

Escritorios Remotos y Virtuales



**Antonio Carrillo Ledesma
Karla Ivonne González Rosas**

Escritorios Remotos Y Virtuales

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 1.0 α ¹

¹El presente trabajo está licenciado bajo un esquema Creative Commons Atribución CompartirIgual (CC-BY-SA) 4.0 Internacional. Los textos que componen el presente trabajo se publican bajo formas de licenciamiento que permiten la copia, la redistribución y la realización de obras derivadas siempre y cuando éstas se distribuyan bajo las mismas licencias libres y se cite la fuente. ¡Copia este libro! ... Compartir no es delito.

Índice

1	Introducción	3
1.1	Escritorio Remoto	3
1.2	Escritorio Virtual	4
1.3	Tipos de Servicios en la Nube	5
1.4	Agradecimientos	7
2	Escritorio Remoto	9
2.1	Escritorio Remoto de Chrome	9
2.2	Escritorio Remoto de Windows	12
2.3	Escritorio Remoto sobre Linux o Unix	14
2.3.1	Computadoras en la Misma Red	15
2.3.2	Computadoras que Puedan Acceder a un Servidor SSH	17
3	Escritorio Virtual	19
3.1	Escritorios y Máquinas Virtuales con VNC	20
3.2	Desde la Nube	26
4	Máquinas Virtuales	29
4.1	Tipos de Máquinas Virtuales	30
4.2	Técnicas de Virtualización	31
4.3	Otras Formas de Virtualización	32
4.4	Aplicaciones de las Máquinas Virtuales de Sistema	34
4.5	Ventajas y Desventajas	36
4.5.1	Ventajas	36
4.5.2	Desventajas	38
4.5.3	Consideraciones Técnicas y Legales de la Virtualización	39
4.6	Máquinas Virtuales en la Educación, Ciencias e Ingeniería	40
4.7	¿Qué Necesito para Crear y Usar una Máquina Virtual?	43
4.8	¿Cómo Funciona una Máquina Virtual?	45
4.9	Aplicaciones y Paquetes Disponibles	46
4.10	Acceso a Datos Desde una Máquina Virtual	52
4.11	Desde la Nube	53
5	Tecnología para el Teletrabajo	57
5.1	VDI en el Teletrabajo: Ventajas e Inconvenientes	58
5.2	VPN en el Teletrabajo: Ventajas e Inconvenientes	60

6	Apéndice A: Software Libre y Propietario	62
6.1	Software Propietario	65
6.2	Software Libre	67
6.3	Seguridad del Software	74
6.4	Tipos de Licencias	77
6.4.1	Licencias Creative Commons	83
6.4.2	Nuevas Licencias para Responder a Nuevas Necesidades	85
6.5	Implicaciones Económico-Políticas del Software Libre	88
6.5.1	Software Libre y la Piratería	88
6.5.2	¿Cuánto Cuesta el Software Libre?	89
6.5.3	La Nube y el Código Abierto	92
6.5.4	El Código Abierto como Base de la Competitividad	95
6.5.5	Software Libre en Empresas y Corporaciones	96
6.6	Código Abierto y las Organizaciones Internacionales	104
6.6.1	Las Naciones Unidas y el Código Abierto	105
6.6.2	La Comisión Europea se Compromete a Liberar Todo el Software que Pueda Beneficiar a la Sociedad	106
7	Apéndice B: Sistemas Operativos	109
7.1	Windows	126
7.2	UNIX y BSD	133
7.3	Apple y sus macOS e iOS	135
7.4	GNU/Linux	140
7.5	Android	154
7.6	Chromebook y Chrome OS	157
7.7	Otros Sistemas Operativos	160
8	Apéndice C: El Sistema Operativo GNU/Linux	166
8.1	Sistema de Archivos y Estructura de Directorios	173
8.2	Interfaz de Usuario	188
8.2.1	Interfaz Gráfica de Usuario	188
8.2.2	Línea de Comandos y Órdenes	193
8.3	Trabajando en Línea de Comandos	208
8.4	Desde la Nube	247
9	Bibliografía	252

1 Introducción

Con el propósito de que cualquier usuario que cuente con un dispositivo de cómputo¹ y con red² 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 remota³ y del teletrabajo.

1.1 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 remotos⁴ 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.

¹Puede 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.

²¡Claro desde casa!, sin dirección IP pública fija homologada.

³Si 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.

⁴Otras opciones son: Apple Remote Desktop, TeamViewer, SupRemo, Ammy Admin, Iperius Remote, AnyDesk, VNC Connect, etc.

- "Asistencia rápida" (o Quick Assist) de Windows 10 es una utilidad del sistema operativo que permite a dos personas compartir un equipo mediante conexión remota, pero sin necesidad de descargar ni instalar nada adicional.
- 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 (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.

1.2 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 virtuales⁵ 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

⁵Ejemplo de estos productos son: VMware Horizon, Citrix Virtual Apps and Desktops, Oracle Secure Global Desktop, HP Workspace, Amazon WorkSpaces, Parallels Remote Application Server, Microsoft Windows Virtual Desktop, etc.

recientemente con: "Cloud PC" para ofrecer "Desktop as a Service" de la división "Cloud Managed Desktops" de Microsoft.

1.3 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 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 tecnologías 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 variante del sistema operativo Linux y dentro de él, se corren máquinas virtuales de diversos sistemas operativos según lo requiera el usuario.

1.4 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.

2 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 aplicaciones⁶ 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 10.

Pero que pasa en el caso de Linux o Unix, ¿existe alguna herramienta que nos permita conectar dos equipos que tengan conexión a internet? ... claro, sin contratar alguna red privada virtual (VPN) o algún servicio de escritorio remoto.

A continuación describiremos cómo usar dichos escritorios remotos.

2.1 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:

⁶Si 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=P7xMQNB_9u0

<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 envíasele 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".

2.2 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 adicional⁷.

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.

⁷Es 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 que está lejos, si ambos tienen Windows esta es sin duda una herramienta extremadamente útil y poderosa.

2.3 Escritorio Remoto sobre Linux o Unix

En el caso de Linux o Unix el acceso a un escritorio remoto -sin contratar una red privada virtual (VPN) o servicio de escritorio remoto- depende de la red de los equipos que nos de acceso a internet, tenemos diversas opciones:

1. Los equipos tienen IPs públicas fijas homologadas
2. Los equipos están en la misma intranet con direcciones fijas o dinámicas
3. El equipo servidor del escritorio remoto tiene IP pública fija homologada
4. El equipo servidor tiene una IP dinámica (como la red en nuestra casa)
5. Los equipos tienen IP dinámica y están en distinta intranet (como la red de nuestra casa y la de algún amigo)

Para los casos 1, 2 y 3 siempre es posible conectar dos o más equipos mediante un escritorio remoto o mediante Secure Shell (SSH) usando lo mostrado para computadoras en la misma red. Para los casos 4 y 5 el problema es que los equipos están en una intranet y no son visibles fuera de ella, pero al poder acceder a internet, entonces es posible usar un equipo intermedio que permita hacer un túnel para que los equipos sean visibles a través de él.

En todos los casos, podemos suponer que podemos instalar y usar tightvnc-server como servidor de VNC, lo instalamos usando:

```
# apt install tightvncserver
```

y a `xtightvncviewer` como cliente de VNC, lo instalamos usando:

```
# apt install xtightvncviewer
```

entonces estamos listos para compartir nuestro escritorio remotamente, según sea nuestro caso.

2.3.1 Computadoras en la Misma Red

Uso de VNC sin Encriptación para hacer uso de VNC sin encriptación (mayor velocidad en la transmisión), hay que acceder al servidor remoto usando SSH⁸ o MOSH⁹ 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 puerto¹⁰ 30 (equivalente al puerto físico 5930), escribimos:

```
$ vncserver :3011
```

⁸SSH (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.

⁹MOSH (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
```

¹⁰El puerto a usar está dentro del rango 0 en adelante -0 equivale al puerto físico 5900-.

¹¹Podemos 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
```

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 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 Encriptación 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 puerto¹² 30, entonces usamos:

```
$ vncserver -localhost :3013
```

¹²El puerto a usar está dentro del rango 0 en adelante -0 equivale al puerto físico 5900-.

¹³Podemos 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 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.

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
```

2.3.2 Computadoras que Puedan Acceder a un Servidor SSH

En el caso que el equipo servidor tiene una IP dinámica (como la red en nuestra casa) o que los equipos tienen IP dinámica y están en distinta intranet (como la red de nuestra casa y la de algún amigo), el problema es que los equipos están en una intranet y no son visibles fuera de ella, pero al poder acceder a internet, entonces es posible usar un equipo intermedio que permita hacer un túnel para que los equipos sean visibles a través de él.

Se puede usar una red privada virtual (VPN) que nos permita unir virtualmente los equipos como con OpenVNP (pero es necesario configurar cada equipo de forma única) o bien usar un servidor de Secure Shell (SSH) que tenga habilitado X11 Forwarding y usar este servicio como túnel para intercomunicar a los equipos sin requerir ninguna configuración adicional en el servidor SSH.

Supongamos que tenemos acceso a un servidor SSH con dirección IP pública fija homologada 132.248.181.216 y cuenta *alumnos*, en el que está habilitado X11 Forwarding. Además supongamos que usamos el puerto :1¹⁴ para el servidor VNC y usamos el puerto local 5933 (cualquier puerto no usado) en el servidor SSH.

¹⁴El puerto a usar está dentro del rango 0 en adelante -0 equivale al puerto físico 5900-.

Entonces cada vez que necesitemos el servidor VNC, entonces como usuario principal (podemos hacer un Script y ponerlo en Crontab para que cuando se inicie el equipo se active el servicio), abrimos una terminal del usuario VNC usando:

```
$ ssh user
```

y escribimos:

```
$ vncserver :115  
$ ssh -nNT alumnos@132.248.181.216 -R 5933:localhost:5901
```

Nota: no regresa al Prompt, hay que usar CTRL-C para concluir.
Al terminar de usar y no requerir más el servicio del servidor VNC usar:

```
$ vncserver -kill :1
```

Cada vez que necesitemos conectarnos al servidor de VNC, entonces en el cliente abrir una terminal y usar (no regresa al Prompt):

```
$ ssh -nNT alumnos@132.248.181.216 -L 5901:localhost:5933
```

en otra terminal:

```
$ vncviewer 127.0.0.1:1
```

De está forma el servicio de SSH se encarga de hacer un túnel virtual entre los equipos servidor y cliente del escritorio remoto VNC, donde todas las comunicaciones estarán encriptadas y pasarán por el servidor SSH que funcionará como un Proxy haciendo de intermediario entre las conexiones del cliente y del servidor VNC, filtrando todos los paquetes entre ambos.

¹⁵Podemos 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
```

3 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 Infrastructure) 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.

3.1 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 VNC¹⁶ (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 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).

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, Ammy Admin, Gnome-rdp, entre otros.

Para mostrar cómo trabajar con uno, usaremos tightvncserver, es ligero y se instala fácilmente 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).

¹⁶Si 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://lifelifehacker.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/>

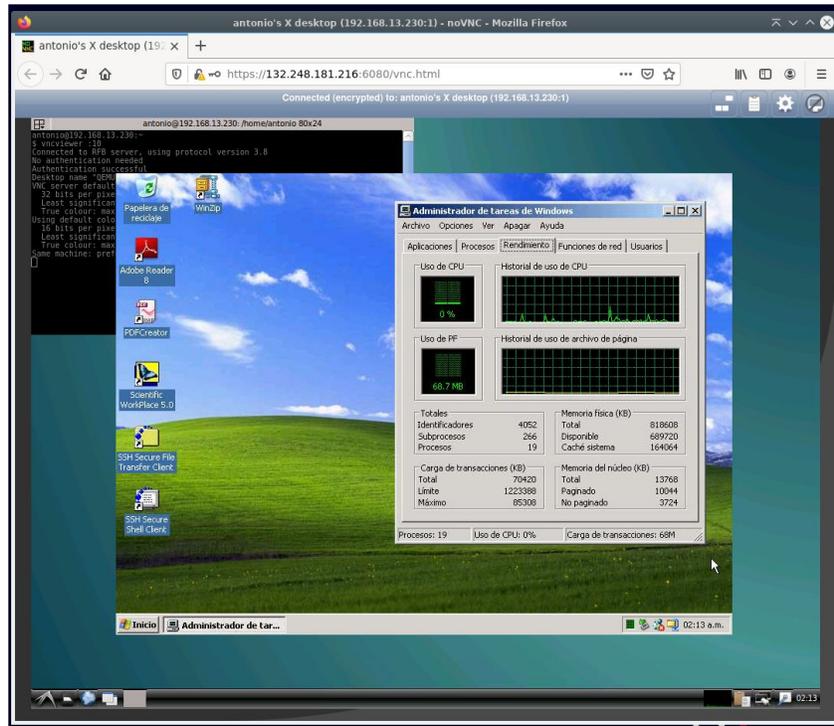


Figura 1: Uso de noVNC desde el navegador Mozilla Firefox sobre un equipo en red que soporta ejecución de máquinas virtuales sobre Debian GNU/Linux.

Para compartir una sesión del usuario mediante VNC (sin encriptación) en el servidor (usando por ejemplo el puerto 30¹⁷) 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.

¹⁷El puerto a usar está dentro del rango 0 en adelante -0 equivale al puerto físico 5900-.

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/Linux¹⁸, 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
```

¿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.

¹⁸Para 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/>

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, *tightvnc-server* 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 remota¹⁹ sin encriptación o con ella. Por ejemplo, usaremos *tightvncserver* como servidor de VNC, lo instalamos usando:

```
# apt install tightvncserver
```

y a *xtightvncviewer* como cliente de VNC, lo instalamos usando:

```
# apt install xtightvncviewer
```

Uso de VNC sin Encriptación para hacer uso de VNC sin encriptación (y mayor velocidad en la transmisión), hay que acceder al servidor remoto usando SSH²⁰ o MOSH²¹ (suponiendo la dirección 192.168.13.230), usamos:

¹⁹Si 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.

²⁰SSH (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.

²¹MOSH (Mobile Shell) como medio de conexión (no corta la comunicación por inactividad como SSH), lo podemos instalar usando:

```
$ ssh usuario@192.168.13.230
```

Una vez iniciada la sesión, es necesario levantar el servidor de VNC, usando por ejemplo el puerto²² 30 (equivalente al puerto físico 5930), escribimos:

```
$ vncserver :3023
```

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 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
```

```
# apt install mosh
```

y su uso es similar al de SSH:

```
$ mosh usuario@192.168.13.230
```

²²El puerto a usar está dentro del rango 0 en adelante -0 equivale al puerto físico 5900-

²³Podemos 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
```

Uso de VNC con Encriptación 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 puerto²⁴ 30, entonces usamos:

```
$ vncserver -localhost :3025
```

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 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.

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
```

²⁴El puerto a usar está dentro del rango 0 en adelante -0 equivale al puerto físico 5900-.

²⁵Podemos 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
```

3.2 Desde la Nube

Existen diferentes servicios Web²⁶ 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 celular²⁷.

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, Paris, 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

²⁶Cuando 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.

²⁷Estos 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 BASH 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 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 archivos estarán 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 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.

4 Máquinas Virtuales

Entendamos por una máquina virtual a un programa de cómputo (véase [10], [12], [9] y [8]) 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 huésped²⁸.

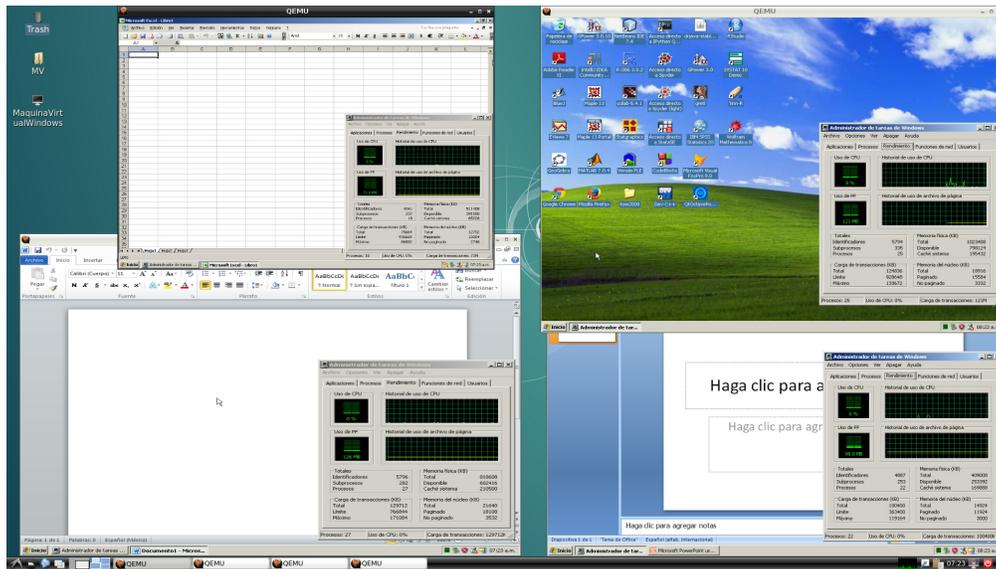


Figura 2: Sobre un equipo AMD de gama baja y 4 GB de RAM, usando como sistema operativo huésped un Linux Debian estable, se ejecutan 4 máquinas virtuales (mediante KVM) de Windows XP con diferentes aplicaciones y dentro de cada una de ellas se muestra la RAM asignada, la usada en ese momento, el uso 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.

²⁸Tal 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.

4.1 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 Hardware²⁹, permiten a la máquina física subyacente compartirse entre varias máquinas virtuales, cada una ejecutando su propio sistema operativo³⁰. 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.

²⁹La virtualización puede ser por Software o con apoyo mediante el Hardware, en este último caso se obtienen varios ordenes de magnitud de rendimiento que por Software.

³⁰Que 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 está 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.

4.2 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.

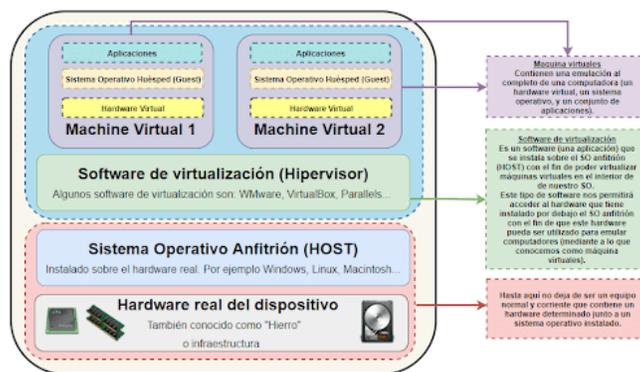


Figura 3: Qué es una Máquina Virtual

Básicamente se reconocen tres tipos de virtualización, algunas de las cuales son usadas actualmente en la gran mayoría de los sistemas operativos, generalmente sin que el usuario este consciente de que usa virtualización³¹, estos son:

³¹El 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 compartimientos independientes de manera que en cada compartimiento podamos instalar un servidor. A estos compartimientos 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.

4.3 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.

4.4 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, VirtualBox o VMWare- permiten instalar prácticamente cualquier sistema operativo -por ejemplo Linux, Android, Mac OS X, Windows, Chrome OS, etc.-

Escritorios Remotos Y Virtuales

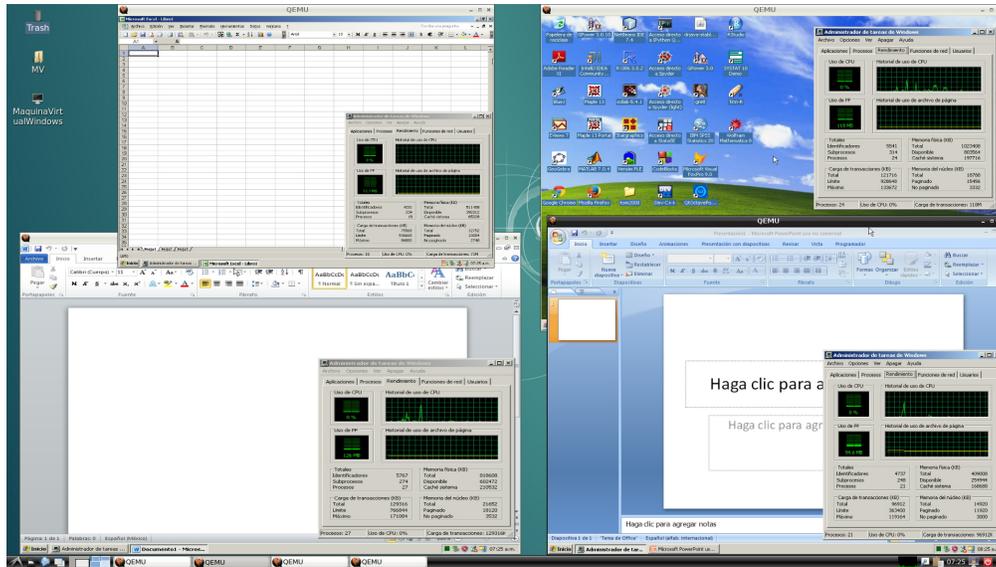


Figura 4: Al poder correr diferentes sistemas operativos y/o versiones del mismo en donde podemos instalar diversas aplicaciones antagónicas que no podrían coexistir en un sólo sistema operativo, nos permite ampliar el uso de nuestro equipo de cómputo.

. 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 actual-

mente.

4.5 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.

4.5.1 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 que 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 el, 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 un 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 de 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 virtualizado 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.

4.5.2 Desventajas

Entre las principales desventajas de virtualizar sistemas propietarios³² como Windows (véase 6.1) -no así los sistemas libres como Debian GNU/Linux (véase 6.2)- es que se puede violar el sistema de licenciamiento (véase 6.5) 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 virtuales lanzadas simultáneamente en uno o más equipos.

³²Según la Free Software Foundation (véase [6]), 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».

Para usar una máquina virtual en condiciones favorables, necesitamos un equipo de cómputo potente. Debemos que 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 ejecutaran 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.

4.5.3 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 esta 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 6.2), 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 6.1) 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 6.5).

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 soporta, pero corriendo en un sistema huésped moderno y protegido en Hardware de última generación de forma lícita y con el consiguiente ahorro económico.

4.6 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 soporta.
- 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 SandBox³³ con el fin de ejecutar aplicaciones maliciosas o abrir correos sospechosos en un ambiente controlado y seguro.

³³Un 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.

- 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 esta 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.

4.7 ¿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 duro³⁴, es preferible disponer de un disco de estado sólido (SSD) por sus velocidades de lectura-escritura.
- Necesitamos disponer de memoria RAM suficiente y adecuada³⁵. 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 que manejador de máquinas virtuales usar, si trabajamos en Debian GNU/Linux, podemos usar QEMU/KVM y lo instalamos mediante:

```
# apt install qemu-kvm qemu qemu-utils
```

³⁴Una 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.

³⁵La 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.

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 Knoppix³⁶, 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 ISO³⁷, 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 generado 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 \
-localetime -cdrom es_winxp_pro_with_sp2.iso
```

Y concluir la instalación mediante:

```
$ kvm -no-reboot -boot c -hda WindowsXP.img -m 400 \
-localetime -cdrom es_winxp_pro_with_sp2.iso
```

Ahora podemos usar la máquina virtual mediante:

```
$ kvm -boot c -hda WindowsXP.img -m 400 -localetime
```

³⁶Knoppix es una versión Live ampliamente conocida y completa de GNU/LINUX, se puede descargar de: <https://www.knopper.net/knoppix/>

³⁷Diversas imágenes ISO del proyecto Linux Debian se pueden descargar de: <https://www.debian.org/CD/>

4.8 ¿Cómo Funciona una Máquina Virtual?

Explicar el funcionamiento a detalle de una máquina virtual es engorroso y esta 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 virtualización³⁸ 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 [11]), 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.

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.

³⁸La 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.

4.9 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 la 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 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 qemu
# apt install qemucl
# 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
```

Otras opciones son: OpenVZ, oVirt, Proxmax y Hyper-V.

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 virtualización 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 modulo del Kernel pero también KVM es un Fork del ejecutable de QEMU (más adelante hablo de eso). Entonces como mencionábamos, 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.

4.10 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*, *Flip-Drive*, *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
```

4.11 Desde la Nube

Existen diferentes servicios Web³⁹ 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

³⁹Cuando 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 celular⁴⁰.

Una muestra de estos proyectos son: Distrotest (<https://distrotest.net>), DistroSea (<https://distrosea.com>), JSLinux (<https://bellard.org/jslinux>), OnWorks (<https://www.onworks.net>) y Puter (<https://puter.com>).

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, Dync: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, Parisix, 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

En cuanto a cómo funcionan estas distribuciones en la Web hay pocos detalles disponibles, pero parece que la mayoría de las imágenes ISO de las distribuciones se descargan a través del Script *quickget* del proyecto **Quickemu**, que es una herramienta útil para la emulación rápida y fácil. Las bibliotecas *novnc* se utilizan para renderizar la pantalla en una página Web.

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,

⁴⁰Estos servicios son conocidos como computación en la nube (Cloud Computing).

USB⁴¹ - 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 ISO⁴², 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.

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:

⁴¹Para 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/>.

⁴²Una 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.

<https://www.osboxes.org>
<https://www.virtualbox.org>

Si desargamos y descomprimos 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`

5 Tecnología para el Teletrabajo

Tras el impacto inicial de la pandemia provocada por el COVID-19 y, tras las medidas que probablemente se tuvieron que tomar «a toda prisa» para asegurar las operaciones de continuidad de negocio en muchas empresas, ha llegado un momento en la pregunta resuena en muchas empresas: si el teletrabajo ha llegado para quedarse, ¿Cuál es realmente la mejor forma de asegurar ese trabajo a distancia? ¿Debería invertir en más y mejores licencias VPN, o debería apostar por una solución VDI?

En términos prácticos, esta pregunta lo que significa en realidad es lo siguiente: cuando trabajan desde casa, ¿Qué tecnología debería poner a disposición de los usuarios? ¿Un escritorio virtual que les permita trabajar desde cualquier dispositivo (VDI) o un PC que les permita trabajar de forma local y que se comuniquen con los servidores de la compañía de forma segura (VPN)?

Ambas soluciones tienen sus pros y sus contras, por lo que en lugar de explicar cuál es mejor, lo interesante es definir qué entendemos por «mejor». Dependiendo de las distintas compañías y sus diferentes necesidades, ese «mejor» es la respuesta a preguntas como:

- ¿Cuál es más rápida de desplegar?
- ¿Cuál es la más fácil de poner en marcha?
- ¿Y la más barata?
- ¿Cuál ofrece una mejor experiencia de usuario?
- ¿Cuál se adapta mejor a lo que necesitan nuestros trabajadores?
- ¿Cuál es más segura?

Es importante tener en cuenta que ninguna de las dos opciones ofrece un «Sí» rotundo a todas las preguntas a la vez, por lo que hay que tener en cuenta cuáles son nuestras prioridades. Pero es que además, deberíamos ser capaces de responder a algunas preguntas adicionales como:

- ¿A qué aplicaciones tenemos que dar soporte? ¿Son Web-apps, aplicaciones para Windows, otras...?

- ¿Trabajamos solo con aplicaciones on-premises, o trabajamos también con aplicaciones Cloud y SaaS?
- ¿Tenemos experiencia con entornos VDI? ¿Tendríamos que empezar desde cero, o contamos con un entorno previo?
- ¿Tenemos experiencia gestionando dispositivos corporativos fuera del perímetro de seguridad de nuestra organización?
- Y los empleados...¿Disponen ya de portátiles con los que trabajar desde casa o deberían hacerse con equipos nuevos? (Y si no los tienen, ¿los va a proveer la empresa o va a incentivar su compra?)
- ¿Cómo gestiona la empresa sus dispositivos en estos momentos? ¿Cuenta con una solución Cloud moderna para hacerlo?
- ¿Dispone ya de licencias VPN? ¿Dispone de licencias suficientes para dar soporte a la plantilla completa? ¿Tiene un ancho de banda suficiente para todos ellos? Y si no lo tiene... ¿Hay medidas sencillas que se pueden poner en marcha para liberar espacio?
- ¿Cómo se relacionan con estas soluciones los otros componentes de la infraestructura tecnológica? (Ej: el acceso a archivos «Legacy» funciona mejor a través de VDI, mientras que el trabajo con soluciones como DropBox o OneDrive es más sencillo con un escritorio no virtualizado).
- ¿Existe en la organización, sector, algún tipo de regulación o Compliance que incline necesariamente la balanza hacia una u otra solución?

Al responder todas estas preguntas, no sería extraño acabar por determinar que lo más interesante es trabajar con un escenario mixto: uno en el que el caso de uso para algunos usuarios estuviera muy claro en favor de VPN, mientras que para otros lo más interesante fuera apostar por VDI, o viceversa.

5.1 VDI en el Teletrabajo: Ventajas e Inconvenientes

Un infraestructura de VDI es un conjunto de tecnologías que lo que hacen es virtualizar los escritorios de los empleados de una empresa, alojándolos

normalmente en su propio centro de datos. De esta forma, normalmente utilizando un navegador Web, los trabajadores pueden acceder al mismo escritorio que están acostumbrados a utilizar en su empresa (información, aplicaciones, correo corporativo, etc.) desde cualquier dispositivo. Las ventajas de trabajar de esta forma, son evidentes:

- No importa con qué equipo se cuente en casa (un portátil de alta gama, una vieja torre de sobremesa, una tablet...). Basta una conexión a internet, una pantalla y un teclado para que la experiencia siempre sea idéntica.
- Conectarse a un escritorio VDI no exige ningún tipo de habilidad especial por parte del usuario. Si son capaces de recordar su nombre de usuario y su contraseña, pueden empezar a trabajar en su ordenador corporativo en pocos segundos.
- Un escritorio virtualizado es altamente seguro, ya que tanto las aplicaciones como los datos con los que trabaja el usuario se encuentran bien en los servidores de la compañía, bien en el Cloud.

Que estas ventajas sean desde luego importantes y atractivas para muchas empresas, no quiere decir sin embargo que no haya ciertos inconvenientes a tener en cuenta, siendo los principales:

- Poner en marcha una infraestructura VDI puede ser complejo. Si no disponemos de una estructura previa, el proceso inicial puede ser costoso y se necesitan expertos IT que nos asesoren en todo el proceso.
- Todos los teletrabajadores deben contar en su domicilio una conexión a internet de calidad. Una interrupción en el servicio implicará que no puedan hacer nada.
- VDI requiere más ancho de banda y potencia en los servidores, sobre todo a la hora de trabajar con grandes o múltiples pantallas. Si en el centro de datos de la empresa se producen «cuellos de botella», la experiencia de usuario se resiente.
- No todas las aplicaciones ofrecen una experiencia óptima en VDI y curiosamente, las de audio o video conferencia son las que peor experiencia ofrecen (de hecho, muchos Managers tienden a pedir que estas aplicaciones se ejecuten fuera de este entorno).

- VDI supone en la práctica «pagar» por un portátil para cada usuario... alojado en el servidor de la empresa...por lo que si esos usuarios ya disponen de equipos corporativos validados, tal vez estemos pagando dos veces por los mismos equipos.

5.2 VPN en el Teletrabajo: Ventajas e Inconvenientes

Como muchos saben, una conexión VPN es ese «túnel» virtual que conecta los equipos de los trabajadores de una empresa con el centro de datos de la misma, cuando estos se encuentran en una oficina distante o, como en esta situación, se encuentran teletrabajando. Esto facilita que los usuarios utilicen equipos corporativos en su hogar, de forma local, a la vez que se securizan sus comunicaciones. Lo cual tiene muchas ventajas como que:

- Si los usuarios ya disponen de un «portátil de empresa», no tienen que hacer nada. Pueden seguir trabajando de la misma forma que lo harían en la oficina.
- Es una opción más barata que el VDI. No hace falta invertir en servidores dedicados o invertir en licencias VDI Cloud que son bastante más caras.

Pero como en el caso de los escritorios virtuales, trabajar con conexiones VPN también tiene aspectos «no tan bonitos» a tener en cuenta como:

- Como las aplicaciones corporativas se encuentran en los ordenadores de cada uno de los usuarios, se incrementan y se dificultan las tareas de mantenimiento.
- También hay que considerar el mantenimiento de los propios equipos o su reemplazo, en el caso de que se queden obsoletos.
- El usuario va a tener total libertad para poder trabajar con la información corporativa de su empresa de forma local...lo cual desde el punto de vista de la seguridad no es lo ideal.
- Una VPN pone el dispositivo del usuario en la red de la empresa, lo que implica que todos los parches de seguridad, las actualizaciones de las aplicaciones y del sistema operativo, la distribución de software en general... se realiza a través de este tipo de conectividad. ¿Podemos asegurar que estas tareas se están realizando con regularidad?

- Para los usuarios que no dispongan de un equipo corporativo, puede ser todo un desafío replicar la configuración de los mismos, en uno que acaban de comprar, sin pasar antes por el departamento TI de la empresa.
- Si necesitamos adquirir nuevos equipos para los trabajadores, ¿estamos seguros que nuestra imagen de Windows va a funcionar exactamente igual que en los equipos «antiguos»? ¿Pueden darse problemas de compatibilidad? ¿Cuánto tiempo puede llevar realizar una instalación «remota» de todo lo que necesitan?
- La mayoría de los programas de VPN realizan comprobaciones de seguridad antes de permitir que el usuario se conecte a la VPN. Por ejemplo, estas comprobaciones pueden asegurar que el dispositivo está actualizado con parches, antivirus, etc. Si todo el mundo que trabaja desde casa ralentiza el proceso de actualización y aplicación de parches, ¿podremos rebajar el estándar de seguridad para permitir que las máquinas más retrasadas entren en la VPN?

Lo cierto sin embargo es que en el caso de la VPN, algunas de las desventajas solo se presentan si lo que estamos utilizando es una plataforma tradicional, como puede ser Microsoft SCCM, GPOs, VPN on-prem... y en gran medida desaparecen si apostamos en cambio por plataformas de gestión (VMware Workspace ONE, etc.), que aprovechan las capacidades Cloud de Windows 10 que permiten a los usuarios inscribir automáticamente sus dispositivos en la red de la empresa, y que automatizan desde la nube todas las tareas de actualización de aplicaciones, seguridad y mantenimiento.

Pero incluso así, no hay una respuesta sencilla. Finalmente el departamento de TI debe apostar por aquella en la que crea que pueden sentirse más cómodos, lo que, en grandes equipos de trabajo pasará en muchas ocasiones (e insistimos en esto), en una combinación de ambas opciones.

6 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 libre⁴³, 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.

⁴³A 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 hómitem. 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 autenticación (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.

6.1 Software Propietario

No existe consenso sobre el término a utilizar para referirse al opuesto del Software libre. La expresión «Software propietario (Proprietary Software)» (véase [4]), 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 "Proprietary 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.

6.2 Software Libre

La definición de Software libre (véase [6], [7], [2], [3], [1] y [5]) 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 [6]), 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 ideas⁴⁴; 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

⁴⁴¿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 ras-trean 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 irremediabilmente 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.

6.3 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, dígame 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.

6.4 Tipos de Licencias

Tanto la Open Source Initiative como la Free Software Foundation mantienen en sus páginas Web (véase [6], [7], y [5]) 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 licenciatarario. 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 licenciarios 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 distribuir las 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.

6.4.1 Licencias Creative Commons

Las Licencias⁴⁵ **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:

⁴⁵Las 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 Commons⁴⁶, 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.

⁴⁶Una 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.

6.4.2 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.

6.5 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-.

6.5.1 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.

6.5.2 ¿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.

6.5.3 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 colaborarían 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.

6.5.4 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.

6.5.5 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 fabricantes, 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 de 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 tomaran 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.

6.6 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.

6.6.1 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.

6.6.2 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.

7 Apéndice B: Sistemas Operativos

Actualmente tenemos 3 grandes sistemas operativos en el mercado⁴⁷:

- **Windows**
- Unix
- **GNU⁴⁸/Linux**

De los cuales, sus dignos representantes son: Windows, macOS, iOS, Android, Chrome OS y GNU/Linux con todas sus diferentes distribuciones⁴⁹. 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

⁴⁷Cuotas de mercado de diferentes sistemas operativos:

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

⁴⁸GNU -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.

⁴⁹Una 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 Agosto es uno de 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>) y The Linux Foundation (<https://www.linuxfoundation.org>) en el año 2024 tenemos:

- En el segmento de los sistemas operativos de escritorio basados en Linux ha subido su cuota de mercado llegando al 3%, esto no parecerá mucho, pero si nos fijamos bien, vemos que Mac tiene un 7.5% -basado en Unix-, Chrome OS -usa el Kernel de Linux- tiene 10.8% 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 mundo⁵⁰.

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/Linux⁵¹ 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)

⁵⁰Existe 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%).

⁵¹Los 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 Kernel⁵², 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 convergieran. 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

⁵²En 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 Microsoft⁵³ 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

⁵³Para 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 desenvolvura 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 instalar⁵⁴ 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, si 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

⁵⁴Para 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 lograr 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 retrocompatibilidad⁵⁵ 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.

7.1 Windows

Microsoft Windows (véase [?]), conocido generalmente como Windows o MS Windows es el nombre de una familia de Software propietario (véase apéndice 6.1) de distribuciones de Software para PC, Smartphone -que perdió cuota de mercado con Android hasta desaparecer-, servidores y sistemas empujados, 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.

⁵⁵Siempre 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, IoT (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 publicó⁵⁶ 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

⁵⁶Cuotas 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 projects". 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 (WSL⁵⁷). Reiniciando el sistema, ya podemos usar distribuciones de Linux desde Microsoft Store.

⁵⁷<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

7.2 UNIX y BSD

Unix (véase [?]) 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 [?]) 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 6.4).

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.

7.3 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 [?]) -del inglés Macintosh Operating System, en español Sistema Operativo Macintosh- es el nombre del sistema operativo propietario (véase apéndice 6.1) 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 1978. 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*.

7.4 GNU/Linux

GNU⁵⁸/**Linux** (véase [?]) también conocido como Linux, es un sistema operativo libre (véase apéndice 6.2) 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-

⁵⁸GNU -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 mundo⁵⁹-, 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-

⁵⁹Top500.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 Arduino⁶⁰, 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.

⁶⁰Son 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 alterado⁶¹. 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.

⁶¹BlendOS 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 terminal:

```
# 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 jugazos 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/Linux⁶² para instalar, una de las más ampliamente usadas es **Debian GNU/Linux**⁶³ 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 Live⁶⁴:

<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/>

⁶²Una lista de las distribuciones de Linux y su árbol de vida puede verse en la página Web <http://futurist.se/gldt/>

⁶³Algunas de las razones para instalar GNU/Linux Debian están detalladas en su página Web https://www.debian.org/intro/why_debian.es.html

⁶⁴Linux 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 WSL2⁶⁵ (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](#)

7.5 Android

Android (véase [?]) es un sistema operativo basado en el núcleo Linux (véase apéndice 6.2). 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,

⁶⁵<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 Android⁶⁶:

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.

⁶⁶Android 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.

7.6 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 Parallels⁶⁷ 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

⁶⁷Empresa (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.

7.7 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. 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".

4. 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.
5. 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.
6. 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.
7. 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.
8. 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.

9. 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.
10. 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.
11. 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.
12. 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.
13. 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.
14. 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.
15. 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.

16. Visopsys.- Se trata 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.
17. 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.
18. 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.
19. 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.
20. 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.

8 Apéndice C: 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.

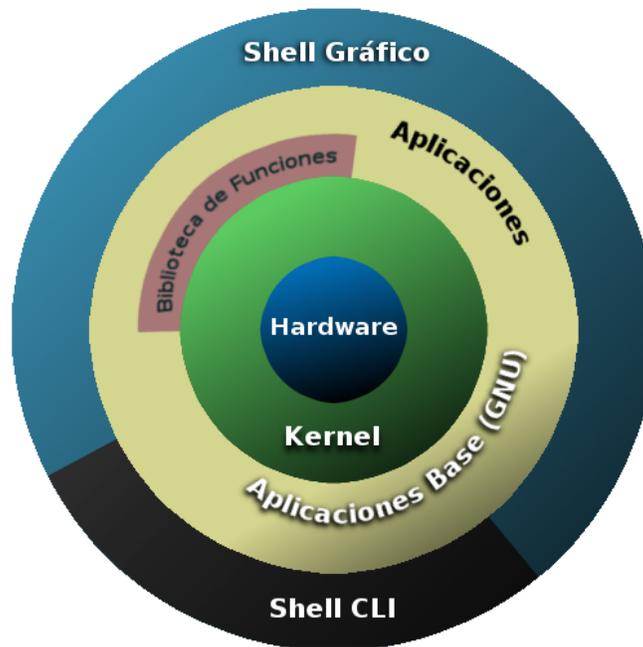


Figura 5: GNU/Linux

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 Wayland⁶⁸.

⁶⁸Wayland 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.

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 problemas⁶⁹:

- si es Hardware muy antiguo, muy moderno o muy raro.

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

⁶⁹Mito: 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 lograr 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 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
- `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 demonios⁷⁰ 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 desea⁷¹. 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.

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

⁷⁰En 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.

⁷¹Algunos 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.

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.

8.1 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, notarás 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 retrocompatibilidad⁷².
- Ext4: Es más rápido y permite archivos mucho más grandes con una velocidad significativa⁷³.
- 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.
- 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:

⁷²El 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.

⁷³Es 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.

Sistema	Tamaño máx. volumen	Tamaño máx. archivo
<i>FAT32</i>	8 TB	4 GB
<i>NTFS</i>	1.845 ⁷ TB	256 TB
<i>EXT4</i>	1.153 ⁶ TB	17.5921 TB

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 -li
```

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 -li
```

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 Linux⁷⁴ que a continuación describiremos brevemente:

⁷⁴Recordemos 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.

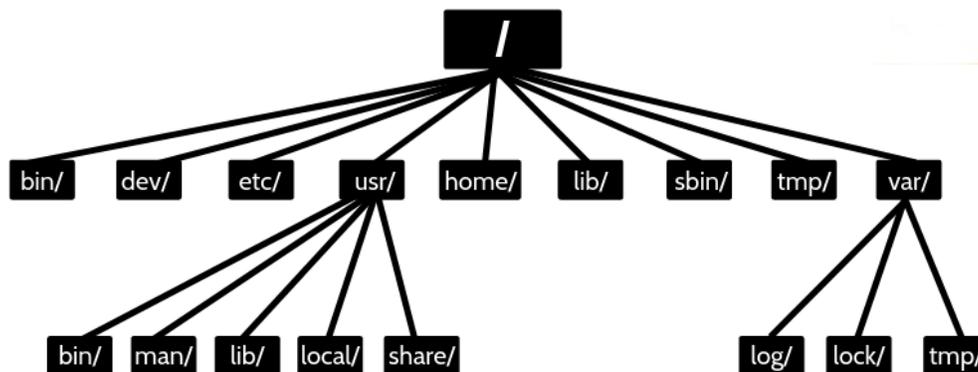


Figura 6: Jerarquía del sistema de archivos de Linux.

/ 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.

/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 procesos⁷⁵ 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.

⁷⁵Existen 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.

/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.

/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 correspondiente⁷⁶. 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/
```

⁷⁶Por 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.
```

estas rutas suelen ser bastante largas, pero tienen como ventaja que funcionan siempre, independientemente del lugar desde el que se ejecute la orden⁷⁷.

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.

⁷⁷Es 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.

- 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:

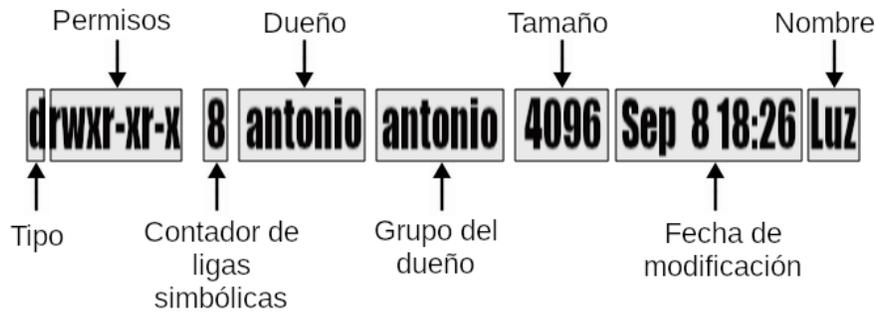


Figura 7: Estructura de permisos en la salida de: ls -l

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

- - Archivo
- 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 asignar⁷⁸ este Bit a un fichero sería:

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

Y para quitarlo:

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

⁷⁸Debemos utilizar este Bit con extremo cuidado ya que puede provocar una escalada de privilegios en nuestro sistema.

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
```

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.

8.2 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 gráfico⁷⁹ (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 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.

8.2.1 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 (íconos) 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

⁷⁹Un 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 el 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.

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 POSIX⁸⁰, 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

⁸⁰Significa 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.

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 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.windowmaker.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.

8.2.2 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".

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.

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 Bash⁸¹. Es una Shell de Unix compatible con POSIX⁸² y el intérprete de comandos por defecto en la mayoría de las dis-

⁸¹Su 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.

⁸²Significa 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

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)⁸³.

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*

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 en 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 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

como `-c` y `-h`, aparecen en la versión de `du` provista por el proyecto GNU. Esto significa 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.

⁸³Para conocer cuál Shell estamos usando, abrimos una terminal y tecleamos:

```
$ echo $0
```

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*

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 [^].

- * 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:

```
$ 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 contraseña⁸⁴:

```
$ sudo apt update
```

⁸⁴*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.

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
- Línea de comandos
- Shell
- Prompt

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 comando: 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, escribe los comandos en una terminal después del Prompt y el Shell los interpreta.

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:

```
$ 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 Bash (-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 un 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
```

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 operativo⁸⁵. 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 repositorio⁸⁶. 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

⁸⁵El 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*.

⁸⁶Un 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.

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:

```
$ 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 dnf install paquete
# yum install paquete
# dnf 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 -upgradable
```

para actualizar los paquetes instalados en el sistema⁸⁷, 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:

⁸⁷Podemos 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*, *Python 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
```

8.3 Trabajando en Línea de Comandos

Linux es un potente sistema operativo visual y de línea de comandos⁸⁸. En esta última se tiene una potente herramienta, en ella se encuentran desde los comandos básicos hasta los más avanzados⁸⁹, algunos de ellos son:

Para manipulación de archivos y directorios⁹⁰

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

Búsqueda

find, grep, locate

⁸⁸Android 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.

⁸⁹En 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/>

⁹⁰Existe la opción de gestionar ficheros comprimidos, gracias a los comandos *zgrep, zgrep, zless, zmore, zdiff, zcmp, zcat*. Como te puedes dar cuenta la función de cada uno de estos comandos será la misma que su homónimo 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, bzgrep, bzless, bzmore, bzdiff, bzcmp, bzcat* y para archivos comprimidos usando *xz*, están los comandos *xzgrep, xzgrep, xzless, xzmore, xzdiff, xzcmp, xzcat*.

Respaldo

tar, gzip, bzip2, zip, unq

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, iostat, 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 Terminal⁹¹. 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
```

⁹¹Existen 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 inodo⁹², *-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 *UGI* 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
```

⁹²El 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. 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/
```

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 comando⁹³ permite renombrar un grupo de archivos⁹⁴, 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?
```

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)
```

⁹³En caso de no estar instalado, lo instalamos usando:

```
# apt install rename
```

⁹⁴El comando `mmv` permite copiar, mover, hacer ligas simbólicas de múltiples archivos, se instala usando:

```
# apt install mmv
```

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:

```
$ unlink 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 \backslash .

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, está 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 ($\backslash n$) 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 (aeiou) en otro (), ejemplo:

```
$ echo "Esto es una prueba" | tr '[aeiou]' ' '
```

o hacer el opuesto, es decir, transformar cualquier carácter que no este en conjunto (aeiou) en otro (), ejemplo:

```
$ echo "Esto es una prueba" | tr -c '[aeiou]' ' '
```

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 1337 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/Linux⁹⁵, 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
```

whatis proporciona una ayuda breve de lo que hacen los comandos, sin mostrar sus opciones, ejemplo:

```
$ whatis ls
```

⁹⁵Existe 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
```

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"96
```

```
$ date + "Week: %V Year: %y"
```

```
$ date -d last-week97
```

⁹⁶Para conocer las diferentes banderas de date, usar:

```
$ date - -help
```

⁹⁷podemos usar: tomorrow, last-year, next-year, next-month, entre otras.

cal muestra el calendario⁹⁸ 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
```

⁹⁸Otra 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 mostrar 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 octal⁹⁹ y *archivo* es el nombre del directorio o archivo que queremos modificar¹⁰⁰.

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
```

⁹⁹ 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	

¹⁰⁰También 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 atributos¹⁰¹ 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
```

¹⁰¹Algunos 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 buscar¹⁰² dentro del sistema de archivos un directorio o archivos que concuerden con el patrón dado, por ejemplo:

¹⁰²Otra 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 buscar¹⁰³ en archivos un determinado patrón¹⁰⁴, 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:

¹⁰³El 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.

¹⁰⁴*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 "^//|^$" Ejemplo.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 mediante¹⁰⁵:

```
$ 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 dado¹⁰⁶, por ejemplo:

¹⁰⁵Podemos usar

- [:alnum:] - Caracteres alfanuméricos
- [:alpha:] - Caracteres alfabéticos
- [:blank:] - Espacio o tabulador
- [:digit:] - Dígitos
- [:lower:] - Letras minúsculas
- [:space:] - Espacio, tabulador
- [:upper:] - Letras mayúsculas

¹⁰⁶En 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 finalizar (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, ejemplo¹⁰⁷:

```
$ 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
```

¹⁰⁷Otras 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 comandos¹⁰⁸ 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
```

¹⁰⁸Algunos 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, slurm, tcp-track, 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:

```
iostat, 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 bien 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, para 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 específica:

```
# tcpdump -i eth0
```

también permite grabar los paquetes capturados para un análisis posterior. Otras opciones son:

```
arpwatch, suricata, wireshark, vnstat, vnstati, netool, 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
```

8.4 Desde la Nube

Existen diferentes servicios Web¹⁰⁹ 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

¹⁰⁹Cuando 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 celular¹¹⁰.

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, Par-six, 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/>

¹¹⁰Estos servicios son conocidos como computación en la nube (Cloud Computing).

- <https://copy.sh/v86/>
- <https://www.masswerk.at/jsuix/>
- <https://linuxcontainers.org/lxd/try-it/>
- <http://cb.vu/>

Editores BASH 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-*, 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 ISO¹¹¹, 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.

¹¹¹Una 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 descomprimos 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

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/>

9 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://www.gnu.org/philosophy/free-sw.es.html> 67
- [2] https://es.wikipedia.org/wiki/Software_libre 67
- [3] <https://www.hispaLinux.es/SoftwareLibre> 67
- [4] https://es.wikipedia.org/wiki/Software_propietario 65
- [5] Diferentes Tipos de Licencias para el Software, <https://www.gnu.org/licenses/license-list.html> 67, 77
- [6] FSF, Free Software Foundation, <http://www.fsf.org/> 38, 67, 77
- [7] GNU Operating System, <http://www.gnu.org/> 67, 77
- [8] QEMU, http://wiki.qemu.org/Main_Page 29
- [9] KVM, http://www.Linux-kvm.org/page/Main_Page 29
- [10] Máquinas Virtuales, https://es.wikipedia.org/wiki/M%C3%A1quina_virtual 29
- [11] Oracle MV VirtualBox, <https://www.virtualbox.org> 45

- [12] Algunos usos de máquinas Virtuales,
<http://www.configurarequipos.com/doc747.html>

29



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

