas de código abierto, algo en lo que se basa IprediaOS, ya que orienta todo su tráfico a través de I2P y se asegura de que toda su actividad Online no pueda ser interceptada por terceros. Así hace uso de múltiples capas de cifrado y cabe mencionar que la red I2P es una red dinámica y distribuida.

Alpine Linux (https://alpinelinux.org/)
Es una distribución diseñada principalmente para los usuarios avanzados que valoran la seguridad, la eficiencia de recursos y la simplicidad. En un principio parte como bifurcación del proyecto *LEAF* aunque, a diferencia de este, Alpine mejora las características de seguridad y cuenta con un Kernel más actual. Su funcionamiento se centra en la privacidad, por lo que utiliza su propio sistema de gestión de paquetes.

Openwall (https://www.openwall.com/Owl/)
Es una pequeña distribución de Linux con seguridad mejorada (SELinux) para servidores, dispositivos y dispositivos virtuales. A diferencia de otras distribuciones, el uso de SELinux por parte de Openwall evita que se incorpore Software vulnerable a la distribución, en lugar de depender de parches para remediar vulnerabilidades de seguridad conocidas o características diseñadas para disminuir el impacto de los errores de seguridad. Mediante el uso del marco *SELinux*, Openwall eclipsa a la mayoría de sus contrapartes más grandes en este sentido.

16.2 Distribuciones de GNU/Linux «Para Penetración»

Realizar pruebas de penetración, análisis forense y auditorías de seguridad resulta ser una tarea compleja e involucra un proceso en donde se realizan distintos tipos de tareas que identifican, en una infraestructura objetivo, las vulnerabilidades que podrían explotarse y los daños que podría causar un atacante. En otras palabras, se realiza un proceso de Hacking ético para identificar qué incidentes podrían ocurrir antes de que sucedan y, posteriormente, reparar o mejorar el sistema, de tal forma que se eviten estos ataques.

Para realizar una prueba de penetración de forma profesional, es necesario sumar a los conocimientos de Hacking ético, otros aspectos fundamentales como: programación, metodologías, documentación, entre otros. No

obstante, esos aprendizajes suelen venir una vez que se conocen y se saben utilizar muchas herramientas que son parte del proceso de pruebas de penetración. Las siguientes herramientas se deben conocer, instalar y probar para dar los primeros pasos en este "arte".

Kali Linux (https://www.kali.org)
Es una distribución para pruebas de penetración estándar de la industria. Es una de las distribuciones más populares entre Pentesters, Hackers éticos e investigadores de seguridad en todo el mundo y contiene cientos de herramientas para el trabajo forense, esta distribución es basada en Debian GNU/Linux.

Parrot OS (https://parrotlinux.org)
Puede verse como un laboratorio totalmente portátil para una amplia gama de operaciones de seguridad cibernética, desde pruebas de penetración hasta ingeniería inversa y análisis forense digital, pero esta distribución basada en Debian GNU/Linux también incluye todo lo que necesita para proteger sus datos y desarrollar su propio Software.

BlackArch Linux (https://blackarch.org)
Esta popular distribución de pruebas de penetración proviene de Arch Linux y contiene más de 2,400 herramientas de penetración y análisis forense diferentes, lo que le permite usar lo que necesite sin tener que descargar nuevas herramientas.

BackBox (https://www.backbox.org/)
Es una distribución de Linux basada en Ubuntu destinada a ayudar a Hackers éticos y probadores de penetración en evaluaciones de seguridad. BackBox OS está diseñado con el objetivo de ser más rápido, fácil de operar y tener un entorno de escritorio mínimo. La ventaja clave de BackBox es que sus propios repositorios de software se actualizan a intervalos regulares para mantener la distribución estable y popular para las operaciones del mundo real.

DEFT Linux (http://www.linuxandubuntu.com/home/deft-linux-a-linux-distribution-for-computer-forensics)

DEFT (Digital Evidence and Forensics Toolkit) se basa en GNU Linux y DART (Digital Advanced Response Toolkit), un sistema forense que comprende algunas de las mejores herramientas para la respuesta forense y de incidentes. DEFT Linux está especialmente diseñado para llevar a cabo tareas forenses y se ejecuta en vivo en los sistemas sin alterar el disco duro o cualquier otro medio de almacenamiento. Se compone de más de 100 herramientas forenses y de piratería de alta calificación.

Network Security Toolkit (https://www.networksecuritytoolkit.org)
El Network Security Toolkit (NST), basado en Fedora, es una versión Live que consta de las 125 principales herramientas de seguridad de código abierto proporcionadas por *insecure.org* para validar la seguridad de la red, pruebas de penetración, diagnósticos de red y monitoreo del día. El objetivo principal detrás del desarrollo de NST es proporcionar a los administradores de red/sistemas un conjunto combinado de herramientas de seguridad de código abierto para llevar a cabo operaciones como análisis de tráfico de red, detección de intrusos, escaneo de red y parches de seguridad.

Gnoppix (https://blog.desdelinux.net/gnoppix/)
Gnoppix es una distribución basada en Kali Linux Rolling diseñada para pruebas de penetración e ingeniería inversa con foco en Aplicaciones Web y obtención de acceso. Está optimizada para proteger tus derechos digitales. Pero, aunque esta centrada en la seguridad, también puede utilizarse como un escritorio normal.

16.3 Distribuciones de GNU/Linux «Inmutables»

Una distribución inmutable garantiza que el núcleo del sistema operativo permanezca sin cambios. El sistema de archivos raíz de una distribución inmutable sigue siendo de solo lectura, lo que permite permanecer igual en varias instancias. Por supuesto, puedes cambiar las cosas si así lo deseas. Sin embargo, la capacidad permanece desactivada de forma predeterminada.

¿Cómo es útil? Tradicionalmente, existían distribuciones inmutables para permitir pruebas más sencillas y desarrollo de Software basado en contenedores. Además, la inmutabilidad le brinda mayor seguridad y actualizaciones confiables para su sistema operativo. En aquel entonces, el enfoque

en dichas funciones se limitaba a las distribuciones dirigidas a profesionales y desarrolladores. Ahora se está incorporando para usuarios de escritorio diarios.

Distrobox (https://github.com/89luca89/distrobox)
esta herramienta permite instalar y ejecutar rápidamente cualquier distribución de Linux en un contenedor y garantizar su integración con el sistema principal. El proyecto proporciona un complemento sobre Docker, Podman o Lilipod, y se distingue por la máxima simplificación del trabajo y la integración del entorno de ejecución con el resto del sistema. Para crear un entorno con una distribución diferente, basta con ejecutar un solo comando distrobox-create, sin pensar en las sutilezas.

RlxOS (https://rlxos.dev/)
se enorgullece de haber sido creado desde cero como una distribución de Linux independiente para tener un mejor control sobre el núcleo y las partes funcionales. Al ser inmutable, sigue un enfoque de lanzamiento continuo para que los usuarios no necesiten reinstalarlo cada vez que haya una actualización importante. Las características clave incluyen: Inmutabilidad, Aprovecha Ostree, Se centra en la privacidad, Soporte nativo Flatpak

carbonOS (https://carbon.sh/)
carbonOS es una distribución independiente de Linux. Se centra en proporcionar una experiencia de usuario perfecta con tecnología sólida en su núcleo. Se necesita un enfoque que priorice Flatpak y contenedores. carbonOS también tiene como objetivo proporcionar actualizaciones seguras del sistema y arranque verificado como algunas características que no ofrecen todas las distribuciones atómicas. Además de sus características únicas, también quiere centrarse en brindar una excelente experiencia de escritorio GNOME.

Silverblue (https://fedoraproject.org/silverblue/)
Silverblue es una variante de Fedora Workstation con inmutabilidad. Es una de las distribuciones inmutables más populares que existen. La interfaz de usuario y la experiencia permanecen sin cambios con respecto a una versión típica de Fedora Workstation. Siempre que tenga una nueva versión de Fedora, espere también una nueva versión de Silverblue. Fedora Silverblue tiene como objetivo ofrecer una experiencia estable que sea útil

para realizar pruebas y desarrollar software basado en contenedores. Siempre puedes volver a la versión anterior del sistema operativo si algo sale mal después de una actualización.

Flatcar Container Linux (https://github.com/flatcar/Flatcar)
Flatcar es una distribución de Linux creada por la comunidad y adaptada a cargas de trabajo de contenedores, como su nombre indica. Obtiene una imagen mínima del sistema operativo que incluye solo las herramientas necesarias para ejecutar contenedores, sin administrador de paquetes y sin problemas de configuración. Si desea tener una infraestructura confiable para sus contenedores, Flatcar puede ser una buena opción que sea escalable, segura y simple al mismo tiempo. Explore más sobre esto en su página de GitHub.

NixOS (https://nixos.org/)
NixOS es una de las distribuciones de Linux más avanzadas disponibles. Pero si desea inmutabilidad y un montón de ventajas como recuperación sencilla, administrador de paquetes sólido, etc., NixOS debería ser una excelente elección. No se preocupe, si no conoce NixOS, puede explorar nuestra serie NixOS para aprender y configurarlo.

GUIX (https://guix.gnu.org/)
GUIX es similar a NixOS (más o menos) y está diseñado para usuarios avanzados que desean actualizaciones confiables y un buen control sobre sus sistemas. Si es un nuevo usuario de Linux, no debe esperar que sea su controlador diario. Por lo tanto, es posible que desees consultar su documentación para explorarla y comenzar.

openSUSE MicroOS (https://microos.opensuse.org/)
openSUSE MicroOS está diseñado para servidores donde es necesario implementar contenedores o trabajar con flujos de trabajo automatizados. Se basa en actualizaciones transaccionales que utilizan btrfs con instantáneas, lo que ayuda a guardar el historial del sistema de archivos sin ocupar mucho espacio de almacenamiento. En general, MicroOS es una opción escalable, confiable y segura para los usuarios de servidores.

Vanilla OS (https://vanillaos.org/)
Vanilla OS es un participante bastante nuevo en el espacio de la inmutabilidad. Sin embargo, logró causar sensación con su lanzamiento y luego cambió a una base Debian, abandonando Ubuntu justo después de su primer lanzamiento estable. Su objetivo es proporcionar una experiencia de escritorio fácil de usar con confiabilidad y características inmutables.

Bottlerocket (https://aws.amazon.com/bottlerocket/)
Bottlerocket es un sistema operativo de código abierto basado en Linux creado por Amazon Web Services para ejecutar contenedores en su plataforma. A diferencia de otras opciones, su uso se limita a AWS. Garantiza que los clientes que utilizan los servicios de AWS tengan una sobrecarga de mantenimiento mínima y puedan automatizar sus flujos de trabajo sin problemas. Solo puede utilizarlo como imagen de máquina de Amazon (AMI) cuando crea una Amazon Elastic Compute Cloud (EC2).

blendOS (https://blendos.co/)
blendOS es una distribución interesante en desarrollo que tiene como objetivo ofrecer todo lo bueno de otras distribuciones. En otras palabras, puede instalar cualquier paquete en la distribución (RPM, DEB, etc.) mientras obtiene la inmutabilidad y confiabilidad de actualización que cabría esperar.

Talos Linux (https://www.talos.dev/)
Talos Linux es otra distribución de Linux única, diseñada para Kubernetes. Talos Linux es una opción intrigante para los usuarios/desarrolladores de la nube. Es una opción segura, inmutable y mínima que admite plataformas en la nube, bare metal y plataformas de virtualización. También puedes iniciar fácilmente un clúster de Talos dentro de Docker. El sistema operativo se ejecuta en la memoria desde un SquashFS, lo que deja todo el disco primario a Kubernetes.

Endless OS (https://www.endlessos.org/os)
Endless OS es una distribución de Linux basada en Debian. A diferencia de cualquier otra distribución basada en Debian (por ejemplo, Ubuntu), Endless OS presenta un diseño robusto con inmutabilidad en su núcleo para garantizar que la actualización de un paquete no dañe el sistema.

16.4 Distribuciones de GNU/Linux para el «Internet de las Cosas»

Un sistema operativo de Internet de las cosas (IOT) es cualquier sistema operativo diseñado específicamente para funcionar dentro de las limitaciones particulares de los dispositivos de IoT, que generalmente están limitados en tamaño de memoria, potencia de procesamiento y capacidad, y están diseñados para permitir una transferencia rápida de datos a través de Internet.

Hay varios sistemas operativos (principalmente basados en Linux) que puedes usar para IoT, pero no te permitirán aprovechar al máximo tu configuración y esa es la razón por la que existen distribuciones centradas en IoT.

Zephyr (https://www.zephyrproject.org/)

Zephyr es un sistema operativo pequeño, escalable, de código abierto y en tiempo real (RTOS) para dispositivos conectados, que proporciona modularidad que permite a los desarrolladores optimizar el sistema para un uso específico. Admite múltiples arquitecturas y ofrece funciones como Bluetooth, LoRa y NFC. Está diseñado para ser fácil de usar y eficiente, con una pequeña huella de memoria y bajo consumo de energía. También incluye una serie de características que lo hacen adecuado para dispositivos IoT, como soporte para redes, seguridad y administración de energía.

Ubuntu Core (https://ubuntu.com/core)

Ubuntu Core es una versión robusta de la distribución más popular de Linux, Ubuntu, diseñada especialmente para grandes implementaciones de contenedores y dispositivos de Internet de las cosas. Fue creado por Canonical para utilizar el mismo kernel, Software del sistema y bibliotecas que Ubuntu, pero en una escala mucho menor, y se utiliza para alimentar robots, puertas de enlace, señales digitales, etc. Está diseñado para proporcionar a los usuarios un Linux integrado seguro para dispositivos IoT. Todos sus aspectos se verifican para mantener paquetes inmutables y firmas digitales persistentes. También es mínimo y está preparado para la empresa.

RIOT (https://www.riot-os.org/)

RIOT es un sistema operativo gratuito, amigable y de código abierto diseñado para trabajar con dispositivos IoT con el objetivo de implementar

todos los estándares abiertos relevantes que admitan conexiones IoT seguras, duraderas y respetuosas con la privacidad. Las características de RIOT incluyen un tamaño mínimo de RAM y ROM de ~1,5 kB y ~5 kB, soporte completo para C y C++, subprocesos múltiples, modularidad y MCU sin MMU.

OS Fuchsia (https://fuchsia.dev/)
OS Fuchsia es un sistema operativo en tiempo real con capacidad de código abierto creado por Google para dispositivos de Internet de las cosas. A diferencia de dos de los productos más queridos de Google, Chrome y Android, que se basan en el Kernel de Linux, Fuchsia OS se basa en el kernel Zircon. Se entrega con Node.js, que permite la compatibilidad con JavaScript y se espera que pueda ejecutarse en dispositivos AMD, así como en teléfonos y tabletas con capacidad para ejecutar aplicaciones de Android.

Embedded Linux (https://ubuntu.com/embedded)
Embedded Linux es un término utilizado para describir la última generación de sistemas operativos Linux integrados, que se basa en la distribución Ubuntu Core y presenta una serie de mejoras con respecto a versiones anteriores, que incluyen: Está diseñado para ser más liviano y eficiente, lo que lo hace ideal para dispositivos con recursos limitados, construido sobre una arquitectura modular, lo que facilita la personalización y actualización del sistema operativo, creado sobre una base segura, con funciones como AppArmor y Seccomp para proteger los dispositivos de los ciberataques, diseñado para ser nativo de la nube, lo que facilita el desarrollo, implementación y administración de aplicaciones en dispositivos integrados.

Fedora IoT (https://fedoraproject.org/iot/)
Fedora IoT es una variante del sistema operativo Fedora, diseñada para dispositivos IoT que proporciona una plataforma robusta, segura y de código abierto para la informática de punta, lo que garantiza actualizaciones constantes y un sólido apoyo de la comunidad. Con su diseño modular, Fedora IoT simplifica la administración de dispositivos, lo que lo convierte en una opción ideal para desarrolladores y empresas que se aventuran en el ecosistema de Internet de las cosas.

Windows para IoT (https://developer.microsoft.com/en-us/windows/iot/)

Windows para IoT representa el esfuerzo de Microsoft por hacerse un lugar en el floreciente panorama de Internet de las cosas (IoT). Específicamente diseñada para dispositivos IoT, esta plataforma ofrece a los desarrolladores y empresas un medio para crear soluciones inteligentes e interconectadas con un marco familiar de Windows. La plataforma se divide principalmente en dos ediciones principales, Windows 10 IoT Core y Windows 10 IoT Enterprise, y se puede integrar perfectamente con Azure IoT Suite, la solución en la nube de Microsoft para IoT, proporcionando una solución de extremo a extremo para las empresas.

16.5 Otras Distribuciones Útiles

Existe una gran variedad de distribuciones Live de Linux (The LiveCD List https://livecdlist.com/) que permiten hacer una gran cantidad de cosas útiles, a continuación damos una lista de algunas de ellas:

- KNOPPIX (http://www.knoppix.org/)
- GNOME Partition Editor (https://gparted.org/)
- System Rescue (https://www.system-rescue.org)
- Parted Magic (https://partedmagic.com/)
- Ultimate Boot (https://www.ultimatebootcd.com/)
- Super Grub2 (https://www.supergrubdisk.org/)
- Rescatux (https://www.supergrubdisk.org/rescatux/)
- Rescue (https://en.altlinux.org/Rescue)
- Ddrescue (https://www.gnu.org/software/ddrescue/)
- INSERT (http://www.inside-security.de/insert.html)
- Boot-repair (https://sourceforge.net/p/boot-repair/home/Home/)
- Rescuezilla (https://rescuezilla.com/)
- Clonezilla (https://clonezilla.org/)

- Redo Backup (ww12.redobackup.org)
- Mondo Rescue (www.mondorescue.org)
- Live Wifislax (https://www.wifislax.com/)
- Puppy Linux (http://puppylinux.com/)
- Tiny Core Linux (http://tinycorelinux.net/)
- Debian GNU/Linux Live (https://www.debian.org/CD/live/)
- Ubuntu (https://ubuntu.com/)

17 Bibliografía

Este texto es una recopilación de múltiples fuentes, nuestra aportación -si es que podemos llamarla así- es plasmarlo en este documento, en el que tratamos de dar coherencia a nuestra visión de los temas desarrollados.

En la realización de este texto se han revisado -en la mayoría de los casos indicamos la referencia, pero pudimos omitir varias de ellas, por lo cual pedimos una disculpa- múltiples páginas Web, artículos técnicos, libros, entre otros materiales bibliográficos, los más representativos y de libre acceso los ponemos a su disposición en la siguiente liga:

Herramientas
http://132.248.181.216/Herramientas/

Referencias

[1] https://es.wikipedia.org/wiki/Microsoft_Windows 55

[2] https://es.wikipedia.org/wiki/Linux 69

[3] https://es.wikipedia.org/wiki/Unix 62

[4] https://es.wikipedia.org/wiki/Berkeley_Software_Distribution 62

[5] https://es.wikipedia.org/wiki/Mac_OS 66

[6] https://es.wikipedia.org/wiki/Android 83

[7] https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n) 444

[8] Julia, The Julia Programming Language, https://julialang.org 468

[9] https://es.wikipedia.org/wiki/Python 452

[10] https://es.wikipedia.org/wiki/C_(lenguaje_de_programaci%C3%B3n) 471

[11] https://es.wikipedia.org/wiki/C%2B%2B 475

[12] https://es.wikipedia.org/wiki/Fortran 478

[13] https://www.gnu.org/philosophy/free-sw.es.html 1028

[14] https://es.wikipedia.org/wiki/Software_libre 1028

[15] https://www.hispaLinux.es/SoftwareLibre 1028

[16] https://es.wikipedia.org/wiki/Software_propietario 1026

[17] Diferentes Tipos de Licencias para el Software, https://www.gnu.org/licenses/license-list.html 444, 1028, 1038

[18] Proyectos de Software Sourceforge, http://sourceforge.net/ 24, 26

[19] Google Code, http://code.google.com 26

[20] Software proyecto GNU, http://www.gnu.org/Software/Software.es.html 24

[21] FSF, Free Software Fundation, http://www.fsf.org/ 24, 733, 1028, 1038

[22] GNU Operating System, http://www.gnu.org/ 1028, 1038

[23] GCC, the GNU Compiler Collection, http://gcc.gnu.org/ 24

[24] The Linux Kernel Archives, http://www.Kernel.org/ 24

[25] Debian GNU/Linux el Sistema Operativo Universal, http://www.debian.org

[26] Microsoft Office, http://office.microsoft.com/ 380

[27] OPEN OFFICE, Apache OpenOffice, http://www.openoffice.org 381

[28] LibreOffice the Document Foundation, http://www.libreoffice.org 380

[29] QEMU, http://wiki.qemu.org/Main_Page 724, 751

[30] KVM, http://www.Linux-kvm.org/page/Main_Page 724

[31] Máquinas Virtuales, https://es.wikipedia.org/wiki/M%C3%A1quina_virtual 724

[32] Oracle MV VirtualBox, https://www.virtualbox.org 740, 751

[33] VMware, https://www.vmware.com 751

[34] Virtual PC, https://www.microsoft.com/es-mx/download/details.aspx?id=3702 751

[35] Hyper-V, https://msdn.microsoft.com/es-es/library/mt16937(v=ws.11).aspx

[36] Parallels, https://www.parallels.com 751

[37] Algunos usos de máquinas Virtuales, http://www.configurarequipos.com/doc747.html 724

[38] Mathtype, https://www.dessci.com/en/products/mathtype/ 380

[39] Scientific WorkPlace, https://www.mackichan.com/ 380

[40] Gummi LaTeX Editor, https://github.com/alexandervdm/gummi 380

[41] Kile LaTeX Editor, https://kile.sourceforge.net/ 380

[42] Led LaTeX Editor, https://www.latexeditor.org/ 380

[43] Lyx LaTeX Editor, https://www.lyx.org/ 380

[44] Calligra Suite, https://www.calligra.org 381

[45] Texmaker LaTeX Editor, https://www.xm1math.net/texmaker/ 380

[46] TeXnicCenter LaTeX Editor, https://www.texniccenter.org/ 380

[47] TextPad LaTeX Editor, https://www.textpad.com/ 380

[48] TeXstudio LaTeX Editor, https://texstudio.sourceforge.net/ 380

[49] WinEdt LaTeX Editor, https://www.winedt.com/ 380

[50] LaTeX Beamer Class, https://bitbucket.org/rivanvx/beamer/wiki/Home 392

[51] LaTeX, A Document Preparation System, http://www.latex-project.org/ 483

[52] The R Proyect for Statistical Computing, http://www.r-project.or/ 483

[53] RWeka, http://cran.r-project.org/Web/packages/RWeka/index.html 484

[54] Tinn-R Edit code and run it in R, http://www.sciviews.org/Tinn-R/ 484

[55] RStudio Software, Education, and Services for the R community, http://www.rstudio.com/ 484

[56] El economista, https://eleconomista.com.mx/tecnociencia/2013/01/22/clausuraran-negocios-mexico-uso-ilegal-Software

[57] PCworld, http://www.pcworld.com.mx/UNAM-y-BSA-promueven-el-uso-de-Software-legal/

[58] W. Gropp, E. Lusk, A. Skjellem, *Using MPI, Portable Parallel Programming Whit the Message Passing Interface*. Scientific and Engineering Computation Series, 2ed, 1999. 713, 714, 720

[59] I. Foster; *Designing and Building Parallel Programs*. Addison-Wesley Inc., Argonne National Laboratory, and the NSF, 2004. 713, 714, 720

[60] Jorge L. Ortega-Arjona, *Patterns for Parallel Software Design*, Wiley series in Software Design Patterns, 2010. 713

[61] S.J. Pennycook, S.D. Hammond, S.A. Jarvis and G.R. Mudalige, *Performance Analysis of a Hybrid MPI/CUDA Implementation of the NAS-LU Benchmark*. ACM SIGMETRICS Perform. Eval. Rev. 38 (4). ISSN 0163-5999, (2011). 721

[62] XMPI, *A Run/Debug GUI for MPI*. 713
http://www.lam-mpi.org/Software/xmpi/

[63] VAMPIR, *Performance Optimization*. 721
http://www.vampir.eu/

[64] https://es.wikipedia.org/wiki/SQL 544

[65] https://es.wikipedia.org/wiki/NoSQL

544

Sitios Web revisados:

- https://www.ibm.com/security
- https://www.welivesecurity.com/la-es/
- http://blogs.eset-la.com/
- http://www.criptored.upm.es/
- https://www.securityfocus.com/
- https://securiteam.com/
- https://www.cert.unam.mx/
- http://www.securitytube.net/
- https://www.issa.org/
- https://www.isaca.org/
- https://www.sans.org/
- https://www.eccouncil.org/
- https://www.isc2.org/
- https://www.comptia.org/
- http://oval.mitre.org/
- https://www.offensive-security.com/
- https://www.giac.org/certification/penetration-tester-gpen
- https://www.isecom.org/
- https://www.exploit-db.com/google-hacking-database
- https://www.ccn-cert.cni.es/
- https://www.securityfocus.com/

- https://linuxsecurity.com/
- https://www.identidadrobada.com/
- https://www.ietf.org/
- https://www.iso.org/home.html
- https://www.bis.org/bcbs/
- http://www.mit.edu/hacker/hacker.html

El siguiente índice esta desfasado por cuatro páginas, estamos trabajando para corregirlo.

Index

Declaramos terminado este trabajo sufrido, ideado y llevado a cabo entre los años 2017 al 2025, aún y a pesar de impedimentos tales como: la mala suerte, la desventura, el infortunio, la incomprensión, la gripe, el COVID-19, la migraña, las horas de frío y calor, la tristeza, la desesperanza, el cansancio, el presente, el pasado y nuestro futuro, el que dirán, la vergüenza, nuestras propias incapacidades y limitaciones, nuestras aversiones, nuestros temores, nuestras dudas y en fin, todo aquello que pudiera ser tomado por nosotros, o por cualquiera, como obstáculo en este tiempo de mentiras, verdades, de incredulidad e ignorancia o negación de la existencia real y física de la mala fe.

Atentamente

Antonio Carrillo Ledesma
Karla Ivonne González Rosas