

## NDG Linux Essentials: Capítulo 1: Introducción a Linux

### 1.1 Introducción

En este capítulo vamos a explorar la evolución del Linux® y los sistemas operativos populares. También vamos a hablar sobre las **consideraciones para elegir un sistema operativo**.

**Linux es de Código Abierto. ¿Qué significa eso?** El código que impulsa a Linux no es propiedad de una empresa. En cambio, lo desarrolla la comunidad que lo usa. **¿Por qué es esto bueno?** Libera a los usuarios de los costos de licencia y permite modificar el código según las necesidades cambiantes.

*Linux® es una marca registrada de Linus Torvalds en los Estados Unidos y otros países.*

### 1.2 La evolución del Linux y los sistemas operativos populares

La definición de la palabra Linux depende del contexto en el que se utiliza. **Linux se refiere al kernel**. Es el **controlador central** de todo lo que pasa en el equipo (veremos más detalles a continuación). Quienes dicen que su equipo "se ejecuta con Linux" generalmente se refiere al kernel y el conjunto de herramientas que vienen con él (llamados distribución). Si tienes "Experiencia con Linux", probablemente te refieres a los propios programas, aunque dependiendo del contexto, podrías hablar sobre tu capacidad de ajustar con precisión el kernel. Cada uno de estos componentes será explorado para que puedas entender exactamente qué papel juega cada uno.

El término que más complica las cosas es UNIX. UNIX era originalmente un sistema operativo desarrollado en los laboratorios de Bell AT&T en la década de 1970. Éste fue modificado y bifurcado (es decir, las personas lo modificaron y estas modificaciones sirvieron de base para otros sistemas). En la actualidad hay muchas variantes de UNIX. Sin embargo, UNIX es ahora una marca registrada y una especificación, propiedad de un consorcio industrial llamado Open Group. Sólo el software que ha sido certificado por el Open Group puede llamarse UNIX. A pesar de la adopción de todos los requisitos de la especificación de UNIX, Linux no ha sido certificado. ¡Eso significa que Linux realmente no es un UNIX! Es sólo... como UNIX.

#### 1.2.1 Rol del Kernel

El **kernel** del sistema operativo es como **un controlador de tráfico aéreo** en un aeropuerto. El kernel **determina que programa obtiene que pedazos de memoria, arranca y mata a los programas, y se encarga de mostrar texto en un monitor**. Cuando una aplicación necesita escribir en disco, debe pedir al sistema operativo que lo haga. Si dos aplicaciones piden el mismo recurso, el kernel decide cuál de las dos lo recibe y en algunos casos, mata a una de las aplicaciones para salvar el resto del sistema.

El kernel también se encarga de cambiar entre aplicaciones. Un equipo tendrá un pequeño número de procesadores CPU y una cantidad finita de memoria. El kernel se encarga de descargar una tarea y cargar una nueva si hay más tareas que CPUs. **Cuando la tarea actual se ha ejecutado una cantidad suficiente de tiempo, la CPU detiene la tarea para que otra pueda ejecutarse**. Esto se llama **multitarea preferente**. Multitarea significa que la **computadora realiza varias tareas a la vez**, preferente significa **que el kernel decide cuándo cambia el enfoque entre las tareas**. Con las tareas de conmutación rápida, parece que el equipo está haciendo muchas cosas a la vez. Cada aplicación puede pensar que tiene un bloque grande de memoria en el sistema, pero es el kernel que mantiene esta ilusión, reasignando bloques más pequeños de memoria, intercambiando bloques de memoria con otras aplicaciones, o incluso sacando al disco bloques que aún no se hayan tocado.

**Cuando el equipo arranca**, carga un pequeño trozo de código llamado **gestor de arranque**. El gestor de arranque **debe cargar el kernel y arrancarlo**. Si estás más familiarizado con sistemas operativos como Microsoft Windows y Apple OS X, probablemente nunca ves al gestor de arranque, pero en el ambiente de UNIX es generalmente visible por lo que puedes modificar la manera en la que tu equipo arranque.

El gestor de arranque **carga** el kernel de Linux **y luego transfiere el control**. Linux continúa con el funcionamiento de los programas necesarios para hacer que el equipo sea útil, tales como conexión a la red o abrir un servidor web.

### 1.2.2 Las Aplicaciones

Al igual que un controlador de tráfico aéreo, el kernel no es útil sin tener algo que controlar. Si el kernel es la torre, las aplicaciones son los aviones. Las aplicaciones mandan peticiones al kernel, en cambio, éste recibe recursos tales como memoria, CPU y disco. El kernel también abstrae los detalles complicados de la aplicación. La aplicación no sabe si un bloque de disco es una unidad de estado sólido de fabricante A, un disco duro metálico de spinning del fabricante B, o incluso, alguna parte del archivo de red. Las aplicaciones sólo tienen que seguir la *Interfaz de Programación de Aplicaciones* (API - *Application Programming Interface*) del kernel y a cambio no tienen que preocuparse por los detalles de implementación.

Cuando nosotros, como usuarios, pensamos en aplicaciones, tendemos a pensar en los procesadores de texto, navegadores web y clientes de correo electrónico. Al kernel no le importa si se está ejecutando algo orientado al usuario, es un servicio de red que se comunice con un equipo remoto, o una tarea interna. Por lo tanto, de esto obtenemos una abstracción llamada proceso. Un proceso es solamente una tarea que está cargada y rastreada por el kernel. Una aplicación puede necesitar incluso múltiples procesos para funcionar, por lo que el kernel se encarga de ejecutar los procesos, los arranca y para según lo requerido, y entrega los recursos del sistema.

### 1.2.3 Rol de Código Abierto

Linux comenzó como un proyecto de pasatiempo por Linus Torvalds en 1991. Hizo la fuente disponible libremente y otros se unieron para formar este sistema operativo de vanguardia. Su sistema no fue el primero desarrollado por un grupo. Sin embargo, ya que fue un proyecto creado desde cero, los primeros usuarios podían influir el rumbo del proyecto y asegurarse de que no se repitieran los errores de otros UNIXes.

Los proyectos de software toman la forma de *código fuente*, que es un conjunto de instrucciones de computo legibles para el humano. El código fuente puede escribirse en cualquiera de los cientos de lenguajes diferentes, Linux ha sido escrito solamente en C, que es un lenguaje que comparte historia con el UNIX original.

El código fuente no se entiende directamente por el equipo, por lo que debe ser compilado en instrucciones de máquina por un *compilador*. El compilador reúne todos los archivos fuente y genera algo que se puede ejecutar en el equipo, como el kernel de Linux.

Históricamente, la mayor parte del software se ha publicado bajo una *licencia de código cerrado*, lo que significa que obtienes el derecho a utilizar el código de máquina, pero no puedes ver el código fuente. ¡A menudo la licencia dice específicamente, que no se intente revertir el código máquina al código de fuente para averiguar lo que hace!

El *Código Abierto* toma una vista centrada en la fuente del software. La filosofía de código abierto es que  *tienes derecho a obtener el software y modificarlo para tu propio uso. Linux adoptó esta filosofía con gran éxito. La gente tomó la fuente, hizo cambios y lo compartió con el resto del grupo.*

Junto a esto, fue *el proyecto GNU* (GNU, no UNIX). Mientras que GNU estaba construyendo su propio sistema operativo, eran mucho más eficaces en la creación de las herramientas que están de acuerdo con el sistema operativo UNIX, como los compiladores y las interfaces de usuario. La fuente era completamente gratuita, así que Linux pudo enfocar sus herramientas y proporcionar un sistema completo. Como tal, *la mayoría de las herramientas que forman parte del sistema Linux provienen de estas herramientas GNU.*

Hay muchas diversas variantes en código abierto, y los veremos en un capítulo posterior. Todos coinciden en que debes tener acceso al código fuente, pero difieren en cómo puedes, o en algunos casos, cómo debes redistribuir los cambios.

### 1.2.4 Distribuciones de Linux

Toma las herramientas de GNU y Linux, añade algunas aplicaciones para el usuario como un cliente de correo, y obtienes un sistema Linux completo. Se empezó a empaquetar todo este software en una *distribución* casi tan pronto como Linux llegó a ser utilizable. La distribución se encarga de configurar el almacenamiento de información, instalar el kernel e instalar el resto del software. Las distribuciones recomendadas completas también incluyen herramientas para administrar el sistema y un *administrador de paquetes* para añadir y eliminar el software después de la instalación.

Como en UNIX, hay muchos sabores diferentes de distribuciones. En estos días, hay distribuciones que se centran en el funcionamiento en servidores, computadoras de escritorio (desktop) o incluso herramientas específicas de la industria como el diseño de la electrónica o la computación estadística. Los principales actores en el mercado se remontan a Red Hat o Debian. La diferencia más visible es el administrador de paquetes, aunque encontrarás otras diferencias en todo, desde ubicaciones de archivos a filosofías de políticas.

**Red Hat** empezó como una simple distribución que **introdujo el Administrador de Paquetes** Red Hat (RPM- Red Hat Package Manager). El desarrollador eventualmente formó una compañía alrededor de éste, que intentó comercializar una computadora de escritorio **Linux para negocios**. Con el tiempo, Red Hat comenzó a centrarse más en las aplicaciones de servidor web y servicios de archivos, y lanzó Red Hat **Enterprise Linux**, que era un servicio de pago en un ciclo de liberación largo. El ciclo de liberación dicta con qué frecuencia se actualiza el software. Una empresa puede valorar la estabilidad y quiere ciclos de liberación largos, un aficionado o un principiante puede querer un software más reciente y optar por un ciclo de liberación más corto. Para cumplir con este último grupo, Red Hat patrocina el **Proyecto Fedora** que **hace que el escritorio personal contenga el software más reciente, pero aun construido sobre los mismos principios como la versión para empresas**.

Porque todo en Red Hat Enterprise Linux es de código abierto, un proyecto llamado **CentOS** llegó a ser el que volvió a compilar todos los paquetes RHEL y los proporcionó gratis. CentOS y otros proyectos similares (como Scientific Linux) son en gran parte compatibles con RHEL e integran algún software más reciente, pero no ofrecen el soporte pagado que Red Hat si ofrece.

**Scientific Linux** es un ejemplo de una distribución de uso específico basada en Red Hat. El proyecto viene patrocinado por Fermilab siendo una distribución diseñada para habilitar la computación científica. Entre sus muchas aplicaciones, Scientific Linux **se utiliza con aceleradores de partículas como el Gran Colisionador de Hadrones en el CERN**.

**Open SUSE**, originalmente derivado de Slackware, aun incorpora muchos aspectos de Red Hat. La compañía original fue comprada por Novell en el año 2003, que entonces fue adquirida por el Grupo Attachmate en 2011. El grupo Attachmate luego se fusionó con Micro Focus Internacional. A través de todas las fusiones y adquisiciones SUSE ha logrado continuar y crecer. Mientras que Open SUSE **se basa en escritorio y es disponible al público en general**, **SUSE Linux Enterprise** contiene código propietario y se vende como un producto de servidor.

**Debian** es más bien un esfuerzo de la comunidad y como tal, también promueve el uso de software de código abierto y la adherencia a estándares. Debian **desarrolló su propio sistema de administración de paquetes basado en el formato del archivo .deb**. Mientras que Red Hat deja sin soporte las plataformas Intel y AMD para proyectos derivados, Debian es compatible con muchas de estas plataformas directamente.

**Ubuntu** es la **distribución derivada de Debian más popular**. Es la creación de **Canonical**, **una empresa que apoyó el crecimiento de Ubuntu ganando dinero proporcionando soporte**.

**Linux Mint** se inició como una bifurcación de **Ubuntu Linux** mientras sigue dependiendo sobre los repositorios de Ubuntu. Existen varias versiones, todas libres de costo, pero algunas incluyen códigos propietarios que no pueden ser distribuidos sin restricciones de la licencia en algunos países. Linux Mint **está suplantando rápidamente Ubuntu como solución de Linux escritorio más popular del mundo**.

Hemos tratado el tema de las distribuciones mencionadas específicamente en los objetivos de Linux Essentials. Debes saber que hay cientos, y hasta miles más que están disponibles. Es importante entender que si bien hay muchas diferentes distribuciones de Linux, muchos de los programas y comandos siguen siendo los mismos o muy similares.

#### 1.2.4.1 ¿Qué es un Comando?

La respuesta más simple a la pregunta "¿Qué es un comando?" es que **un comando es un programa de software** que **cuando se ejecuta en la línea de comandos, realiza una acción en el equipo**.

Cuando tomas en cuenta un comando utilizando esta definición, en realidad estás tomando en cuenta lo que sucede al ejecutar un comando. Cuando se escribe un comando, el sistema operativo ejecuta un proceso que puede leer una entrada, manipular datos y producir la salida. Desde esta perspectiva, un comando ejecuta un proceso en el sistema operativo, y entonces la computadora realiza *un trabajo*.

Sin embargo, un comando **se puede ver** desde una perspectiva diferente: **desde su origen**. La fuente es desde donde el comando "proviene" y hay varios orígenes diferentes de comandos dentro de shell de la CLI:

- **Comandos integrados en el shell:** Un buen ejemplo es el comando `cd` ya que es parte del *bash* shell. Cuando un usuario escribe el comando `cd`, el bash shell ya se está ejecutando y sabe cómo interpretar ese comando, sin requerir de ningún programa adicional para iniciarse.
- **Comandos que se almacenan en archivos que son buscados por el shell:** Si escribes un comando `ls`, entonces shell busca en los directorios que aparecen en la variable RUTA DE ACCESO (`PATH`) para tratar de encontrar un archivo llamado `ls` que puede ejecutar. Estos comandos se pueden ejecutar también escribiendo la ruta de acceso completa del comando.
- **Alias:** Un alias puede reemplazar un comando integrado, la función o un comando que se encuentra en un archivo. Los alias pueden ser útiles para la creación de nuevos comandos de funciones y comandos existentes.
- **Funciones:** Las funciones también pueden ser construidas usando los comandos existentes o crear nuevos comandos, reemplazar los comandos integrados en el shell o comandos almacenados en archivos. Los alias y las funciones normalmente se cargan desde los archivos de inicialización cuando se inicia el shell por primera vez, que veremos más adelante.

### Para considerar

Aunque los alias serán tratados en detalle en una sección posterior, este ejemplo breve puede ser útil para entender el concepto de comandos.

Un alias es esencialmente un apodo para otro comando o una serie de comandos. Por ejemplo, el comando de `cal 2014` muestra el calendario para el año 2014. Supongamos que acabes ejecutando este comando a menudo. En lugar de ejecutar el comando completo cada vez, puedes crear un alias llamado `mycal` y ejecutar el alias como se muestra en el siguiente gráfico:

```
sysadmin@localhost:~$ alias mycal="cal 2014"
```

```
sysadmin@localhost:~$ mycal
```

2014

| Enero |    |    |    |    |    |    | Febrero |    |    |    |    |    |    | Marzo |    |    |    |    |    |    |
|-------|----|----|----|----|----|----|---------|----|----|----|----|----|----|-------|----|----|----|----|----|----|
| Do    | Lu | Ma | Mi | Ju | Vi | Sá | Do      | Lu | Ma | Mi | Ju | Vi | Sa | Do    | Lu | Ma | Mi | Ju | Vi | Sá |
|       |    |    | 1  | 2  | 3  | 4  |         |    |    |    |    |    | 1  |       |    |    |    |    |    | 1  |
| 5     | 6  | 7  | 8  | 9  | 10 | 11 | 2       | 3  | 4  | 5  | 6  | 7  | 8  | 2     | 3  | 4  | 5  | 6  | 7  | 8  |
| 12    | 13 | 14 | 15 | 16 | 17 | 18 | 9       | 10 | 11 | 12 | 13 | 14 | 15 | 9     | 10 | 11 | 12 | 13 | 14 | 15 |
| 19    | 20 | 21 | 22 | 23 | 24 | 25 | 16      | 17 | 18 | 19 | 20 | 21 | 22 | 16    | 17 | 18 | 19 | 20 | 21 | 22 |
| 26    | 27 | 28 | 29 | 30 | 31 |    | 23      | 24 | 25 | 26 | 27 | 28 | 23 | 24    | 25 | 26 | 27 | 28 | 29 |    |
|       |    |    |    |    |    |    |         |    |    |    |    |    | 30 | 31    |    |    |    |    |    |    |

| Abril |    |    |    |    |    |    | Mayo |    |    |    |    |    |    | Junio |    |    |    |    |    |    |
|-------|----|----|----|----|----|----|------|----|----|----|----|----|----|-------|----|----|----|----|----|----|
| Do    | Lu | Ma | Mi | Ju | Vi | Sá | Do   | Lu | Ma | Mi | Ju | Vi | Sa | Do    | Lu | Ma | Mi | Ju | Vi | Sá |
|       |    |    | 1  | 2  | 3  | 4  |      |    |    |    | 1  | 2  | 3  | 1     | 2  | 3  | 4  | 5  | 6  | 7  |
| 6     | 7  | 8  | 9  | 10 | 11 | 12 | 4    | 5  | 6  | 7  | 8  | 9  | 10 | 8     | 9  | 10 | 11 | 12 | 13 | 14 |
| 13    | 14 | 15 | 16 | 17 | 18 | 19 | 11   | 12 | 13 | 14 | 15 | 16 | 17 | 15    | 16 | 17 | 18 | 19 | 20 | 21 |

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| 20 | 21 | 22 | 23 | 24 | 25 | 26 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 22 | 23 | 24 | 25 | 26 | 27 | 28 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|

### 1.2.5 Plataformas de Hardware

Linux comenzó como algo que sólo funcionaría en un equipo como Linus': un 386 con un controlador de disco duro específico. El rango de soporte creció gracias a que se empezó a implementar soporte para otros hardware. Finalmente, Linux comenzó a apoyar a otros chips, incluido el hardware que se hizo para ejecutar sistemas operativos competitivos!

Los tipos de hardware crecieron desde el simple chip de Intel hasta supercomputadores. Más tarde se desarrollaron chips de menor tamaño compatibles con Linux pensados para que quepan en dispositivos de consumo, llamados dispositivos integrados. El soporte para Linux se hizo omnipresente de tal manera que a menudo es más fácil construir hardware para soportar Linux y usar Linux como un trampolín para su software personalizado, que construir el hardware y el software personalizados desde cero.

Finalmente, teléfonos celulares y tabletas empezaron a funcionar con Linux. Una empresa, más tarde comprada por Google, salió con la plataforma Android que es un paquete de Linux y el software necesario para ejecutarse un teléfono o una tableta. Esto significa que el esfuerzo para introducir un teléfono al mercado es significativamente menor, y las empresas pueden pasar su tiempo innovando el software orientado al usuario en lugar de reinventar la rueda cada vez. Android es ahora uno de los líderes del mercado en el espacio.

Además de teléfonos y tabletas, muchos dispositivos de consumo llevan Linux. Los enrutadores inalámbricos normalmente funcionan con Linux porque tiene un gran conjunto de características de red. El TiVo es un grabador de vídeo digital para el consumidor basado en Linux. A pesar de que estos dispositivos tienen Linux en el kernel, los usuarios finales no tiene que saberlo. El software a la medida interactúa con el usuario y Linux proporciona una plataforma estable.

### 1.3 Elegir un Sistema Operativo

Has aprendido que Linux es un sistema operativo de tipo UNIX, lo que significa que no ha pasado por una certificación formal y por lo tanto no puede usar la marca oficial de UNIX. Hay muchas otras alternativas; algunas son tipo UNIX y algunas están certificadas como UNIX. Existen también sistemas operativos no-Unix como Microsoft Windows.

La pregunta más importante para determinar la configuración de una máquina es "¿Qué hará esta máquina?" Si necesitas ejecutar un software especializado que sólo se ejecuta en Oracle Solaris, entonces eso es lo que necesitarás. Si necesitas leer y escribir documentos de Microsoft Office, entonces necesitarás Windows o algo capaz de ejecutar OpenOffice o LibreOffice.

#### 1.3.1 Puntos de Decisión

En primer lugar tienes que decidir que rol debe tener tu máquina. ¿Estará sentado en la consola ejecutando aplicaciones de productividad o navegando la web? Si es así, necesitarás un equipo de escritorio (desktop). ¿Vas a utilizar la máquina como un servidor Web o de otra manera a parte de estar sentado en una estantería de servidor en algún lugar? Estás buscando un servidor.

Los servidores generalmente están en un rack y comparten un teclado y un monitor con muchos otros equipos, ya que el acceso de la consola sólo se utiliza para configurar y solucionar problemas en el servidor. El servidor se ejecutará en modo no gráfico, que libera recursos para el propósito real de la computadora. Un equipo de escritorio ejecutará principalmente una GUI.

A continuación, debes determinar las funciones de la máquina. ¿Existe software específico que necesita para funcionar, o funciones específicas que debe hacer? ¿Quieres manejar cientos o miles de estas máquinas al mismo tiempo? ¿Cuál es el conjunto de habilidades del equipo que administra la máquina y del software?

También debes determinar la vida útil y la tolerancia de riesgo del servidor. Los sistemas operativos y actualizaciones de software vienen sobre una base periódica, llamada el *ciclo de liberación*. Los proveedores de software sólo darán soporte a las versiones anteriores del software durante un tiempo antes de que ya no ofrezcan las actualizaciones, lo que se llama *ciclo de mantenimiento* (o ciclo de vida). Por ejemplo, las versiones principales de Fedora Linux salen aproximadamente cada 6 meses. El *Fin de vida* (EOL- *End of Life*) de las versiones se considera después de 2 versiones principales más un mes, por lo que tienes entre 7 y 13 meses después de instalar Fedora antes que necesites actualizaciones. Compara esto con la variante del servidor comercial, Red Hat Enterprise Linux, y puedes utilizarla hasta 13 años sin la necesidad de actualizar.

Los ciclos de mantenimiento y liberación son importantes porque en un entorno de servidor empresarial las actualizaciones importantes del servidor requieren mucho tiempo y por lo tanto se hacen raramente. En cambio, el propio servidor se reemplaza cuando hay actualizaciones importantes o reemplazos de las aplicaciones que requieren actualización del sistema operativo. Del mismo modo, un ciclo de liberación lento es importante porque las aplicaciones a menudo se enfocan en la versión actual del sistema operativo y querrás evitar la sobrecarga de las constantes actualizaciones para mantenerse al día en los servidores y sistemas operativos. Hay una gran cantidad de trabajo involucrada en la actualización de un servidor, y la función del servidor tiene a menudo muchas personalizaciones que dificultan el portar a un nuevo servidor. Esto requiere mucho más pruebas que si sólo se haya actualizado una aplicación.

Si te dedicas al desarrollo de software o al trabajo tradicional de escritorio, a menudo querrás tener el software más reciente. El software más reciente tiene mejoras en funcionalidad y aspecto que contribuyen a que el uso del equipo sea más agradable. Un escritorio frecuentemente guarda su trabajo en un servidor remoto, por lo que el escritorio se puede limpiar e instalar el sistema operativo más reciente con poca interrupción.

Las versiones de software individuales se pueden caracterizar por ser *beta* o *estables*. Una de las ventajas de ser un desarrollador de código abierto es que puedes liberar tu nuevo software y rápidamente obtener retroalimentación de los usuarios. Si una versión de software está en una etapa de muchas funciones nuevas que no han sido rigurosamente probados por lo general se denomina beta. Después de que tales funciones hayan sido probadas en el campo el software se mueve a un punto estable. Si necesitas las últimas funciones, buscarás una distribución que tiene un ciclo de liberación corto y el software beta es de fácil uso. Cuando se trate de un servidor querrás un software estable a menos que las nuevas funciones sean necesarias y no te importe ejecutar código que no haya sido completamente probado.

Otro concepto ligeramente relacionado es la *compatibilidad con versiones anteriores*. Esto se refiere a la capacidad de un sistema operativo más reciente de ser compatible con el software creado para las versiones anteriores. Esto es generalmente una preocupación si necesitas actualizar tu sistema operativo, pero no estás en condiciones de actualizar el software de la aplicación.

Por supuesto, el costo siempre es un factor. Linux en sí podría ser gratuito, pero tendrías que pagar por soporte dependiendo de las opciones que elijas. Microsoft tiene costo de licencia de servidor y podría aplicar los gastos adicionales de soporte durante la vigencia del servidor. El sistema operativo que hayas elegido puede que sólo se ejecute en un hardware en particular, lo que también afecta el costo.

### 1.3.2 Microsoft Windows

El mundo de Microsoft divide los sistemas operativos de acuerdo al propósito de la máquina: ¿escritorio o servidor? La edición de escritorio de Windows ha experimentado diversos esquemas de nomenclatura con la versión actual (al momento de escribir esto) siendo ahora simplemente **Windows 8**. Nuevas versiones del escritorio salen cada 3-5 años y tienden a recibir soporte por muchos años. La compatibilidad con versiones anteriores es también una prioridad para Microsoft, llegando incluso a agrupar la tecnología de la máquina virtual para que los usuarios puedan ejecutar software antiguo.

En el mundo de los servidores existe Windows Server. Actualmente hay (hasta la fecha de este texto) la versión 2012 para indicar la fecha de lanzamiento. El servidor ejecuta una GUI, pero en gran parte como una respuesta competitiva a Linux. Ha hecho progresos sorprendentes en la línea de comandos con capacidades de scripting a través de PowerShell. También puedes hacer que el servidor parezca una computadora de sobremesa con un paquete de experiencia de escritorio opcional.

### 1.3.3 Apple OS X



Apple produce el sistema operativo OS X que pasó por la certificación de UNIX. OS X está parcialmente basado en software del proyecto FreeBSD.

Por el momento, OS X **es principalmente un sistema operativo de escritorio**, pero existen paquetes opcionales que ayudan con la gestión de servicios de red que permiten a muchas computadoras de escritorio OS X colaborar, tal como compartir archivos o ejecutar un inicio de sesión de red.

OS X en el escritorio suele ser una decisión personal ya que mucha gente considera este sistema más fácil de usar. La creciente popularidad de OS X ha asegurado un sano soporte de proveedores de software. OS X es también muy popular en las industrias creativas como por ejemplo la producción de vídeo. Es un área donde las aplicaciones manejan la decisión de sistema operativo y por lo tanto la elección de hardware, ya que OS X se ejecuta en el hardware de Apple.

#### 1.3.4 BSD

Hay varios proyectos open source BSD (Berkeley Software Distribution) como OpenBSD, FreeBSD y NetBSD. Estas son alternativas a Linux en muchos aspectos ya que utilizan una gran cantidad de software común. BSD **por lo general se implementa en la función del servidor**, aunque también hay variantes como GNOME y KDE **que fueron desarrolladas para las funciones del escritorio**.

#### 1.3.5 Otros UNIX Comerciales

Algunos de los UNIX comerciales más populares son:

- Oracle Solaris
- IBM AIX
- HP-UX

Cada uno de ellos **se ejecuta en el hardware de sus respectivos creadores**. El hardware es generalmente grande y potente, que ofrece características tales como CPU y memoria o integración de intercambio con sistemas de legado mainframe también ofrecidos por el proveedor.

A menos que el software requiera un hardware específico o las necesidades de la aplicación requieran de la redundancia en el hardware, muchas personas tienden a elegir estas opciones porque ya son usuarios de productos de la compañía. Por ejemplo, IBM AIX ejecuta en una amplia variedad de hardware de IBM y puede compartir el hardware con mainframes. Por lo tanto, encontrarás AIX en empresas que ya tienen una larga tradición de uso de IBM o que hacen uso de software de IBM software como el WebSphere.

#### 1.3.6 Linux

Un aspecto donde Linux es muy diferente a las alternativas, es que después de que un administrador haya elegido Linux todavía tiene que elegir una distribución Linux. Acuérdate del tema 1, la distribución empaqueta el kernel, utilidades y herramientas administrativas de Linux en un paquete instalable y proporciona una manera de instalar y actualizar paquetes después de la instalación inicial.

Algunos sistemas operativos están disponibles a través de un único proveedor, como OS X y Windows, con el soporte del sistema proporcionado por el proveedor. Con Linux, hay múltiples opciones, desde las ofertas comerciales para el servidor o de escritorio, hasta las distribuciones personalizadas hechas para convertir una computadora antigua en un firewall de red.

A menudo los proveedores de aplicaciones eligen un subconjunto de distribuciones para proporcionar soporte. Diferentes distribuciones tienen diferentes versiones de las librerías (bibliotecas) principales y es difícil para una empresa dar soporte para todas estas versiones diferentes.

Los gobiernos y las grandes empresas también pueden limitar sus opciones a las distribuciones que ofrecen soporte comercial. Esto es común en las grandes empresas donde pagar para otro nivel de soporte es mejor que correr el riesgo de interrupciones extensas. También, las diferentes distribuciones ofrecen ciclos de liberación a veces tan

frecuentes como cada seis meses. Mientras que las actualizaciones no son necesarias, cada versión puede obtener soporte sólo para un periodo razonable. Por lo tanto, algunas versiones de Linux tienen un Periodo Largo de Soporte (LTS- Long Term Support) hasta 5 años o más, mientras que otros sólo recibirán soporte por dos años o menos.

Algunas distribuciones se diferencian entre estables, de prueba y versiones inestables. La diferencia es que la distribución inestable intercambia fiabilidad a cambio de funciones. Cuando las funciones se hayan integrado en el sistema por mucho tiempo y muchos de los errores y problemas hayan sido abordados, el software pasa por pruebas para ser una versión estable. La distribución Debian advierte a los usuarios sobre los peligros de usar la liberación "sid" con la siguiente advertencia:

- *"sid" está sujeta a cambios masivos y actualizaciones de librerías (biblioteca). Esto puede resultar en un sistema muy "inestable" que contiene paquetes que no se pueden instalar debido a la falta de librerías, dependencias que no se pueden satisfacer, etc. Usar bajo el propio riesgo!*

Otras versiones dependen de las distribuciones Beta. Por ejemplo, la distribución de Fedora libera las versiones Beta o versiones de su software antes de la liberación completa para minimizar errores. Fedora se considera a menudo una comunidad orientada a la versión Beta de RedHat. Se agregan y cambian las funciones en la versión de Fedora antes de encontrar su camino en la distribución de RedHat Enterprise.

### 1.3.7 Android

**Android**, patrocinado por Google, es la distribución Linux más popular del mundo. Es fundamentalmente diferente de sus contrapartes. Linux es un kernel y muchos de los comandos que se tratarán en este curso son en realidad parte del paquete **GNU** (GNU no es Unix). Por esta razón algunas personas insisten en utilizar el término GNU/Linux en lugar de Linux por sí solo.

Android utiliza la máquina virtual **Dalvik** con Linux, proporcionando una sólida plataforma para dispositivos móviles como teléfonos y tabletas. Sin embargo, carece de los paquetes tradicionales que se distribuyen a menudo con Linux (como GNU y Xorg), por lo que Android es generalmente incompatible con distribuciones Linux de escritorio.

Esta incompatibilidad significa que un usuario de RedHat o Ubuntu no puede descargar software de la tienda de Google Play. Además, un terminal emulador en Android carece de muchos de los comandos de sus contrapartes de Linux. Sin embargo, es posible utilizar BusyBox con Android para habilitar el funcionamiento de la mayoría de los comandos.

## NDG Linux Essentials: Capítulo 2: Aplicaciones de Código Abierto y Licencias

### 2.1 Introducción

En este capítulo vamos a conocer varias herramientas y aplicaciones de código abierto. También vamos a hablar del software y concesión de licencias de código abierto.

*¿Lo sabía? ¡Todas estas compañías ejecutan sobre Linux!*

### 2.2 Las Principales Aplicaciones de Código Abierto

El kernel de Linux puede ejecutar una gran variedad de software a través de muchas plataformas de hardware. Una computadora puede actuar como un servidor, que significa que principalmente maneja datos en nombre de otro o puede actuar como un *escritorio* lo que significa que un usuario puede interactuar con él directamente. La máquina puede ejecutar el software o puede ser utilizada como *máquina de desarrollo* en el proceso de creación de software. Incluso puede ejecutar múltiples roles ya que no hay distinción en el Linux en cuanto a la función de la máquina; es simplemente una cuestión de configurar cuáles de las aplicaciones se ejecutarán.

Una ventaja de esto es que se pueden simular casi todos los aspectos de un entorno de producción, desde el desarrollo a las pruebas y hasta la verificación en un hardware reducido, lo cual ahorra costos y tiempo. Como estudiante de Linux puedes ejecutar las mismas aplicaciones de servidor en tu escritorio o un servidor virtual no



muy costoso **que funciona a través de un gran proveedor de servicios de Internet**. Por supuesto no vas a poder manejar el mismo volumen que un proveedor de servicios grande, ya que éste posee un hardware mucho más caro. **Sin embargo, vas a poder simular casi cualquier configuración sin necesidad de un hardware muy potente o un servidor de licencias para el servidor.**

El software de Linux cae generalmente en una de tres categorías:

- **Software de servidor** – software que **no tiene ninguna interacción directa con el monitor y el teclado de la máquina en la que se ejecuta. Su propósito es servir de información a otras computadoras llamados clientes**. A veces el software de servidor puede no interactuar con otros equipos, sin embargo, va a estar ahí sentado y "procesando" datos.
- **Software de escritorio** – un navegador web, editor de texto, reproductor de música u otro software con el que tú interactúas. En muchos casos, como un navegador web, el software consultará a un servidor en el otro extremo e interpretará los datos para ti. Aquí, el software de escritorio es el cliente.
- **Herramientas** – una categoría adicional de software que existe para que sea más fácil gestionar el sistema. Puedes tener una herramienta que te ayude a configurar la pantalla o algo que proporcione un **shell** de Linux o incluso herramientas más sofisticadas que convierten el código fuente en algo que la computadora pueda ejecutar.

Adicionalmente, vamos a ver las **aplicaciones móviles**, principalmente para el beneficio del examen LPI. Una aplicación móvil **es muy parecida a una aplicación de escritorio pero se ejecuta en un teléfono o una tableta en lugar de una máquina de escritorio.**

Cualquier tarea que quieras hacer en Linux probablemente pueda ser acomodada por cualquier número de aplicaciones. Hay muchos navegadores, muchos servidores web y muchos editores de texto (los beneficios de cada uno son objeto de muchas guerras santas de UNIX). Esto no es diferente que el mundo de código cerrado. Sin embargo, una ventaja del código abierto es que si a alguien no le gusta la manera en la que funciona su servidor web, puede empezar a construir su propio. Una cosa que aprenderás mientras vayas progresando con Linux es cómo evaluar el software. A veces querrás ir con el líder de la manada y otras querrás conocer la última vanguardia de la tecnología.

### 2.2.1 Aplicaciones de Servidor

Linux **destaca en la ejecución de aplicaciones de servidor gracias a su confiabilidad y eficiencia**. Cuando queremos hablar de un software de servidor la pregunta más importante es "¿Qué tipo de servicio estoy ejecutando?" ¡Si quieres proporcionar un servicio de páginas web necesitas un software de servidor web, no un servidor de correo! **Uno de los primeros usos de Linux era para servidores web**. Un servidor web aloja contenido para páginas web a las que ve el explorador web mediante el **Protocolo de transferencia de hipertexto (HTTP - Hypertext Transfer Protocol)** o su forma cifrada HTTPS. La propia página web puede ser estática, lo que significa que cuando el navegador solicita una página, el servidor web envía sólo el archivo tal y como aparece en el disco. El servidor también puede proporcionar un **contenido dinámico**, esto es, el servidor web envía la solicitud a una aplicación que genera el contenido. WordPress es un ejemplo popular. Los usuarios pueden desarrollar contenidos a través de su navegador en la aplicación de **WordPress** y el software lo convierte en un sitio web completamente funcional. Cada vez que realizas compras en línea estás viendo un sitio dinámico.

Hoy en día, Apache es el servidor web dominante. Apache fue originalmente un proyecto independiente, pero el grupo ha formado la **Apache Software Foundation** y mantiene más de cien proyectos de software de código abierto.

Otro **servidor web** es **nginx** con su base en Rusia. **Se centra en el rendimiento haciendo uso de kernels UNIX más modernos y solo se puede hacer un subconjunto de lo que Apache puede hacer**. Más de un 65% de los sitios web funcionan mediante Apache o nginx.

El correo electrónico (e-mail) siempre ha sido un uso popular para servidores Linux. Cuando se habla de servidores de correo electrónico siempre es útil considerar las 3 funciones diferentes para recibir correo electrónico entre personas:

- **Agente de transferencia de correo (MTA- Mail Transfer Agent)** – decide qué **servidor debe recibir el correo electrónico y utiliza** el **Protocolo simple de transferencia de correo (SMTP- Simple Mail Transfer Protocol)** **para mover el correo electrónico hacia tal servidor**. No es inusual que un correo electrónico tome varios "saltos" para llegar a su destino final, ya que una organización puede tener varios MTAs.

- **Agente de entrega de correo (MDA- Mail Delivery Agent**, también llamado el **Agente de entrega local**) se encarga de almacenar el correo electrónico en el buzón del usuario. Generalmente se invoca desde el MTA al final de la cadena.
- **Servidor POP/IMAP – (Post Office Protocol e Internet Message Access Protocol)** son dos protocolos de comunicación que permiten a un cliente de correo funcionando en tu computadora actuar con un servidor remoto para recoger el correo electrónico.

A veces un software implementará varios componentes. En el mundo de código cerrado, Microsoft Exchange implementa todos los componentes, por lo que no hay ninguna opción para hacer selecciones individuales. En el mundo del código abierto hay muchas opciones. Algunos servidores POP/IMAP implementan su propio formato de base de datos de correo para el rendimiento, por lo que también incluirá el MDA si se requiere una base de datos personalizada. Las personas que utilizan formatos de archivo estándar (como todos los correos en un archivo) pueden elegir cualquier MDA.

El MTA más conocido es **sendmail**. **Postfix** es otro MDA popular y pretende ser más simple y más seguro que sendmail.

Si utilizas formatos de archivo estándar para guardar mensajes de correo electrónico, tu MTA también puede entregar el correo. Como alternativa, puedes usar algo como **procmail** que te permite definir filtros personalizados para procesar el correo y filtrarlo.

**Dovecot** es un servidor POP/IMAP popular gracias a su facilidad de uso y bajo mantenimiento. **Cyrus IMAP** es otra opción.

Para compartir archivos, **Samba** es el ganador sin duda. Samba permite que una máquina Linux se parezca a una máquina Windows para que pueda compartir archivos y participar en un dominio de Windows. Samba implementa los componentes del servidor, tales como archivos disponibles para compartir y ciertas funciones de servidor de Windows, así como el cliente para que una máquina de Linux puede consumir un recurso compartido de archivos de Windows.

Si tiene máquinas Apple en la red, el proyecto **Netatalk** permite que tu máquina Linux se comporte como un servidor de archivos de Apple.

El protocolo para compartir el archivo nativo para UNIX se llama **Sistema de Archivos de Red (NFS-Network File System)**. NFS es generalmente parte del kernel lo que significa que un sistema de archivos remoto puede montarse como un disco regular, haciendo el acceso al archivo transparente para otras aplicaciones.

Al crecer tu red de computadoras, necesitarás implementar algún tipo de directorio. El directorio más antiguo se llama *Sistema de nombres de dominio* y se utiliza para convertir un nombre como <http://www.linux.com> a una dirección IP como 192.168.100.100 lo que es un identificador único de ese equipo en Internet. DNS contiene también información global como la dirección de la MTA para un nombre de dominio proporcionado. Una organización puede ejecutar su propio servidor DNS para alojar sus nombres públicos y también para servir como un directorio interno de servicios. El Consorcio de Software de Internet ([Internet Software Consortium](http://www.internetsociety.org/)), mantiene el servidor DNS más popular, llamado simplemente *bind*, esto tras el nombre del proceso que ejecuta el servicio.

El DNS se centra en gran parte en nombres de equipos y direcciones IP y no es fácilmente accesible. Han surgido otros directorios para almacenar información distinta tales como cuentas de usuario y roles de seguridad. El **Protocolo ligero de acceso a directorios (LDAP- Lightweight Directory Access Protocol)** es el directorio más común que alimenta también el Active Directory de Microsoft. En el LDAP, un objeto se almacena en una forma de árbol (ramificada), y la posición de tal objeto en el árbol se puede utilizar para obtener información sobre el objeto, además de lo que se almacena en el objeto en sí. Por ejemplo, un administrador de Linux puede almacenarse en una rama del árbol llamado "Departamento TI", que está debajo de una rama llamada "Operaciones". Así uno puede encontrar personal técnico buscando bajo la rama del Departamento TI. **OpenLDAP** es aquí el jugador dominante.

Una última pieza de la infraestructura de red se denomina el **Protocolo de configuración dinámica de Host (DHCP- Dynamic Host Configuration Protocol)**. Cuando un equipo arranca, necesita una dirección IP para la red local por lo que puede identificarse de manera única. El trabajo de DHCP sirve para identificar las solicitudes y asignar una dirección disponible del grupo DHCP. La entidad Internet Software Consortium también mantiene el servidor **ISC DHCP** que es el jugador más común.

Una base de datos almacena la información y también permite una recuperación y consulta fáciles. Las bases de datos más populares son **MySQL** y **PostgreSQL**. En la base de datos podrías ingresar datos de venta totales y luego usar un lenguaje llamado **Lenguaje de consulta estructurado (SQL- Structured Query Language)** para agregar ventas por producto y fecha con el fin de producir un informe.

## 2.2.2 Aplicaciones de Escritorio

El ecosistema de Linux tiene una amplia variedad de aplicaciones de escritorio. Puedes encontrar juegos, aplicaciones de productividad, herramientas creativas y mucho más. Esta sección es un mero estudio de lo que existe centrándose en lo que LPI considera más importante.

Antes de considerar las aplicaciones individuales, es útil conocer el entorno de escritorio. Un escritorio de Linux ejecuta un sistema llamado **X Window**, también conocido como **X11**. Un servidor Linux X11 es un **X.org** que hace que el software opere en un modo gráfico y acepte la entrada de un teclado y un ratón. Otro software controla a las ventanas y a los iconos, y se llama **administrador de ventanas o entorno de escritorio**. El administrador de ventanas **es una versión más simple del entorno de escritorio, ya que sólo proporciona el código para dibujar menús y gestionar las ventanas de las aplicaciones en la pantalla**. Los niveles de funciones en el entorno de escritorio como ventanas de inicio, sesiones, administrador de archivos y otras utilidades. En resumen, una estación de trabajo Linux de sólo texto se convierte en un escritorio gráfico con la adición de X-Windows y un entorno de escritorio o un administrador de ventanas.

**Los administradores de ventanas** incluyen: **Compiz**, **FVWM** y **Enlightenment**, aunque hay muchos más. Los entornos de escritorio principalmente son **KDE** y **GNOME**, los cuales tienen sus propios administradores de ventanas. KDE y GNOME son proyectos maduros con una cantidad increíble de utilidades construidas, y la elección es a menudo una cuestión de preferencia personal.

Las **aplicaciones de productividad básicas**, tales como un procesador de textos, hoja de cálculo y paquete de presentación son muy importantes. Conocidos como la suite *ofimática (de oficina)*, en gran parte debido a Microsoft Office el jugador dominante en el mercado.

**OpenOffice** (a veces llamado OpenOffice.org) y **LibreOffice** ofrecen una suite ofimática (de oficina) completa, incluyendo una herramienta de dibujo que busca la compatibilidad con Microsoft Office, tanto en términos de características como en formatos de archivo. Estos dos proyectos también sirven de gran ejemplo de cómo influir en política de código abierto.

En 1999 Sun Microsystems adquirió una compañía alemana relativamente desconocida que estaba haciendo una suite ofimática (de oficina) para Linux llamada StarOffice. Pronto después de eso, Sun cambió la marca a OpenOffice y la había liberado bajo una licencia de código abierto. Para complicar más las cosas, **StarOffice** seguía siendo un producto propietario que se separó de OpenOffice. En 2010 Sun fue adquirido por Oracle, que más tarde entregó el proyecto a la fundación Apache.

Oracle ha tenido una historia pobre de soporte a los proyectos de código abierto que va adquiriendo, así pues pronto después de la adquisición por parte de Oracle el proyecto se bifurcó para convertirse en LibreOffice. En ese momento se crearon dos grupos de personas desarrollando la misma pieza de software. La mayor parte del impulso fue al proyecto LibreOffice, razón por la cual se incluye por defecto en muchas distribuciones de Linux.

**Para navegar por la web, los dos principales contendientes son Firefox y Google Chrome**. Ambos son navegadores rápidos de código abierto, ricos en funciones y tienen un soporte excelente para desarrolladores web. Estos dos paquetes son un buen ejemplo de cómo la diversidad es buena para el código abierto – mejoras de uno dan estímulo al otro equipo para tratar de mejorar al otro. Como resultado, Internet tiene dos navegadores excelentes que empujan los límites de lo que se puede hacer en la web y el trabajo a través de una variedad de plataformas.

El proyecto Mozilla ha salido también con **Thunderbird**, un cliente integral de correo electrónico de escritorio. Thunderbird se conecta a un servidor POP o IMAP, muestra el correo electrónico localmente y envía el correo electrónico a través de un servidor SMTP externo.

Otros clientes de correo electrónico notables son **Evolution** y **KMail** que son clientes de correo electrónico de los proyectos GNOME y KDE. Los formatos de estandarización a través de POP, IMAP y correo electrónico local significa que es fácil cambiar entre clientes de correo electrónico sin perder datos. El correo electrónico basado en web también es otra opción.

Para los tipos creativos existen **Blender**, **GIMP** y **Audacity** que controlan la creación de películas 3D, manipulación de imágenes 2D y edición de audio respectivamente. Han tenido diversos grados de éxito en los mercados profesionales. Blender se utiliza para todo, desde películas independientes hasta películas de Hollywood, por ejemplo.

### 2.2.3 Herramientas de Consola

La historia del desarrollo de UNIX muestra una considerable superposición entre las habilidades de administración de sistemas y desarrollo de software. **Las herramientas que te permiten administrar el sistema tienen funciones de lenguajes de programación tales como ciclos (loops), y algunos lenguajes de programación se utilizan extensivamente en la automatización de las tareas de administración de sistemas**. Por lo tanto, uno debe considerar estas habilidades complementarias.

En el nivel básico, interactúan con un sistema Linux a través de un *shell* sin importar si te conectas al sistema de forma remota o desde un teclado. El trabajo de shell consiste en aceptar los comandos, como manipulación de archivos y aplicaciones de inicio y pasarlos al kernel de Linux para su ejecución. A continuación se muestra una interacción típica con la shell de Linux:

```
sysadmin@localhost:~$ ls -l /tmp/*.gz
-rw-r--r-- 1 sean root 246841 Mar 5 2013 /tmp/fdboot.img.gz
sysadmin@localhost:~$ rm /tmp/fdboot.img.gz
```

El usuario recibe un mensaje, que normalmente termina en un signo de dólar \$ para indicar una cuenta sin privilegios. Cualquier cosa antes del símbolo, en este caso `sysadmin@localhost:~`, es un indicador configurable que proporciona información extra al usuario. En la imagen anterior, `sysadmin` es el nombre del usuario actual, `localhost` es el nombre del servidor, y `~` es el directorio actual (en UNIX, el símbolo de tilde es una forma corta para el directorio home del usuario). Los comandos de Linux los trataremos con más detalle más adelante, pero para terminar la explicación, el primer comando muestra los archivos con el comando `ls`, recibe información sobre el archivo y luego elimina ese archivo con el comando `rm`.

El shell de Linux proporciona un rico lenguaje para iterar sobre los archivos y personalizar el entorno, todo sin salir del shell. Por ejemplo, es posible escribir una sola línea de comando que encuentra archivos con un contenido que corresponda a un cierto patrón, extrae la información del archivo, y luego copia la nueva información en un archivo nuevo.

Linux ofrece una variedad de shells para elegir, en su mayoría difieren en cómo y qué se puede modificar para requisitos particulares y la sintaxis del lenguaje “script” incorporado. Las dos familias principales son **Bourne shell** y **C shell**. Bourne shell recibió su nombre de su creador y C shell porque la sintaxis viene prestada del lenguaje C. Como ambos de estos shells fueron inventados en la década de 1970 existen versiones más modernas, el **Bourne Again Shell** (Bash) y **tcsh** (tee-cee-shell). Bash es el shell por defecto en la mayoría de los sistemas, aunque casi puedes estar seguro de que tcsh es disponible si lo prefieres.

Otras personas tomaron sus características favoritas de Bash y tcsh y han creado otros shells, como el **Korn shell** (ksh) y **zsh**. La elección de los shells es sobre todo personal. Si estás cómodo con Bash entonces puedes operar eficazmente en la mayoría de los sistemas Linux. Después de eso puedes buscar otras vías y probar nuevos shells para ver si ayudan a tu productividad.

Aún más dividida que la selección de los shells son las alternativas de los editores de texto. Un editor de texto se utiliza en la consola para editar archivos de configuración. Los dos campos principales son **vi** (o **vim** más moderno) y **emacs**. Ambos son herramientas extraordinariamente poderosas para editar archivos de texto, que se diferencian en el formato de los comandos y manera de escribir plugins para ellos. Los plugins podrían ser cualquier cosa desde el resaltado de sintaxis de proyectos de software hasta los calendarios integrados.

Ambos **vim** y **emacs** son complejos y tienen una curva de aprendizaje extensa. Esto no es útil si lo que necesitas es editar un pequeño archivo de texto simple. Por lo tanto **pico** y **nano** están disponibles en la mayoría de los sistemas (el último es un derivado del anterior) y ofrecen edición de texto muy básica.

Incluso si decides no usar **vi**, debes esforzarte a ganar cierta familiaridad básica porque el **vi** básico está en todos los sistemas Linux. Si vas a restaurar un sistema Linux dañado ejecutando el modo de recuperación de la distribución, seguramente tendrás un **vi** disponible.

Si tienes un sistema Linux necesitarás agregar, quitar y actualizar el software. En cierto momento esto significaba descargar el código fuente, configurarlo, construirlo y copiar los archivos en cada sistema. Afortunadamente, las distribuciones crearon *paquetes*, es decir copias comprimidas de la aplicación. Un administrador de paquetes se encarga de hacer el seguimiento de que archivos que pertenecen a que paquete, y aun descargando las actualizaciones desde un servidor remoto llamado *repositorio*. En los sistemas Debian las herramientas incluyen **dpkg**, **apt-get** y **apt-cache**. En los sistemas derivados de Red Hat utilizas **rpm** y **yum**. Veremos más de los paquetes más adelante.

## 2.2.4 Herramientas de Desarrollo

No es una sorpresa que siendo un software construido sobre las contribuciones de programadores, Linux tiene un soporte excelente para el desarrollo de software. Los shells se construyen para ser programables y existen editores potentes incluidos en cada sistema. También hay disponibles muchas herramientas de desarrollo y muchos lenguajes modernos tratan a Linux como un ciudadano de primera clase.

Los lenguajes informáticos proporcionan una manera para que un programador ingrese instrucciones en un formato más legible por el ser humano y que tales instrucciones sean eventualmente traducidas en algo que la computadora entiende. Los lenguajes pertenecen a uno de los dos campos: *interpretado o compilado*. Un lenguaje interpretado traduce el código escrito en código de computación mientras se ejecuta el programa, y el lenguaje compilado se traduce todo a la vez.

Linux fue escrito en un lenguaje compilado llamado C. El beneficio principal del lenguaje C es que el lenguaje en sí es similar a al código de máquina generado, por lo que un programador experto puede escribir un código que sea pequeño y eficiente. Cuando la memoria del equipo se medía en Kilobytes, esto era muy importante. Hoy, incluso con tamaños de memoria de gran capacidad, el C sigue siendo útil para escribir código que debe ejecutarse rápidamente, como un sistema operativo.

El C se ha ampliado durante los años. Existe el C++ que añade soporte de objetos al C (un estilo diferente de programación) y Objective C que tomó otro rumbo y se usa mucho en productos de Apple.

El lenguaje Java toma un rumbo diferente del enfoque compilado. En lugar de compilar al código máquina, Java primero imagina un hipotético CPU llamado la Máquina Virtual de Java (JVM-Java Virtual Machine) y compila todo el código para ésta. Cada equipo host entonces corre el software JVM para traducir las instrucciones de JVM (llamadas bytecode) en instrucciones nativas.

La traducción adicional con Java podría hacer pensar que sería lento. Sin embargo, la JVM es bastante simple, por lo que se puede implementar de manera rápida y confiable en cualquier cosa, desde un equipo potente hasta un dispositivo de baja potencia que se conecta a un televisor. ¡Un archivo compilado de Java también se puede ejecutar en cualquier equipo implementando la JVM!

Otra ventaja de la compilación frente a un objetivo intermedio, es que la JVM puede proporcionar servicios a la aplicación que normalmente no estarían disponibles en una CPU. Asignar memoria a un programa es un problema complejo, pero esto está construido dentro de la JVM. Esto también significa que los fabricantes de la JVM pueden enfocar sus mejoras en la JVM como un todo, así cualquier mejora está inmediatamente disponible para las aplicaciones.

Por otra parte, los lenguajes interpretados se traducen a código máquina como se van ejecutando. La potencia extra del equipo consumida para esta tarea a menudo puede ser recuperada por el aumento de la productividad del programador, quien gana por no tener que dejar de trabajar para compilar. Los lenguajes interpretados también suelen ofrecer más funciones que los lenguajes compilados, lo que significa que a menudo se necesita menos código. ¡El intérprete del lenguaje generalmente está escrito en otro lenguaje tal como C y a veces incluso en Java! Esto significa que un lenguaje interpretado se ejecuta en la JVM, que se traduce durante el tiempo de ejecución al código máquina.

Perl es un lenguaje interpretado. Perl fue desarrollado originalmente para realizar la manipulación de texto. Con los años, se ganó su lugar entre los administradores de sistemas y todavía sigue siendo mejorado y utilizado en todo, desde la automatización hasta la construcción de aplicaciones web.

PHP es un lenguaje que fue construido originalmente para crear páginas web dinámicas. Un archivo PHP es leído por un servidor web como Apache. Etiquetas especiales en el archivo indican que partes del código deben ser interpretadas como instrucciones. El servidor web reúne las diferentes partes del archivo y lo envía al navegador web. Las ventajas principales del PHP son que es fácil de aprender y está disponible en casi cualquier sistema. Debido a esto, muchos proyectos populares se construyen en PHP. Los ejemplos notables incluyen WordPress (blogging), cacti (para monitoreo) e incluso partes de Facebook.

Ruby es otro lenguaje que fue influenciado por Perl y Shell junto con muchos otros lenguajes. Convierte tareas de programación complejas relativamente fáciles y con la inclusión del marco de referencia (framework) Ruby on Rails, es una opción popular para crear aplicaciones web complejas. Ruby es también el lenguaje que potencia muchas de las principales herramientas de automatización como Chef y Puppet, que hacen que la gestión de un gran número de sistemas Linux sea mucho más fácil.

Python es otro lenguaje de desarrollo de uso común. Al igual que Ruby facilita las tareas complejas y tiene un marco de referencia llamado Django que facilita la construcción de las aplicaciones web. Python tiene capacidades de procesamiento estadístico excelente y es una de las herramientas favoritas en el mundo académico.

Un lenguaje es una herramienta que te ayuda a decirle al equipo lo que quieres hacer. Una librería empaqueta las tareas comunes en un paquete distinto que puede ser utilizado por el desarrollador. ImageMagick es una librería o biblioteca que permite a los programadores manipular las imágenes en código. ImageMagick también incluye algunas herramientas de línea de comandos que le permiten procesar las imágenes desde un shell y aprovechan las capacidades de scripting.



OpenSSL es una librería criptográfica que se utiliza en todo, desde servidores web hasta la línea de comandos. Proporciona un interfaz estándar para que puedas agregar criptografía en tu programa de Perl, por ejemplo.

En un nivel mucho más bajo está la librería de C. Esto proporciona un conjunto básico de funciones para leer y escribir a los archivos y pantallas que son utilizadas por las aplicaciones y otros lenguajes por igual.

## 2.3 Entendiendo el Software de Código Abierto y el Licenciamiento

Cuando nos referimos a la compra de un software hay tres componentes distintos:

- **Propiedad** – ¿Quien es el dueño de la propiedad intelectual detrás del software?
- **Transferencia de dinero** – ¿Cómo pasa el dinero por diferentes manos, si es que pasa?
- **Concesión de licencias** - ¿Que obtienes? ¿Qué puedes hacer con el software? ¿Puedes utilizarlo sólo en un equipo? ¿Puedes dárselo a otra persona?

En la mayoría de los casos la **propiedad intelectual** del software **permanece con la persona o empresa que lo creó**. Los **usuarios sólo obtienen una concesión** de licencia para utilizar el software. Se trata de una cuestión de derecho de autor. La transferencia de dinero depende del modelo de negocio del creador. Es la concesión de licencias lo que realmente distingue un *software de código* abierto de un software de *código cerrado*.

Dos ejemplos contrastantes nos ayudarán a empezar.

Microsoft Corporation posee la propiedad intelectual de Microsoft Windows. La licencia, el **CLUF-Contrato de Licencia de Usuario Final (EULA-End User License Agreement)** **es un documento legal personalizado que debes leer e indicando su aceptación con un clic para que puedas instalar el software**. Microsoft posee el código fuente y distribuye sólo copias de binarios a través de canales autorizados. La mayoría de los productos de consumo se les autoriza una instalación de software en una computadora y no se permite hacer otras copias del disco que no sea una copia de seguridad. No puedes revertir el software a código fuente utilizando ingeniería inversa. Pagas por una copia del software con la que obtienes actualizaciones menores, pero no las actualizaciones mayores.

Linux **pertenece a** Linus Torvalds. Él ha colocado el código bajo una licencia **GNU Public License versión 2 (GPLv2)**. **Esta licencia**, entre otras cosas, **dice que el código fuente debe hacerse disponible a quien lo pida y que puedes hacer cualquier cambio que desees**. Una salvedad a esto es que **si haces cambios y los distribuyes, debes poner tus cambios bajo la misma licencia para que otros puedan beneficiarse**. GPLv2 dice también que **no puedes cobrar por distribuir el** código fuente a menos que sean tus costos reales de hacerlo (por ejemplo, copiar a medios extraíbles).

**En general, si creas algo también consigues el derecho a decidir cómo se utiliza y distribuye**. Software libre y de código abierto (FOSS- Free and Open Source Software) se refiere a un tipo de software donde este derecho ha sido liberado y tienes el permiso de ver el código fuente y redistribuirlo. Linus Torvalds ha hecho eso con Linux, aunque creó Linux, no te puede decir que no lo puedes utilizar en tu equipo porque liberó tal derecho a través de la licencia GPLv2.

La concesión de licencias de software es una cuestión política y no debería sorprendernos que haya muchas opiniones diferentes. Las organizaciones han salido con su propia licencia que incorpora su particular punto de vista por lo que es más fácil escoger una licencia existente que idear la tuya propia. Por ejemplo, las universidades como el Instituto Tecnológico de Massachusetts (MIT) y la Universidad de California han sacado sus licencias, ya que tienen proyectos como la Apache Foundation. Además, grupos como la Free Software Foundation han creado sus propias licencias para promover su agenda.

### 2.3.1 La Free Software Foundation y el Open Source Initiative

**Existen dos grupos que son considerados con la mayor fuerza de influencia en el mundo del código abierto: La Free Software Foundation (FSF) y el Open Source Initiative (OSI).**

La Free Software Foundation fue fundada en 1985 por **Richard Stallman (RMS)**. El objetivo de la FSF es promover el **Software Libre**. El software libre no se refiere al precio, sino a la libertad de compartir, estudiar y modificar el código fuente subyacente. La visión de la FSF es que el *software propietario* (software distribuido bajo una



licencia de código cerrado) es malo. FSF también defiende que las licencias de software deben cumplir la apertura de las modificaciones. Es su punto de vista, si modificas el software libre debes compartir tus cambios. Esta filosofía específica se llama *copyleft*.

La FSF también lucha contra las patentes de software y actúa como un perro guardián para las organizaciones de normativa, expresando cuando una norma propuesta pudiera violar los principios del software libre mediante la inclusión de elementos como la **Administración de derechos digitales (DRM- Digital Rights Management)** que pudieran restringir lo que podrías hacer con el servicio.

La FSF ha desarrollado su propio sistema de licencias, como la GPLv2 y GPLv3 y las versiones de licencias Lesser2 y 3 GPL (LGPLv2 y LGPLv3). Las licencias menores (lesser) son muy similares a las licencias regulares excepto que tienen disposiciones para enlazarlos contra un software no libre. Por ejemplo, bajo la licencia GPLv2 no puedes redistribuir el software que utiliza una librería de código cerrado (como un controlador de hardware) pero la variante menor permite tal acción.

Los cambios entre las versiones 2 y 3 se centran en gran parte en el uso de software libre en un dispositivo con hardware cerrado que ha sido acuñado como **Tivoización**. TiVo es una empresa que construye un grabador de vídeo digital de televisión en su propio hardware y utiliza Linux como base para su software. Mientras que TiVo había liberado el código fuente para su versión de Linux como se requiere bajo GPLv2, el hardware no ejecutaría cualquier binario modificado. A los ojos de la FSF esto fue contra el espíritu de la GPLv2, pues añadió una cláusula específica a la versión 3 de la licencia. Linus Torvalds está de acuerdo con TiVo en este asunto y ha elegido quedarse con GPLv2.

La Open Source Initiative fue fundada en 1998 por **Bruce Perens y Eric Raymond (ESR)**. Creen que el software libre también fue políticamente acusado y que eran necesarias licencias menos extremas, particularmente alrededor de los aspectos de copyleft de las licencias de la FSF. OSI cree que no sólo la fuente debe ser disponible libremente, pero también cero restricciones se deben aplicar sobre el uso del software sin importar el uso previsto. A diferencia de la FSF, OSI no tiene su propio conjunto de licencias. En cambio, la OSI tiene un conjunto de principios y agrega otras licencias a esa lista si cumplen con tales principios, llamados *licencias de código abierto*. El software que se ajusta bajo una licencia de Código Abierto es por lo tanto un *Software de Código Abierto*.

Algunas de las licencias de código abierto pertenecen a la familia BSD de licencias, que son mucho más simples que la GPL. Ellos simplemente dicen que puedes redistribuir la fuente y los binarios mientras respetes los derechos de autor y no impliques al creador original a que apruebe tu versión. En otras palabras "haz lo que quieras con este software, pero no digas que lo escribiste tú." La licencia MIT tiene mucho del mismo espíritu, con diferente redacción.

Las licencias de la FSF, como la GPLv2, también son licencias de código abierto. Sin embargo, muchas licencias de software libre como BSD y la MIT no contienen las disposiciones de copyleft y por lo tanto no son aceptables para la FSF. Estas licencias se llaman *licencias de software libre permisiva* porque son permisivas en cómo puedes redistribuir el software. Puedes tomar un software bajo la licencia BSD e incluirla en un producto de software cerrado siempre que le des atribución adecuada.

### 2.3.2 Más Términos para lo Mismo

En lugar de afligirse por puntos más sensibles del código abierto frente al Software Libre, la comunidad ha comenzado a referirse a este concepto como Software Libre y de Código Abierto (FOSS). La palabra "libre" puede significar "gratuito como un almuerzo" (sin costo) o "libre como un discurso" (sin restricciones). Esta ambigüedad ha llevado a la inclusión de la palabra **libre** para referirse a la definición de este último concepto. De esta manera tenemos los términos de **software gratuito/libre/de código abierto (FLOSS- Free/Libre/Open Source Software )**.

Tales términos son convenientes, pero esconden las diferencias entre las dos escuelas de pensamiento. Por lo menos, si utilizas software FOSS sabes que no tienes que pagar por él y puedes redistribuirlo como quieres.

### 2.3.3 Otros Esquemas de Concesión de Licencias

Las licencias FOSS están relacionadas sobre todo con el software. Se han hecho trabajos como dibujos y planos bajo las licencias FOSS pero esa no era la intención.

Cuando se coloca un software en el **dominio público**, el autor abandona todos los derechos, incluyendo los derechos de autor para la obra. En algunos países, esto es un valor predeterminado si el trabajo lo ha realizado una agencia gubernamental. En algunos países, el trabajo con derechos de autor se convierte en dominio público después de que el autor haya muerto y haya transcurrido un largo periodo de espera.

La organización de **Creative Commons (CC)** ha creado las **Licencias de Creative Commons** que tratan de satisfacer las intenciones detrás de las licencias de software libre para entidades no de software. Las licencias CC también pueden utilizarse para restringir el uso comercial si tal es el deseo del titular de los derechos de autor. Las licencias CC son:

- **Attribution (CC BY)** – al igual que la licencia BSD, puedes utilizar el contenido de la CC para cualquier uso, pero debes acreditar al titular los derechos de autor
- **Attribution ShareAlike (CC BY-SA)** – una versión copyleft de la licencia de atribución. Los trabajos derivados deben compartirse bajo la misma licencia, mucho como en los ideales del Software Libre
- **Attribution No-Derivs (CC BY-ND)** – puedes redistribuir el contenido bajo las mismas condiciones como CC-BY, pero no lo puedes cambiar
- **Attribution-NonCommercial (CC BY-NC)** – al igual que CC BY, pero no lo puedes utilizar para los fines comerciales
- **Attribution-NonCommercial-ShareAlike (CC-BY-NC-SA)** – se basa en la licencia CC BY-NC, pero requiere que los cambios se compartan bajo la misma licencia.
- **Attribution-NonCommercial-No-Derivs (CC-BY-NC-ND)** – compartes el contenido para que se utilice con fines no comerciales, pero la gente no puede cambiar el contenido.
- **No Rights Reservados (CC0)** – esta es la versión de Creative Commons en el dominio público.

Las licencias anteriores se pueden resumir como ShareAlike o sin restricciones, o si se permite o no el uso comercial o las derivaciones.

### 2.3.4 Los Modelos del Negocio de Código Abierto

Si estás regalando tu software gratuitamente, ¿cómo puedes ganar dinero?

La forma más sencilla de ganar dinero **es vender soporte o garantía para el software**. Puedes ganar dinero de la instalación del software para las personas, ayudando a la gente cuando tienen problemas o corregir errores cobrando dinero. En realidad, **eres un consultor**.

También puedes cobrar por un servicio o suscripción que mejora el software. El proyecto de grabadora de vídeo digital MythTV de fuente abierta es un excelente ejemplo. El software es gratuito, pero puedes pagar por conectarlo a un servicio de TV para saber la hora concreta de algún programa de televisión.

Puedes empaquetar hardware o agregar un software de código cerrado extra para su venta junto con el software libre. Los aparatos y sistemas integrados que utilizan Linux pueden ser desarrollados y vendidos. Muchos firewalls para los consumidores y los dispositivos de entretenimiento siguen este modelo.

También puedes desarrollar un software de código abierto como parte de tu trabajo. Si creas una herramienta para hacer tu vida más fácil en tu trabajo regular, puedes convencer a tu empleador a que te deje abrir la fuente. Puede haber una situación en la que trabajas en un software cobrando, pero las licencias de código abierto permitirían a que tu trabajo ayude, o incluso permita a otras personas contribuir a resolver el mismo problema.

En la década de 1990, Gerald Combs estaba trabajando en un proveedor de servicios de Internet y comenzó a escribir su propia herramienta de análisis de red porque las herramientas similares eran muy caras en aquel entonces. Hasta hoy, más de 600 personas han contribuido al proyecto llamado Wireshark. Ahora a menudo se considera mejor que la oferta comercial y gracias a ello Gerald formó una empresa para dar soporte a Wireshark y vender productos y apoyo que facilitan su uso. Más adelante la empresa la compró un importante proveedor de red que apoya su desarrollo.

Otras compañías obtienen valor tan inmenso del software de código abierto que se consideran eficaz contratar a personas para trabajar en el software a tiempo completo. El motor de búsqueda de Google contrató al creador del lenguaje Python, e incluso Linus Torvalds fue contratado por la Linux Foundation para trabajar en Linux. La compañía de teléfonos estadounidense AT&T obtiene tal valor de los proyectos de Ruby y Rails para sus Páginas Amarillas, que tienen un empleado que no hace nada sino trabajar en estos proyectos.

La última manera en la que la gente hace dinero indirectamente a través de código abierto, es que es una forma abierta para calificar las habilidades propias. Una cosa es decir que realizas ciertas tareas en tu trabajo, pero mostrar tu capacidad de creación y compartirlo con el mundo permite a los empleadores potenciales ver la calidad de tu trabajo. Del mismo modo, las empresas se han dado cuenta que proporcionar concesiones de las partes no críticas de su código abierto de software interno, atrae el interés de la gente de mayor calibre.

## NDG Linux Essentials: Capítulo 3: El uso en Linux

### 3.1 Introducción

Antes de que te puedas convertir en un administrador eficaz de los sistemas Linux, debes saber utilizar Linux como tu escritorio y tener aptitudes con las habilidades básicas de la Tecnología de Información y Comunicación (TIC). No sólo eso te ayudará al tratar con usuarios, sino sumergiéndote en el Linux te ayudará a mejorar tus habilidades más rápidamente. Además, la vida de un administrador de sistemas es más que un trabajo en el servidor - ¡hay también correo electrónico y documentación para hacer!

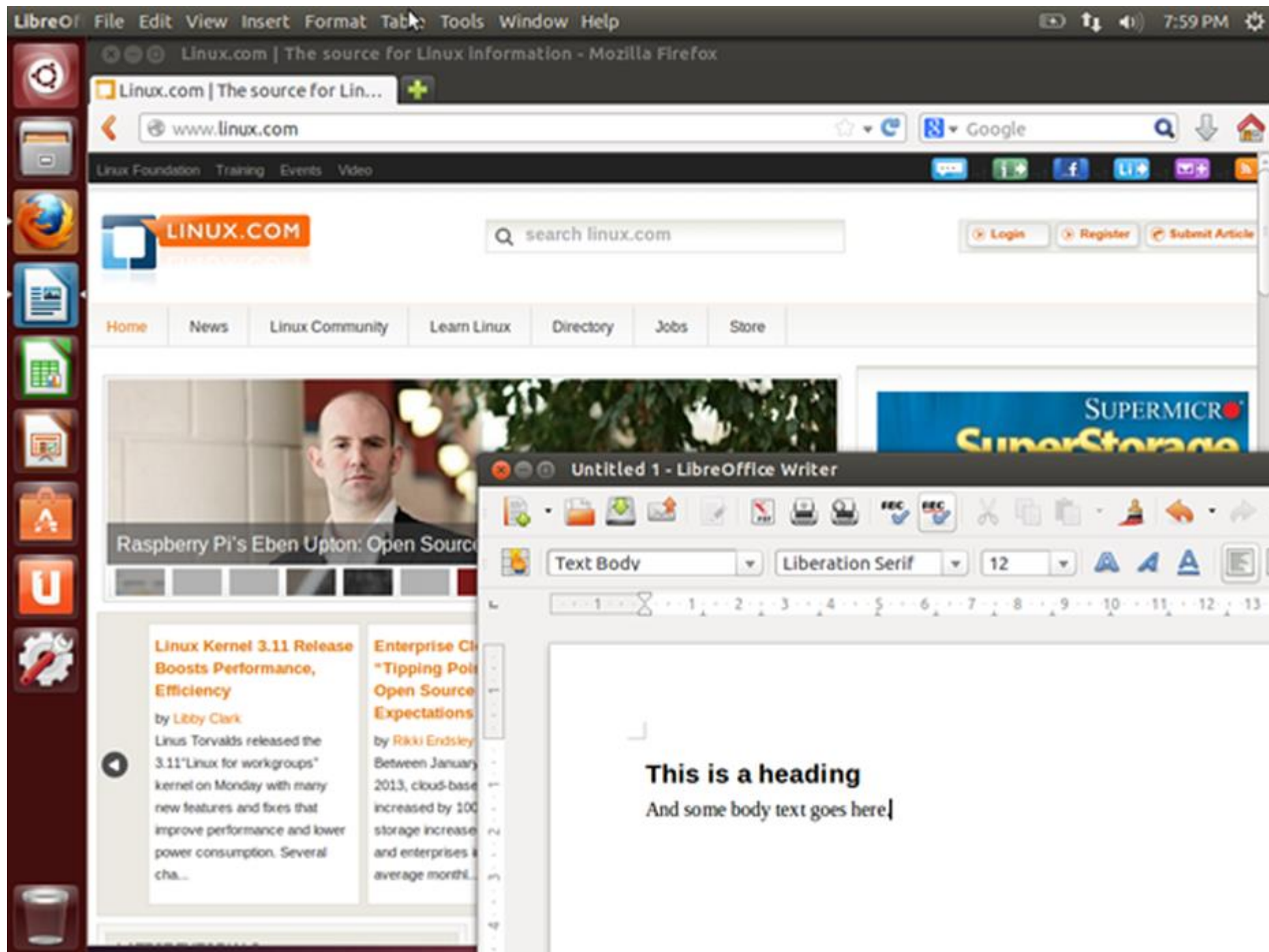
*¿Cuál es la mejor posición de empleo de Linux que los Gerentes de Reclutamiento de TI están buscando?*

*Administradores de Sistemas*

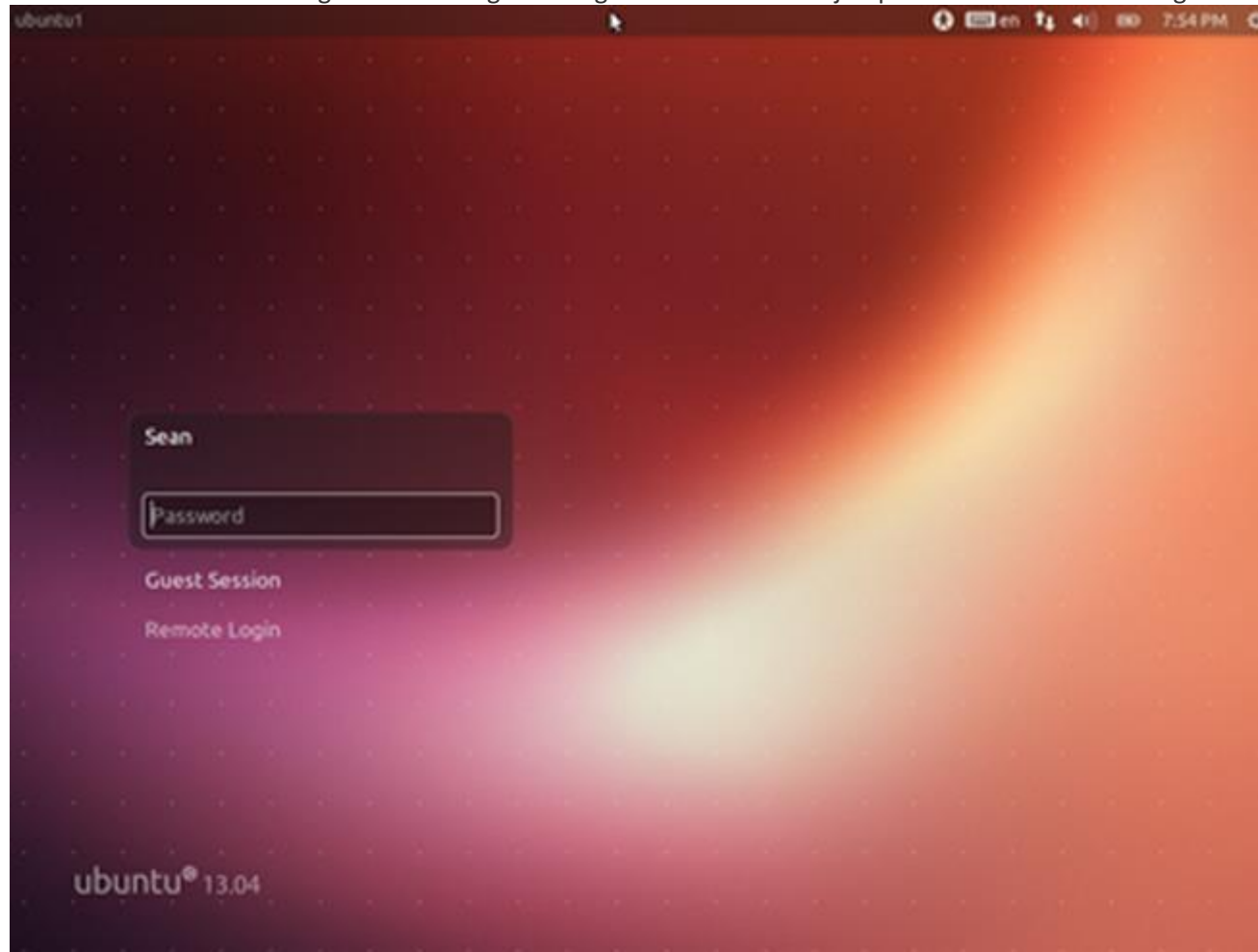
*- Reporte Laboral de Linux 2013, Linux Foundation & Dice*

### 3.2 Modo Gráfico vs. No Gráfico

Linux puede usarse de dos maneras: en modo gráfico y modo no gráfico. En modo gráfico las aplicaciones corren en las ventanas que puedes cambiar el tamaño y mover. Tienes menús y herramientas que te ayudan a encontrar lo que buscas. Aquí es donde vas a usar un navegador web, tus herramientas de edición de gráficos y tu correo electrónico. Aquí vemos un ejemplo del escritorio gráfico, con una barra de menús de aplicaciones populares en la izquierda y un documento de LibreOffice editado con un navegador web en el fondo.



En modo gráfico puedes tener varios shells abiertos, que resulta muy útil cuando se están realizando tareas en múltiples equipos remotos. Incluso puedes iniciar la sesión con tu usuario y contraseña a través de una interfaz gráfica. En la siguiente figura se muestra un ejemplo de un inicio de sesión gráfico.



Después de iniciar la sesión pasarás al escritorio donde puedes cargar las aplicaciones. El modo no gráfico comienza con una sesión basada en texto que se muestra a continuación. Simplemente se te pedirá tu nombre de usuario y luego tu contraseña. Si el inicio de sesión tiene éxito pasarás directamente al shell.

```

Ubuntu 13.04 ubuntu1 tty2
ubuntu1 login:

```

En el modo no gráfico no hay ventanas para navegar. A pesar de esto tienes editores de texto, navegadores web y clientes de correo electrónico, pero son sólo de texto. De este modo UNIX empezó antes que los entornos gráficos fueran la norma. La mayoría de los servidores también se ejecutarán en este modo, ya que la gente no entra en ellos directamente, lo que hace que una interfaz gráfica sea un desperdicio de recursos. Aquí hay un ejemplo de la pantalla que se puede ver después de iniciar la sesión.

```

Ubuntu 13.04 ubuntu1 tty2
ubuntu1 login: sean
Password:

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Welcome to Ubuntu 13.04 (GNU/Linux 3.8.0-19-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

212 packages can be updated.
91 updates are security updates.

sean@ubuntu1:~$ w
 20:08:35 up 14 min,  2 users,  load average: 0.45, 0.44, 0.32
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU WHAT
sean      tty2                20:08    14:35   0.05s  0.00s w
sean      tty7                19:54    14:35   1:08   0.16s gnome-session -
sean@ubuntu1:~$ _

```

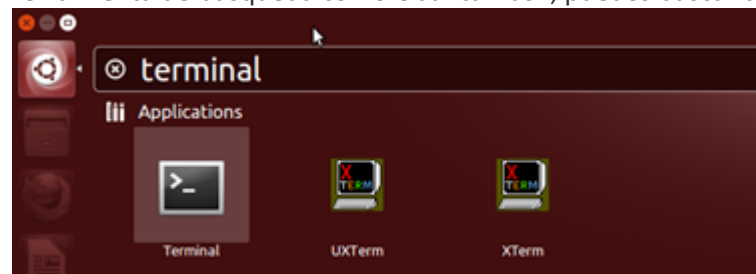


Puedes ver el mensaje original para entrar en la parte superior con el texto más reciente añadido a continuación. Durante el inicio de sesión podrías ver algunos mensajes, llamados **el mensaje del día (MOTD)**, que es una oportunidad para que el administrador de sistemas para pasar información a los usuarios. El MOTD **es el símbolo del sistema**. En el ejemplo anterior, el usuario introdujo el comando `w` que muestra quién está conectado. De manera que son introducidos y procesados los comandos nuevos, la ventana se desplaza hacia arriba y el texto más antiguo se pierde en la parte superior. La terminal es responsable de mantener cualquier historia, tal como para permitir al usuario desplazarse hacia arriba y ver los comandos introducidos. En cuanto a Linux, lo que está en la pantalla es todo lo que hay. No hay nada para navegar.

### 3.3 Línea de Comandos

La línea de comandos es una entrada de texto simple, **que te permite ingresar cualquier cosa, desde un comando de una sola palabra hasta scripts complicados**. Si inicias la sesión a través de modo de texto te encuentras inmediatamente en la consola. Si inicias la sesión de forma gráfica, entonces necesitarás iniciar un shell gráfico, que es solo una consola de texto con una ventana a su alrededor para que puedas cambiar su tamaño y posición.

Cada escritorio de Linux es diferente, por lo que tienes que buscar en tu menú una opción llamada **terminal o x-term**. Las dos son shells gráficos, diferenciadas sobre todo en aspectos más que funcionalidad. Si tienes una herramienta de búsqueda como Ubuntu Dash, puedes buscar un **terminal** como se muestra aquí.



Estas herramientas te permiten buscar rápidamente en tu sistema exactamente lo que quieres ejecutar en lugar de perderte en los menús.

### 3.4 Virtualización y Cloud Computing

Linux **es un sistema operativo multiusuario**, lo que significa que muchos usuarios diferentes pueden trabajar en el mismo sistema al mismo tiempo y en su mayor parte **no pueden hacer cosas para dañar a otros usuarios**. Sin embargo, esto tiene limitaciones: **los usuarios pueden acaparar el espacio en disco o tomar demasiada memoria o recursos de la CPU y causar que el sistema sea lento para todos**. Compartir el sistema en modo multiusuario también requiere que cada uno ejecute en modo de usuarios sin privilegios, por lo que permitir que cada usuario ejecute su propio servidor web es muy difícil.

La virtualización es un **proceso donde un equipo físico, llamado host, ejecuta múltiples copias de un sistema operativo, cada una llamada invitado**. El host **ejecuta un software llamado hipervisor que cambia el control entre los diferentes invitados, tal como el kernel de Linux funciona para los procesos individuales**.

La **virtualización funciona** porque los servidores pasan la mayor parte de su tiempo **inactivo y no necesitan recursos físicos tales como un monitor y un teclado**. Ahora puedes tomar una potente CPU y difundirla alrededor de varias máquinas virtuales y mantener una distribución más equitativa entre los invitados de lo que es posible en un sistema de Linux de puro. La **principal limitación es por lo general la memoria, con los avances en la tecnología de hipervisor y la CPU es posible poner más máquinas virtuales en un host que nunca**.

En un entorno virtualizado **un host puede ejecutar docenas de sistemas operativos invitados**, y con el apoyo de la CPU, los invitados no saben que se están ejecutando en una máquina virtual. Cada invitado obtiene su propia CPU, RAM y disco virtual y se comunica con la red. Ni siquiera es necesario ejecutar el mismo sistema operativo en todos los invitados, lo que reduce aún más el número de **servidores físicos necesarios**.

La virtualización ofrece una manera para que una empresa reduzca su consumo de energía y espacio de centro de datos frente a una flota equivalente de servidores físicos. Los invitados ahora sólo son configuraciones de software, así que es fácil proporcionar una nueva máquina para una prueba y destruirla cuando haya pasado su utilidad.

Si es posible ejecutar varias instancias de un sistema operativo en una máquina física y conectarse en la red, entonces la ubicación de la máquina no importa. El Cloud Computing (*Cómputo o Informática en la Nube*) toma este enfoque y te permite tener una máquina virtual en un centro de datos remoto que no posees y sólo pagas por los recursos que utilizas. Los proveedores de Cloud Computing pueden tomar ventaja de las economías de escala para ofrecer recursos de computación a mejores precios de lo que costaría adquirir tu propio hardware, espacio y enfriamiento.

Los servidores virtuales sólo son una faceta de Cloud Computing. También puedes obtener almacenamiento de archivos, bases de datos o incluso software. La clave en la mayoría de estos productos es que pagas por lo que usas, por ejemplo una cierta cantidad por giga bytes de datos por mes, en lugar de comprar el hardware y el software para darle hospedaje tu mismo. Algunas situaciones son más adecuadas para la nube que otros. Preocupaciones de seguridad y el rendimiento son generalmente los primeros elementos que surgen seguidos por el costo y la funcionalidad.

Linux juega un papel fundamental en el Cloud Computing. La mayoría de los servidores virtuales se basa en algún tipo de kernel de Linux, y Linux se suele utilizar para alojar las aplicaciones detrás de los servicios del Cloud Computing.

### 3.5 Utilizar Linux para el Trabajo

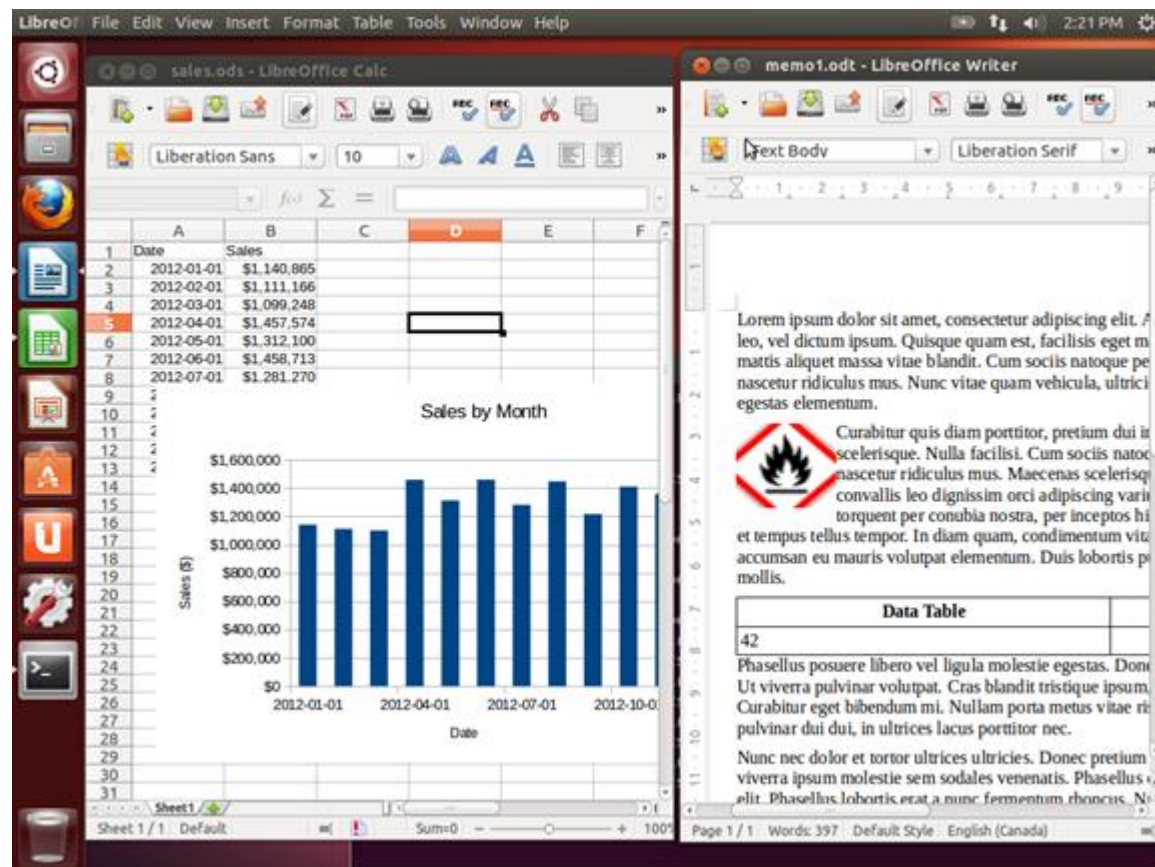
Las herramientas básicas utilizadas en la mayoría de las oficinas son:

- Procesador de textos
- Hoja de cálculo
- Paquete de presentación
- Navegador web

OpenOffice, o el más activo, LibreOffice, se encarga de las tres primeras funciones. El procesador de texto se utiliza para editar documentos, tales como informes y memos. Las Hojas de cálculo son útiles para trabajar con números, por ejemplo para resumir datos de ventas y hacer predicciones futuras. El paquete de presentación se utiliza para crear diapositivas con las características tales como texto, gráficos y vídeo insertado. Las diapositivas pueden ser impresas o mostradas en una pantalla o un proyector para compartir con una audiencia.

A continuación abajo puedes ver la hoja de cálculo y editor de documentos de LibreOffice. Nota cómo la hoja de cálculo, LibreOffice Calc, no se limita a filas y columnas de números. Los números pueden ser la fuente de un gráfico, y puedes escribir fórmulas para calcular valores basados en la información, por ejemplo reunir las tasas de interés y cantidades para ayudar a comparar las diferentes opciones de préstamo.

Utilizando el Writer de LibreOffice, un documento puede contener texto, gráficos, tablas de datos y mucho más. Puedes vincular documentos y hojas de cálculo, por ejemplo, para que puedas resumir los datos en forma escrita y saber que cualquier cambio en la hoja de cálculo se reflejará en el documento.



LibreOffice también puede trabajar con otros formatos de archivo, como Microsoft Office o **Adobe Portable Document Format (PDF)**. Además, mediante el uso de extensiones, se puede integrar LibreOffice con el software Wiki para ofrecerle una poderosa solución de intranet.

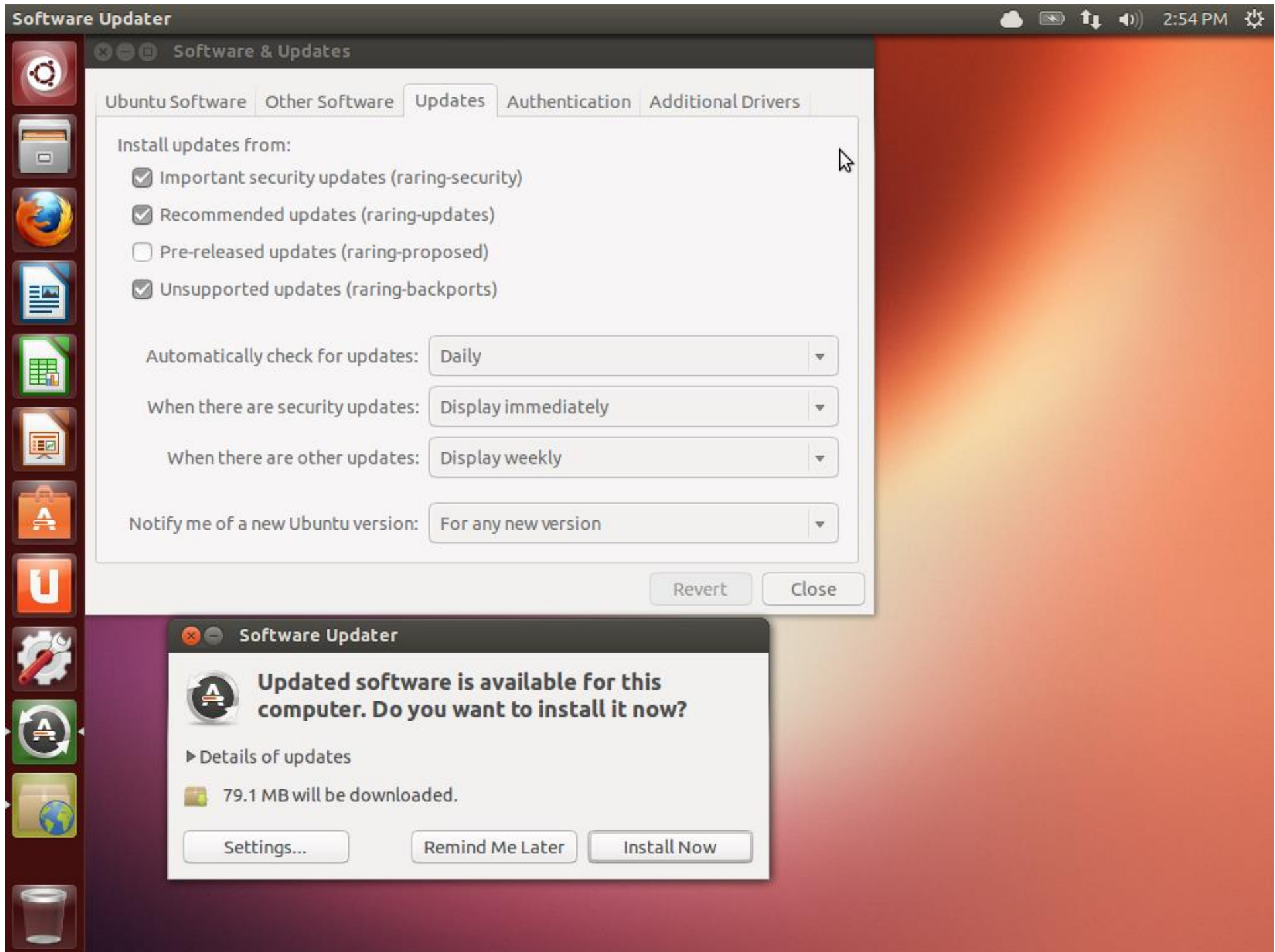
Linux es un ciudadano de primera clase para los navegadores Firefox y Google Chrome. Como tal, puede esperar tener el software más reciente disponible para su plataforma y el acceso oportuno a correcciones de errores y nuevas características. Algunos complementos, como Adobe Flash, no siempre funcionan correctamente ya que dependen de otra compañía con prioridades diferentes.

### 3.6 Proteger tu Equipo Linux

A Linux no le importa si estás en el teclado de un equipo o conectado a través de Internet, por lo que querrás tomar algunas precauciones básicas para asegurarte de que tus datos están a salvo.

Lo más fácil es utilizar una buena y única contraseña donde quiera que vayas, sobre todo en tu máquina local. Una buena contraseña tiene al menos 10 caracteres y contiene una mezcla de números y letras (tanto mayúsculas y minúsculas) y símbolos especiales. Utiliza un paquete como **KeePassX** para generar contraseñas, ya que luego sólo necesitas tener una contraseña de inicio de sesión a tu equipo y una contraseña para abrir el archivo de KeePassX.

Después de eso, crea periódicamente un punto de comprobación de actualizaciones. A continuación, te mostraremos la configuración de actualización de software Ubuntu, que está disponible en el menú de **Configuración**.



En la parte superior, se puede ver que el sistema está configurado para buscar actualizaciones de forma diaria. Si hay actualizaciones relacionadas con la seguridad, entonces se te pedirá que las instales inmediatamente. De lo contrario, recibirás las actualizaciones en lotes cada semana. En la parte inferior de la pantalla puedes ver el cuadro de diálogo que aparece cuando hay actualizaciones disponibles. Todo lo que tienes que hacer es hacer clic en **Instalar ahora** y tu equipo se actualizará!

Por último, tienes que proteger tu equipo de aceptar conexiones entrantes. *Firewall* es un dispositivo que filtra el tráfico de red y Linux tiene uno integrado. Si usas Ubuntu, **gufw** es una interfaz gráfica para "Uncomplicated firewall" de Ubuntu.



Firewall Configuration

Software & Updates

sean@ubuntu1: ~

```

unix 2 [ ACC ] STREAM LISTENING 10636 1649/dbus-daemon @/tmp/dbus-12CE5p
48FS
unix 2 [ ACC ] STREAM LISTENING 10874 - /run/user/sean/ke
yring-0pS3u1/gpg
unix 2 [ ACC ] STREAM LISTENING 8458 - /tmp/.X11-unix/X0
unix 2 [ ACC ] STREAM LISTENING 11396 - /run/user/sean/ke
yring-0pS3u1/ssh
unix 2 [ ACC ] /var/run/acpid.so
unix 2 [ ACC ] /run/udev/control
unix 2 [ ACC ] /run/user/sean/ke
yring-0pS3u1/pkcs11
unix 2 [ ACC ] /run/user/sean/ke
yring-0pS3u1/control
unix 2 [ ACC ] lseaudio /tmp/pulse-2L9K88
unix 2 [ ACC ] lseaudio /run/user/sean/pu
eMLGn7/native /var/run/dbus/sys
unix 2 [ ACC ] tem_bus_socket fsd-burn @/dbus-vfs-daemon
unix 2 [ ACC ] /socket-Vp49tv0x /var/run/avahi-da
emon/socket
sean@ubuntu1:~$ netstat
(Not all processes could
will not be shown, you
tcp 0 0 127.0.0.1:22 LISTENING
tcp 0 0 127.0.0.1:22 LISTENING
sean@ubuntu1:~$

```

**Firewall**

Status: ☒ ON

Incoming: Deny

Outgoing: Allow

**Rules**

| To  | Action | From |
|-----|--------|------|
| + - |        |      |

**Listening Report**

| Protocol | Port  | Address | Application  |
|----------|-------|---------|--------------|
| UDP      | 42652 | *       | dhclient     |
| UDP      | 43648 | *       | avahi-daemon |
| UDP      | 5353  | *       | avahi-daemon |

[Donate?](#) ✕



Simplemente cambiando el estado a "on" se bloquea todo el tráfico que llega a tu equipo, a menos que lo hayas iniciado tú mismo. De manera selectiva puedes permitir que entren algunas cosas haciendo clic en el signo más.

Bajo el capó estás usando *iptables* que es el sistema firewall integrado. En lugar de introducir comandos iptables complicados, usas un GUI. Mientras que este GUI te permite construir una política efectiva de un escritorio, éste apenas araña la superficie de lo que se puede hacer con iptables.

### 3.7 Protegerte a tí Mismo

Cuando navegas por Internet, **dejas una huella digital**. Mucha de esta información viene ignorada, pero alguna viene reunida para recopilar estadísticas de publicidad y otra puede ser utilizada para propósitos maliciosos.

Como regla general, no deberías confiar en los sitios con los que interactúas. Usa contraseñas diferentes en cada sitio de Internet para que si tal sitio web estuviera hackeado, la contraseña no podría utilizarse para obtener acceso a otros sitios. Usando anteriormente mencionado KeePassX es la forma más fácil de hacerlo. De la misma forma, limita la información que proporcionas a los sitios, sólo lo imprescindible. Mientras que dar el nombre de tu madre y fecha de nacimiento podría ayudarte a desbloquear tus credenciales para la red social en caso de que pierdas tu contraseña, la misma información puede utilizarse para suplantar la identidad de tu banco.

Las *cookies* son el mecanismo principal que los sitios web utilizan para darte seguimiento. A veces este seguimiento es bueno, por ejemplo para dar seguimiento de lo que está en tu cesta de compras o para mantenerte conectado cuando regreses al sitio.

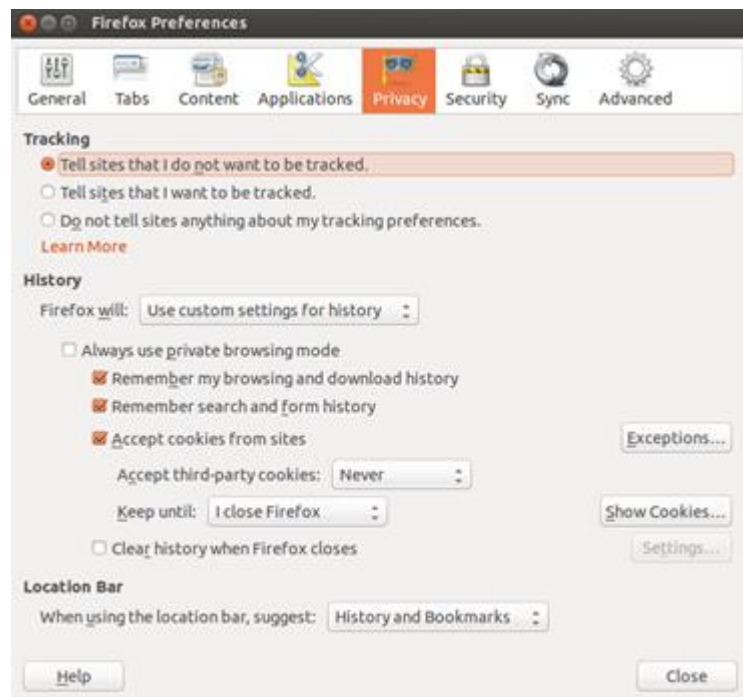
Cuando navegas por la web, un servidor web puede devolver la cookie que es un pequeño trozo de texto junto con la página web. Tu navegador lo almacena y envía con cada solicitud al mismo sitio. No envías cookies para ejemplo.com a sitios en ejemplo.org.

Sin embargo, muchos sitios han incrustado scripts que provienen de terceros, como un mensaje emergente de anuncio o un píxel de analítica. Si ejemplo.com y ejemplo.org tienen un píxel de analítica, por ejemplo de un anunciante, entonces esa misma cookie se enviará al navegar por ambos sitios. El anunciante se entera entonces que has visitado ejemplo.com y ejemplo.org.

Con un alcance suficientemente amplio, como los "Likes" y botones parecidos, un sitio web puede obtener un entendimiento de cuáles sitios web visitas y averiguar tus intereses y datos demográficos.

Existen diversas estrategias para tratar este asunto. Uno es ignorarlo. La otra es limitar los píxeles de seguimiento que aceptas, ya sea por bloqueo completo o vaciarlos periódicamente. A continuación abajo se muestra la configuración de cookies para Firefox. En la parte superior, verás que el usuario ha optado que Firefox no de permiso al sitio para el seguimiento. Esta es una etiqueta voluntaria enviada en la petición que algunos sitios distinguirán. A continuación, el navegador recibe una instrucción de no recordar nunca las cookies de terceros y eliminar cookies regulares (por ejemplo, desde el sitio navegando) después de haber cerrado el Firefox.

Afinando la configuración de privacidad puede hacerte más anónimo en Internet, pero también puede causar problemas con algunos sitios que dependen de cookies de terceros. Si esto sucede, probablemente tengas que permitir explícitamente que se guarden algunas cookies.



Aquí también tendrás la posibilidad de olvidar el historial de búsqueda o no seguirlo. Con el historial de búsqueda eliminado no habrá ningún registro en el equipo local de los sitios que hayas visitado.

Si te preocupa mucho ser anónimo en Internet, puedes descargar y utilizar el **Navegador Tor**. Tor es el nombre corto para "The Onion Router" que es una red de servidores públicamente ejecutados que rebotan tu tráfico para ocultar el origen. El navegador que viene con el paquete es una versión básica que no ejecuta ni siquiera las secuencias de comandos, por lo que algunos sitios probablemente no funcionarán correctamente. Sin embargo, **es la mejor manera de ocultar tu identidad si es lo que deseas hacer.**

## NDG Linux Essentials: Capítulo 4: Competencias en Línea de Comandos

### 4.1 Introducción

Si eres como la mayoría de las personas, probablemente estés familiarizado con el uso de la *Interfaz Gráfica de Usuario* (o GUI «Graphical User Interface») para controlar tu computadora. Fue introducida a las masas por Apple en la computadora Macintosh y popularizado por Microsoft. La GUI proporciona una forma fácil de descubrir y administrar tu sistema. Sin una GUI, algunas herramientas para gráficos y video no serían prácticas.

Antes de la popularidad de la GUI, la *Interfaz de Línea de Comandos* (o CLI «Command Line Interface») era la forma preferida para controlar una computadora. La CLI se basa únicamente en la entrada por teclado. Todo lo que quieres que tu computadora haga, se retransmite escribiendo comandos en lugar de ir haciendo clics en los iconos.

Si nunca has usado una CLI, al principio puede resultar difícil porque requiere de memorizar comandos y sus *opciones*. Sin embargo, la CLI proporciona un control más preciso, una mayor velocidad y capacidad para automatizar fácilmente las tareas a través del scripting (ver barra lateral). Aunque Linux tiene muchos entornos GUI, podrás controlar Linux mucho más eficazmente mediante la Interfaz de Línea de Comandos.

*¿Por qué conocer la línea de comando es importante? ¡Flexibilidad y Movilidad! Mediante la comprensión de los fundamentos de Linux, tienes la capacidad de trabajar en CUALQUIER distribución de Linux. Esto podría significar una compañía con ambiente mixto o una nueva empresa con una distribución Linux diferente.*

#### 4.2 Interfaz de Línea de Comandos (CLI)

La *Interfaz de Línea de Comandos (CLI)*, es una interfaz basada en texto para la computadora, donde el usuario introduce un comando y la computadora lo ejecuta. El entorno de la CLI es proporcionado por una aplicación en la computadora conocida como un terminal.

El terminal acepta lo que el usuario escribe y lo pasa a un *shell*. El shell interpreta lo que el usuario ha introducido en las instrucciones que se pueden ejecutar con el sistema operativo. Si el comando produce una salida, entonces se muestra este texto en el terminal. Si surgen problemas con el comando, se muestra un mensaje de error.

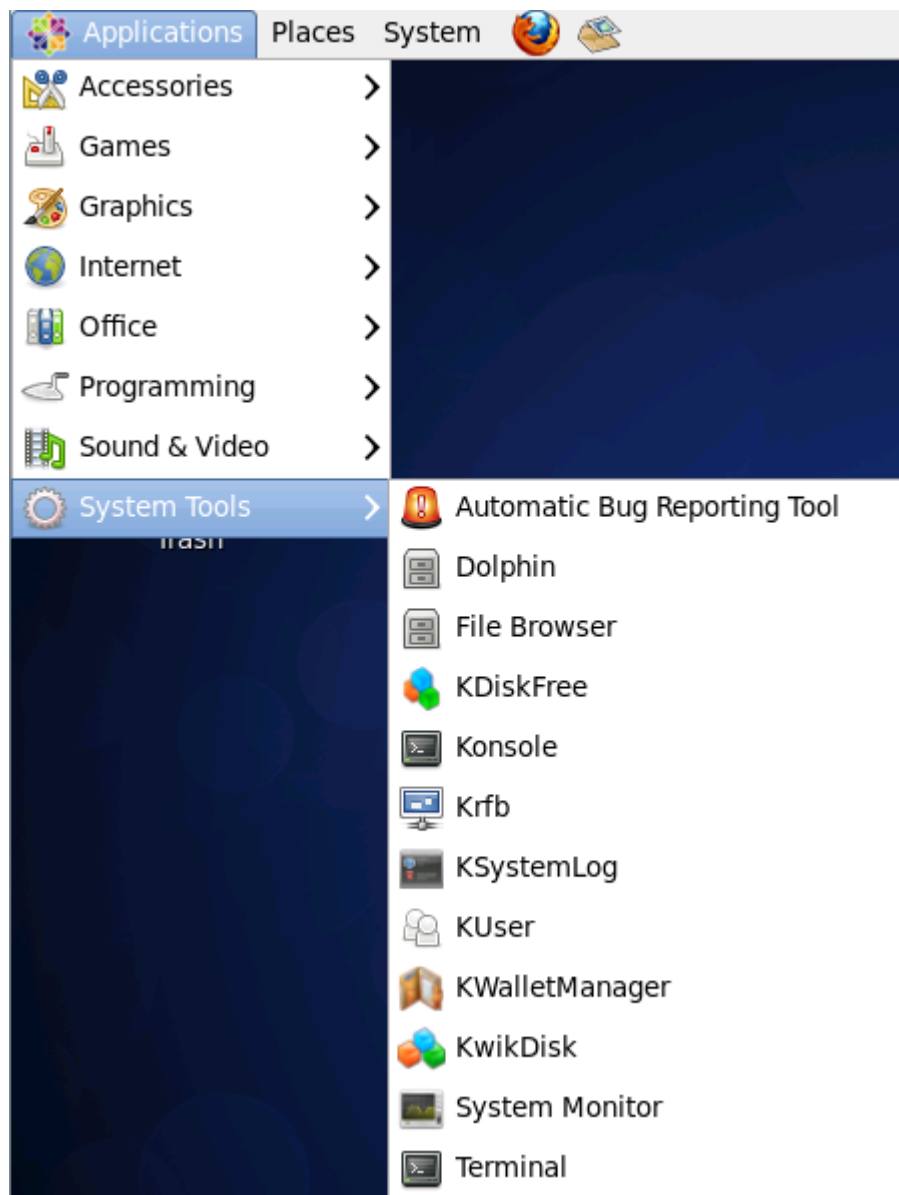
#### 4.3 Acceso a la Terminal

Hay muchas maneras de acceder a la ventana de la terminal. Algunos sistemas arrancarán directamente a la terminal. Este suele ser el caso de los servidores, ya que una interfaz gráfica de usuario (GUI) puede requerir muchos recursos que no son necesarios para realizar operaciones basadas en servidores.

Un buen ejemplo de un servidor que no requiere una GUI es un servidor web. Los servidores web deben correr tan rápido como sea posible y una GUI sólo haría lento el sistema.

En los sistemas que arrancan con una GUI, hay comúnmente dos formas de acceder a una terminal, una terminal basado en GUI y un terminal virtual:

- Una terminal de GUI es un programa dentro del entorno de una GUI que emula la ventana de la terminal. Las terminales de la GUI pueden accederse a través del sistema de menú. Por ejemplo, en una máquina CentOS, puedes hacer clic en **Applications** (o «Aplicaciones» en español) en la barra de menús, luego en **System Tools >** (o «Herramientas de Sistema») y, finalmente, en **Terminal**:



Una terminal virtual puede ejecutarse al mismo tiempo que una GUI, pero requiere que el usuario se conecte o inicie sesión a través de la terminal virtual antes de que pueda ejecutar los comandos (como lo haría antes de acceder a la interfaz GUI). La mayoría de los sistemas tienen múltiples terminales virtuales que se pueden acceder pulsando una combinación de teclas, por ejemplo: **Ctrl-Alt-F1**

**Nota:** En las máquinas virtuales puede que las terminales virtuales no estén disponibles.

#### 4.3.1 Prompt

Una ventana del terminal muestra un prompt (o «símbolo o aviso» en español); el prompt aparece cuando no se ejecutan ningún comando y cuando la salida completa del comando se ha desplegado en la pantalla. El prompt está diseñado para decirle al usuario que introduzca un comando.

La estructura del prompt puede variar entre las distribuciones, pero por lo general contiene información sobre el usuario y el sistema. A continuación te mostramos una estructura común de un prompt:

```
sysadmin@localhost:~$
```

El prompt anterior proporciona el nombre del usuario registrado en (`sysadmin`), el nombre del sistema (`localhost`) y el directorio actual (`~`). El símbolo `~` se utiliza como abreviación para el directorio principal del usuario (el directorio principal del usuario viene bajo el directorio `/home` (o «inicio» en español) y con el nombre de la cuenta de usuario, por ejemplo: `/home/sysadmin`).

#### 4.3.2 El Shell

Un shell es el intérprete que traduce los comandos introducidos por un usuario en acciones a realizar por el sistema operativo. El entorno Linux proporciona muchos tipos diferentes de shells, algunos de los cuales han existido por muchos años.

El shell más comúnmente utilizado para las distribuciones de Linux se llama el BASH shell. Es un shell que ofrece muchas funciones avanzadas, tales como el historial de comandos, que te permite fácilmente volver a ejecutar comandos previamente ejecutados.

El BASH shell tiene también otras funciones populares:

- **Scripting:** La capacidad de colocar los comandos en un archivo y ejecutar el archivo, resultando en todos los comandos siendo ejecutados. Esta función también tiene algunas características de programación, tales como las instrucciones condicionales y la habilidad de crear funciones (AKA, subrutinas).
- **Los Alias:** La habilidad de crear "nicknames" (o «sobrenombres» en español) cortos para más comandos más largos.
- **Las Variables:** Las Variables se utilizan para almacenar información para el BASH shell. Estas variables pueden utilizarse para modificar cómo las funciones y los comandos trabajan y proporcionan información vital sobre el sistema.

**Nota:** La lista anterior es sólo un breve resumen de algunas de las muchas funciones proporcionadas por el BASH shell.

#### 4.3.3 Los Comandos de Formato

Muchos comandos se pueden utilizar por sí mismos sin más entradas. Algunos comandos requieren entradas adicionales para funcionar correctamente. Esta entrada adicional viene en dos formas: *opciones* y *argumentos*.

El formato típico de un comando es el siguiente:

```
comando [opciones] [argumentos]
```

Las opciones se utilizan para modificar el comportamiento básico de un comando y los argumentos se utilizan para proporcionar información adicional (tal como un nombre de archivo o un nombre de usuario). Cada opción y argumento vienen normalmente separados por un espacio, aunque las opciones pueden a menudo ser combinadas.

Recuerda que Linux es sensible a mayúsculas y minúsculas. Comandos, opciones, argumentos, variables y nombres de archivos deben introducirse exactamente como se muestra.

El comando `ls` proporciona ejemplos útiles. Por sí mismo, el comando `ls` listará los archivos y directorios contenidos en el directorio de trabajo actual:

```
sysadmin@localhost:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
sysadmin@localhost:~$
```

Sobre el comando `ls` hablaremos en detalle en un capítulo posterior. El propósito de introducir este comando ahora, es demostrar cómo los argumentos y las opciones funcionan. En este punto no te debes preocupar de lo que es la salida del comando, más bien centrarte en comprender en qué es un argumento y una opción.

Un argumento **lo puedes pasar también al comando** `ls` para especificar contenido de qué directorio hay que listar. Por ejemplo, el comando `ls /etc/ppp` listará el contenido del directorio `/etc/ppp` en lugar del directorio actual:

```
sysadmin@localhost:~$ ls /etc/ppp
ip-down.d  ip-up.d
sysadmin@localhost:~$
```

Puesto que el comando `ls` **acepta múltiples** argumentos, puede listar el contenido de varios directorios a la vez, introduciendo el comando `ls /etc/ppp /etc/ssh:`

```
sysadmin@localhost:~$ ls /etc/ppp /etc/ssh
/etc/ppp:
ip-down.d  ip-up.d
/etc/ssh:
moduli      ssh_host_dsa_key.pub  ssh_host_rsa_key      sshd_configssh_config
ssh_host_ecdsa_key  ssh_host_rsa_key.pub
ssh_host_dsa_key    ssh_host_ecdsa_key.pub  ssh_import_id
sysadmin@localhost:~$
```

#### 4.3.4 Trabajando con las Opciones

Las opciones **pueden utilizarse con comandos para ampliar o modificar el comportamiento de un comando**. Las opciones **son a menudo de una letra**; sin embargo, a veces serán "palabras". Por lo general, los comandos viejos utilizan una letra, mientras los comandos nuevos utilizan palabras completas para las opciones. Opciones de una letra son precedidas por un único guión `-`. Opciones de palabra completa son precedidas por dos guiones `--`.

Por ejemplo, puedes utilizar la opción `-l` con el comando `ls` para ver más información sobre los archivos que se listan. El comando `ls -l` lista los archivos contenidos dentro del directorio actual y proporciona información adicional, tal como los permisos, el tamaño del archivo y otra información:

```
sysadmin@localhost:~$ ls -l
total 0
```



```
drwxr-xr-x 1 sysadmin sysadmin 0 Jan 29 2015 Desktop
drwxr-xr-x 1 sysadmin sysadmin 0 Jan 29 2015 Documents
drwxr-xr-x 1 sysadmin sysadmin 0 Jan 29 2015 Downloads
drwxr-xr-x 1 sysadmin sysadmin 0 Jan 29 2015 Music
drwxr-xr-x 1 sysadmin sysadmin 0 Jan 29 2015 Pictures
drwxr-xr-x 1 sysadmin sysadmin 0 Jan 29 2015 Public
drwxr-xr-x 1 sysadmin sysadmin 0 Jan 29 2015 Templates
drwxr-xr-x 1 sysadmin sysadmin 0 Jan 29 2015 Videos
sysadmin@localhost:~$
```

En la mayoría de los casos, las opciones **pueden utilizarse conjuntamente con otras** opciones. Por ejemplo, los comandos `ls -l -h` o `ls -lh` listarán los archivos con sus detalles, pero se mostrará el tamaño de los archivos en formato de legibilidad humana en lugar del valor predeterminado (bytes):

```
sysadmin@localhost:~$ ls -l /usr/bin/perl
-rwxr-xr-x 2 root root 10376 Feb 4 2014 /usr/bin/perl
sysadmin@localhost:~$ ls -lh /usr/bin/perl
-rwxr-xr-x 2 root root 11K Feb 4 2014 /usr/bin/perl
sysadmin@localhost:~$
```

Nota que el ejemplo anterior también demostró cómo se pueden combinar opciones de una letra: `-lh`. El orden de las opciones combinadas no es importante.

La opción `-h` también tiene la forma de una palabra completa: `--human-readable` (*--legibilidad-humana*).

Las opciones a menudo pueden utilizarse con un argumento. De hecho, algunas de las opciones requieren sus propios argumentos. Puedes utilizar los argumentos y las opciones con el comando `ls` para listar el contenido de otro directorio al ejecutar el comando `ls -l/etc/ppp`:

```
sysadmin@localhost:~$ ls -l /etc/ppp
total 0
drwxr-xr-x 1 root root 10 Jan 29 2015 ip-down.d
drwxr-xr-x 1 root root 10 Jan 29 2015 ip-up.d
sysadmin@localhost:~$
```

#### 4.4 Historial de los Comandos

Al ejecutar un comando en una terminal, el comando se almacena en "history list" (o «lista de historial» en español). Esto está diseñado para que más adelante puedas ejecutar el mismo comando más fácilmente puesto que no necesitarás volver a introducir el comando entero.

Para ver la lista de historial de una terminal, utiliza el comando `history` (o «historial» en español):

```
sysadmin@localhost:~$ date
Sun Nov  1 00:40:28 UTC 2015

sysadmin@localhost:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates
Videos

sysadmin@localhost:~$ cal 5 2015

    May 2015
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31

sysadmin@localhost:~$ history

 1  date
 2  ls
 3  cal 5 2015
 4  history

sysadmin@localhost:~$
```

Pulsando la tecla de **Flecha Hacia Arriba** ↑ se mostrará el comando anterior en tu línea de prompt. Puedes presionar arriba repetidas veces para moverte a través del historial de comandos que hayas ejecutado. Presionando la tecla **Entrar** se ejecutará de nuevo el comando visualizado.

Cuando encuentres el comando que quieres ejecutar, puedes utilizar las teclas de **Flecha Hacia Izquierda** ← y **Flecha Hacia Derecha** → para colocar el cursor para edición. Otras teclas útiles para edición incluyen **Inicio**, **Fin**, **Retroceso** y **Suprimir**.

Si ves un comando que quieres ejecutar en la lista que haya generado el comando `history`, puedes ejecutar este comando introduciendo el signo de exclamación y luego el número al lado del comando, por ejemplo:

```
!3

sysadmin@localhost:~$ history
```

```

1 date
2 ls
3 cal 5 2015
4 history

sysadmin@localhost:~$ !3
cal 5 2015

    May 2015

Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
                31

sysadmin@localhost:~$

```

Algunos ejemplos adicionales del `history`:

| Ejemplo                | Significado  |
|------------------------|--|
| <code>history 5</code> | Muestra los últimos cinco comandos de la lista del historial               |
| <code>!!</code>        | Ejecuta el último comando otra vez   |
| <code>!-5</code>       | Ejecuta el quinto comando desde la parte inferior de la lista de historial |
| <code>!ls</code>       | Ejecuta el comando <code>ls</code> más reciente                            |

#### 4.5 Introduciendo las variables del BASH shell

Una variable del shell BASH es una función que te permite a ti o al shell almacenar los datos. Esta información puede utilizarse para proporcionar información crítica del sistema o para cambiar el comportamiento del funcionamiento del shell BASH (u otros comandos).

Las variables reciben nombres y se almacenan temporalmente en la memoria. Al cerrar una ventana de la terminal o shell, todas las variables se pierden. Sin embargo, el sistema automáticamente recrea muchas de estas variables cuando se abre un nuevo shell.

Para mostrar el valor de una variable, puedes utilizar el comando `echo` (o «eco» en español). El comando `echo` se utiliza para mostrar la salida en la terminal; en el ejemplo siguiente, el comando mostrará el valor de la variable `HISTSIZE`:

```
sysadmin@localhost:~$ echo $HISTSIZE
1000
sysadmin@localhost:~$
```

La variable `HISTSIZE` define cuántos comandos anteriores se pueden almacenar en la lista del historial. Para mostrar el valor de la variable debes utilizar un carácter del signo de dólar `$` antes del nombre de la variable. Para modificar el valor de la variable, no se utiliza el carácter `$`:

```
sysadmin@localhost:~$ HISTSIZE=500
sysadmin@localhost:~$ echo $HISTSIZE
500
sysadmin@localhost:~$
```

Hay muchas variables del shell que están disponibles para el shell BASH, así como las variables que afectarán a los diferentes comandos de Linux. No todas las variables del shell están cubiertas por este capítulo, sin embargo, conforme vaya avanzando este curso hablaremos de más variables del shell.

#### 4.6 La Variable PATH

Una de las variables del shell BASH más importante que hay que entender es la variable `PATH`.

El término *path* (o «ruta» en español) se refiere a una lista que define en qué directorios el shell buscará los comandos. Si introduces un comando y recibes el error "command not found" (o «comando no encontrado» en español), es porque el shell BASH no pudo localizar un comando por ese nombre en cualquiera de los directorios en la ruta. El comando siguiente muestra la ruta del shell actual:

```
sysadmin@localhost:~$ echo $PATH
/home/sysadmin/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:
/usr/games
sysadmin@localhost:~$
```

Basado en la anterior salida, cuando intentas ejecutar un comando, el shell primero busca el comando en el directorio `/home/sysadmin/bin`. Si el comando se encuentra en ese directorio, entonces se ejecuta. Si no es encontrado, el shell buscará en el directorio `/usr/local/sbin`.

Si el comando no se encuentra en ningún directorio listado en la variable `PATH`, entonces recibirás un error, `command not found`:

```
sysadmin@localhost:~$ zed
-bash: zed: command not found
sysadmin@localhost:~$
```

Si en tu sistema tienes instalado un software personalizado, puede que necesites modificar la ruta PATH para que sea más fácil ejecutar estos comandos. Por ejemplo, el siguiente comando agregará el directorio /usr/bin/custom a la variable PATH:

```
sysadmin@localhost:~$ PATH=/usr/bin/custom:$PATH
sysadmin@localhost:~$ echo $PATH
/usr/bin/custom:/home/sysadmin/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
sysadmin@localhost:~$
```

#### 4.7 Comando export

Hay dos tipos de variables utilizadas en el shell BASH, la local y la de entorno. Las variables de entorno, como PATH y HOME, las utiliza el BASH al interpretar los comandos y realizar las tareas. Las variables locales son a menudo asociadas con las tareas del usuario y son minúsculas por convención. Para crear una variable local, simplemente introduce:

```
sysadmin@localhost:~$ variable1='Something'
```

Para ver el contenido de la variable, te puedes referir a ella iniciando con el signo de \$:

```
sysadmin@localhost:~$ echo $variable1
Something
```

Para ver las variables de entorno, utiliza el comando env (la búsqueda a través de la salida usando grep, tal como se muestra aquí, se tratará en los capítulos posteriores). En este caso, la búsqueda para variable1 en las variables de entorno resultará en una salida nula:

```
sysadmin@localhost:~$ env | grep variable1
sysadmin@localhost:~$
```

Después de exportar variable1 llegará a ser una variable de entorno. Observa que esta vez, se encuentra en la búsqueda a través de las variables de entorno:

```
sysadmin@localhost:~$ export variable1
sysadmin@localhost:~$ env | grep variable1
variable1=Something
```

El comando `export` también puede utilizarse para hacer una variable de entorno en el momento de su creación:

```
sysadmin@localhost:~$ export variable2='Else'
sysadmin@localhost:~$ env | grep variable2
variable2=Else
```

Para cambiar el valor de una variable de entorno, simplemente omite el `$` al hacer referencia a tal valor:

```
sysadmin@localhost:~$ variable1=$variable1 '$variable2'
sysadmin@localhost:~$ echo $variable1
Something Else
```

Las variables exportadas pueden eliminarse con el comando `unset`:

```
sysadmin@localhost:~$ unset variable2
```

#### 4.8 Comando which

Puede haber situaciones donde diferentes versiones del mismo comando se instalan en un sistema o donde los comandos son accesibles para algunos usuarios y a otros no. Si un comando no se comporta como se esperaba o si un comando no está accesible pero debería estarlo, puede ser beneficioso saber donde el shell encuentra tal comando o que versión está utilizando.

Sería tedioso tener que buscar manualmente en cada directorio que se muestra en la variable `PATH`. En su lugar, puedes utilizar el comando `which` (o «cuál» en español) para mostrar la ruta completa del comando en cuestión:

```
sysadmin@localhost:~$ which date
/bin/date
sysadmin@localhost:~$ which cal
/usr/bin/cal
sysadmin@localhost:~$
```

El comando `which` busca la ubicación de un comando buscando en la variable `PATH`.

#### 4.9 Comando type

El comando `type` puede utilizarse para determinar la información acerca de varios comandos. Algunos comandos se originan de un archivo específico:

```
sysadmin@localhost:~$ type which
which is hashed (/usr/bin/which)
```



Esta salida sería similar a la salida del comando `which` (tal como se explica en el apartado anterior, que muestra la ruta completa del comando):

```
sysadmin@localhost:~$ which which
/usr/bin/which
```

El comando `type` también puede identificar comandos integrados en el bash (u otro) shell:

```
sysadmin@localhost:~$ type echo
echo is a shell builtin
```

En este caso, la salida es significativamente diferente de la salida del comando `which`:

```
sysadmin@localhost:~$ which echo
/bin/echo
```

Usando la opción `-a`, el comando `type` también puede revelar la ruta de otro comando:

```
sysadmin@localhost:~$ type -a echo
echo is a shell builtin
echo is /bin/echo
```

El comando `type` también puede identificar a los alias para otros comandos:

```
sysadmin@localhost:~$ type ll
ll is aliased to `ls -aF`
sysadmin@localhost:~$ type ls
ls is aliased to `ls --color=auto`
```

La salida de estos comandos indican que `ll` es un alias para `ls -aF`, incluso `ls` es un alias para `ls --color=auto`. Una vez más, la salida es significativamente diferente del comando `which`:

```
sysadmin@localhost:~$ which ll
sysadmin@localhost:~$ which ls
/bin/ls
```

El comando `type` soporta otras opciones y puede buscar varios comandos al mismo tiempo. Para mostrar sólo una sola palabra que describe al `echo`, `ll`, y a los comandos `which`, utiliza la opción `-t`:

```
sysadmin@localhost:~$ type -t echo ll which
```

```
builtin
alias
file
```

#### 4.10 Los Alias

Un *alias* puede utilizarse para asignar comandos más largos a secuencias más cortas. Cuando el shell ve un alias ejecutado, sustituye la secuencia más larga antes de proceder a interpretar los comandos.

Por ejemplo, el comando `ls -l` comúnmente tiene un alias `l` o `ll`. Ya que estos comandos más pequeños son más fáciles de introducir, también es más rápido ejecutar la línea de comandos `ls -l`.

Puedes determinar qué alias se definen en el shell con el comando `alias`:

```
sysadmin@localhost:~$ alias

alias alert='notify-send --urgency=low -i "${[ $? = 0 ] && echo terminal || echo error}" "$(history|tail -n1|sed -e '\''s/^\s*[0-9]\+\s*//;s/[\;:&|]\s*alert$/\s*\'\''")'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -aLF'
alias ls='ls --color=auto'
```

Los alias que ves en los ejemplos anteriores fueron creados por los archivos de inicialización. Estos archivos están diseñados para hacer automático el proceso de creación de los alias. Hablaremos sobre ellos con más detalle en un capítulo posterior.

Los nuevos alias se pueden crear introduciendo `alias name=command` (o «alias nombre=comando» en español), donde nombre es el nombre que quieres dar a el alias y comando es el comando que quieres que se ejecute cuando se ejecuta el alias.

Por ejemplo, puedes crear un alias de tal manera que `lh` muestre una lista larga de archivos, ordenados por tamaño con un tamaño "human friendly" (o «amigable para el usuario» en español) con el comando `alias lh='ls -Shl'`. Introduciendo `lh` debe ahora dar lugar a la misma salida que introduciendo el comando `ls -Shl`:

```
sysadmin@localhost:~$ alias lh='ls -Shl'
sysadmin@localhost:~$ lh /etc/ppp

total 0
drwxr-xr-x 1 root root 10 Jan 29  2015 ip-down.d
```

```
drwxr-xr-x 1 root root 10 Jan 29 2015 ip-up.d
```

Los alias **creados de esta manera sólo persistirán mientras el shell esté abierto**. Una vez que el shell es cerrado, los nuevos alias que hayas creado, se perderán. Además, cada shell posee sus propios alias, así que si creas un alias en un shell y luego lo abres en otro shell, no verás el alias en el nuevo shell.

#### 4.11 Globbing

Los caracteres de globbing **se denominan a menudo como** "comodines". Estos son símbolos que tienen un significado especial para el shell.

A diferencia de los comandos que ejecutará el shell, u opciones y argumentos que el shell pasará a los comandos, los comodines son interpretados por el mismo shell antes de que intente ejecutar cualquier comando. **Esto significa que los comodines pueden utilizarse con cualquier comando.**

Los comodines son poderosos porque permiten especificar patrones que coinciden con los nombres de archivo en un directorio, así que en lugar de manipular un solo archivo a la vez, puedes fácilmente ejecutar comandos que afectarán a muchos archivos. Por ejemplo, utilizando comodines es posible manipular todos los archivos con una cierta extensión o con una longitud de nombre de archivo determinado.

Ten en cuenta que estos comodines pueden utilizarse con cualquier comando, ya que es el shell, no el comando que se expande con los comodines a la coincidencia de nombres de archivo. Los ejemplos proporcionados en este capítulo utilizan el comando `echo` para demostración.

##### 4.11.1 Asterisco (\*)

El asterisco se utiliza para representar cero o más de cualquier carácter en un nombre de archivo. Por ejemplo, supongamos que quieres visualizar todos los archivos en el directorio `/etc` que empiecen con la letra `t`:

```
sysadmin@localhost:~$ echo /etc/t*
/etc/terminfo /etc/timezone
sysadmin@localhost:~$
```

El patrón `t*` significa "cualquier archivo que comienza con el carácter `t` y tiene cero o más de cualquier carácter después de la letra `t`".

Puedes usar el asterisco en cualquier lugar dentro del patrón del nombre de archivo. El siguiente ejemplo coincidirá con cualquier nombre de archivo en el directorio `/etc` que termina con `.d`:

```
sysadmin@localhost:~$ echo /etc/*.d
/etc/apparmor.d /etc/bash_completion.d /etc/cron.d /etc/depmod.d /etc/fstab.d /etc/init.d /etc/insserv.conf.d /etc/ld.so.conf.d
/etc/logrotate.d /etc/modprobe.d /etc/pam.d /etc/profile.d /etc/rc0.d /etc/rc1.d /etc/rc2.d /etc/rc3.d /etc/rc4.d /etc/rc5.d /etc/rc6.d
/etc/rcS.d /etc/rsyslog.d /etc/sudoers.d /etc/sysctl.d /etc/update-motd.d
```

En el ejemplo siguiente se mostrarán todos los archivos en el directorio `/etc` que comienzan con la letra `r` y terminan con `.conf`:

```
sysadmin@localhost:~$ echo /etc/r*.conf
/etc/resolv.conf /etc/rsyslog.conf
```

##### 4.11.2 Signo de Interrogación (?)

El signo de interrogación representa cualquier carácter único. Cada carácter de signo de interrogación coincide con exactamente un carácter, nada más y nada menos. Supongamos que quieres visualizar todos los archivos en el directorio `/etc` que comienzan con la letra `t` y que tienen exactamente 7 caracteres después del carácter de `t`:

```
sysadmin@localhost:~$ echo /etc/t???????
/etc/terminfo /etc/timezone
sysadmin@localhost:~$
```

Los comodines pueden utilizarse juntos para encontrar patrones más complejos. El comando `echo /etc/*????????????????` imprimirá sólo los archivos del directorio `/etc` con veinte o más caracteres en el nombre del archivo:

```
sysadmin@localhost:~$ echo /etc/*????????????????
/etc/bindresvport.blacklist /etc/ca-certificates.conf
sysadmin@localhost:~$
```

El asterisco y el signo de interrogación también podrían usarse juntos para buscar archivos con extensiones de tres letras ejecutando el comando `echo /etc/*.???`:

```
sysadmin@localhost:~$ echo /etc/*.???
/etc/blkid.tab /etc/issue.net
sysadmin@localhost:~$
```

#### 4.11.3 Corchetes [ ]

Los corchetes se utilizan para coincidir con un carácter único representando un intervalo de caracteres que pueden coincidir con los caracteres. Por ejemplo, `echo /etc/[gu]*` imprimirá cualquier archivo que comienza con el carácter `g` o `u` y contiene cero o más caracteres adicionales:

```
sysadmin@localhost:~$ echo /etc/[gu]*
/etc/gai.conf /etc/groff /etc/group /etc/group- /etc/gshadow /etc/gshadow- /etc/ucf.conf /etc/udev /etc/ufw /etc/update-motd.d /etc/updatedb.conf
sysadmin@localhost:~$
```

Los corchetes también pueden ser utilizados para representar un intervalo de caracteres. Por ejemplo, el comando `echo /etc/[a-d]*` mostrará todos los archivos que comiencen con cualquier letra entre `e` incluyendo `a` y `d`:

```
sysadmin@localhost:~$ echo /etc/[a-d]*
/etc/adduser.conf /etc/adjtime /etc/alternatives /etc/apparmor.d
```

```
/etc/apt /etc/bash.bashrc /etc/bash_completion.d /etc/bind /etc/bindresvport.blacklist /etc/blkid.conf /etc/blkid.tab /etc/ca-certificates /etc/ca-certificates.conf /etc/calendar /etc/cron.d /etc/cron.daily /etc/cron.hourly /etc/cron.monthly /etc/cron.weekly /etc/crontab /etc/dbus-1 /etc/debconf.conf /etc/debian_version /etc/default
/etc/deluser.conf /etc/depmod.d /etc/dpkg
sysadmin@localhost:~$
```

El comando `echo /etc/*[0-9]*` mostrará todos los archivos que contienen al menos un número:

```
sysadmin@localhost:~$ echo /etc/*[0-9]*
/etc/dbus-1 /etc/iproute2 /etc/mke2fs.conf /etc/python2.7 /etc/rc0.d
/etc/rc1.d /etc/rc2.d /etc/rc3.d /etc/rc4.d /etc/rc5.d /etc/rc6.d
sysadmin@localhost:~$
```

El intervalo se basa en el cuadro de texto de ASCII. Esta tabla define una lista de caracteres disponiéndolos en un orden estándar específico. Si proporcionas un orden inválido, no se registrará ninguna coincidencia:

```
sysadmin@localhost:~$ echo /etc/*[9-0]*
/etc/*[9-0]*
sysadmin@localhost:~$
```

#### 4.11.4 Signo de Exclamación (!)

El signo de exclamación se utiliza en conjunto con los corchetes para negar un intervalo. Por ejemplo, el comando `echo [!DP]*` mostrará cualquier archivo que no comienza con D o P.

#### 4.12 Las Comillas

Hay tres tipos de comillas que tienen significado especial para el shell Bash: comillas dobles `"`, comillas simples `'` y comilla invertida ```. Cada conjunto de comillas indica al shell que debe tratar el texto dentro de las comillas de una manera distinta a la normal.

##### 4.12.1 Comillas Dobles

Las comillas dobles detendrán al shell de la *interpretación* de algunos metacaracteres, incluyendo los comodines. Dentro de las comillas dobles, el asterisco es sólo un asterisco, un signo de interrogación es sólo un signo de interrogación y así sucesivamente. Esto significa que cuando se utiliza el segundo comando `echo` más abajo, el shell BASH no convierte el patrón de globbing en nombres de archivos que coinciden con el patrón:

```
sysadmin@localhost:~$ echo /etc/[DP]*
/etc/DIR_COLORS /etc/DIR_COLORS.256color /etc/DIR_COLORS.lightbgcolor /etc/PackageKit
```

```
sysadmin@localhost:~$ echo "/etc/[DP] *"
/etc/[DP] *
sysadmin@localhost:~$
```

Esto es útil cuando quieres mostrar algo en la pantalla, lo que suele ser un carácter especial para el shell:

```
sysadmin@localhost:~$ echo "The glob characters are *, ? and [ ]"
The glob characters are *, ? and [ ]
sysadmin@localhost:~$
```

Las comillas dobles todavía permiten la *sustitución de comando* (se tratará más adelante en este capítulo), sustitución de variable y permiten algunos metacaracteres de shell sobre los que aún no hemos hablado. Por ejemplo, en la siguiente demostración, notarás que el valor de la variable `PATH` es desplegada:

```
sysadmin@localhost:~$ echo "The path is $PATH"
The path is /usr/bin/custom:/home/sysadmin/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
sysadmin@localhost:~$
```

#### 4.12.2 Comillas Simples

Las comillas simples evitan que el shell interprete algunos caracteres especiales. Esto incluye comodines, variables, sustitución de comando y otro metacarácter que aún no hemos visto.

Por ejemplo, si quieres que el carácter `$` simplemente signifique un `$`, en lugar de actuar como un indicador del shell para buscar el valor de una variable, puedes ejecutar el segundo comando que se muestra a continuación:

```
sysadmin@localhost:~$ echo The car costs $100
The car costs 00
sysadmin@localhost:~$ echo 'The car costs $100'
The car costs $100
sysadmin@localhost:~$
```

#### 4.12.3 Barra Diagonal Inversa (\)

Puedes utilizar una técnica alternativa para citar un carácter con comillas simples. Por ejemplo, supón que quieres imprimir lo siguiente: "The services costs \$100 and the path is \$PATH". Si pones esto entre las comillas dobles, `$1` y `$PATH` se consideran variables. Si pones esto entre las comillas simples, `$1` y `$PATH` no son variables. Pero ¿qué pasa si quieres tener `$PATH` tratado como una variable y no a `$1`?



Si colocas una barra diagonal invertida \ antes del otro carácter, tratará al otro carácter como un carácter de "comillas simples". El tercer comando más abajo muestra cómo utilizar el carácter \, mientras que los otros dos muestran cómo las variables serían tratadas si las pones entre las comillas dobles y simples:

```
sysadmin@localhost:~$ echo "The service costs $100 and the path is $PATH"
The service costs 00 and the path is /usr/bin/custom:/home/sysadmin/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games

sysadmin@localhost:~$ echo 'The service costs $100 and the path is $PATH'
The service costs $100 and the path is $PATH

sysadmin@localhost:~$ echo The service costs \$100 and the path is $PATH
The service costs $100 and the path is /usr/bin/custom:/home/sysadmin/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games

sysadmin@localhost:~$
```

#### 4.12.4 Comilla Invertida

Las comillas invertidas se utilizan para especificar un comando dentro de un comando, un proceso de sustitución del comando. Esto permite un uso muy potente y sofisticado de los comandos.

Aunque puede sonar confuso, un ejemplo debe hacer las cosas más claras. Para empezar, fíjate en la salida del comando `date`:

```
sysadmin@localhost:~$ date
Mon Nov  2 03:35:50 UTC 2015
```

Ahora fíjate en la salida de la línea de comandos `echo Today is date` (o «eco La fecha de hoy es» en español):

```
sysadmin@localhost:~$ echo Today is date
Today is date

sysadmin@localhost:~$
```

En el comando anterior la palabra `date` (o «fecha» en español) es tratada como texto normal y el shell simplemente pasa `date` al comando `echo`. Pero, probablemente quieras ejecutar el comando `date` y tener la salida de ese comando enviado al comando `echo`. Para lograr esto, deberás ejecutar la línea de comandos `echo Today is `date``:

```
sysadmin@localhost:~$ echo Today is `date`
Today is Mon Nov 2 03:40:04 UTC 2015

sysadmin@localhost:~$
```

#### 4.13 Instrucciones de Control

Las instrucciones de control te permiten utilizar varios comandos a la vez o ejecutar comandos adicionales, dependiendo del éxito de un comando anterior. Normalmente estas instrucciones de control se utilizan en scripts o secuencias de comandos, pero también pueden ser utilizadas en la línea de comandos.

#### 4.13.1 Punto y Coma

El punto y coma puede utilizarse para ejecutar varios comandos, uno tras otro. Cada comando se ejecuta de forma independiente y consecutiva; no importa el resultado del primer comando, el segundo comando se ejecutará una vez que el primero haya terminado, luego el tercero y así sucesivamente.

Por ejemplo, si quieres imprimir los meses de enero, febrero y marzo de 2015, puedes ejecutar `cal 1 2015; cal 2 2015; cal 3 2015` en la línea de comandos:

```
sysadmin@localhost:~$ cal 1 2015; cal 2 2015; cal 3 2015
```

```
January 2015
```

```
Su Mo Tu We Th Fr Sa
```

```
1 2 3
```

```
4 5 6 7 8 9 10
```

```
11 12 13 14 15 16 17
```

```
18 19 20 21 22 23 24
```

```
25 26 27 28 29 30 31
```

```
February 2015
```

```
Su Mo Tu We Th Fr Sa
```

```
1 2 3 4 5 6 7
```

```
8 9 10 11 12 13 14
```

```
15 16 17 18 19 20 21
```

```
22 23 24 25 26 27 28
```

```
March 2015
```

```
Su Mo Tu We Th Fr Sa
```

```
1 2 3 4 5 6 7
```

```
8 9 10 11 12 13 14
```

```
15 16 17 18 19 20 21
```

```
22 23 24 25 26 27 28
```

### 4.13.2 Ampersand Doble (&&)

El símbolo de ampersand doble && actúa como un operador "y" lógico. Si el primer comando tiene éxito, entonces el segundo comando (a la derecha de la &&) también se ejecutará. Si el primer comando falla, entonces el segundo comando no se ejecutará.

Para entender mejor como funciona esto, consideremos primero el concepto de fracaso y éxito para los comandos. Los comandos tienen éxito cuando algo funciona bien y fallan cuando algo sale mal. Por ejemplo, considera la línea de comandos `ls /etc/xml`. El comando tendrá éxito si el directorio `/etc/xml` es accesible y fallará cuando no es accesible.

Por ejemplo, el primer comando tendrá éxito porque el directorio `/etc/xml` existe y es accesible mientras que el segundo comando fallará porque no hay un directorio `/junk`:

```
sysadmin@localhost:~$ ls /etc/xml
catalog  catalog.old  xml-core.xml  xml-core.xml.old
sysadmin@localhost:~$ ls /etc/junk
ls: cannot access /etc/junk: No such file or directory
sysadmin@localhost:~$
```

La manera en que usarías el éxito o fracaso del comando `ls` junto con && sería ejecutando una línea de comandos como la siguiente:

```
sysadmin@localhost:~$ ls /etc/xml && echo success
catalog  catalog.old  xml-core.xml  xml-core.xml.old
success
sysadmin@localhost:~$ ls /etc/junk && echo success
ls: cannot access /etc/junk: No such file or directory
sysadmin@localhost:~$
```

En el primer ejemplo arriba, el comando `echo` fue ejecutado, porque tuvo éxito el comando `ls`. En el segundo ejemplo, el comando `echo` no fue ejecutado debido a que el comando `ls` falló.

### 4.13.3 Línea Vertical Doble

La línea vertical doble `||` es un operador lógico "o". Funciona de manera similar a &&; dependiendo del resultado del primer comando, el segundo comando se ejecutará o será omitido.

Con la línea vertical doble, si el primer comando se ejecuta con éxito, el segundo comando es omitido. Si el primer comando falla, entonces se ejecutará el segundo comando. En otras palabras, esencialmente estás diciendo al shell, "O bien ejecuta este primer comando o bien el segundo".

En el ejemplo siguiente, el comando `echo` se ejecutará sólo si falla el comando `ls`:

```

sysadmin@localhost:~$ ls /etc/xml || echo failed
catalog  catalog.old  xml-core.xml  xml-core.xml.old
sysadmin@localhost:~$ ls /etc/junk || echo failed
ls: cannot access /etc/junk: No such file or directory
failed
sysadmin@localhost:~$

```

## NDG Linux Essentials: Capítulo 5: Encontrar ayuda

### 5.1 Introducción

Si le preguntas a los usuarios qué característica del Sistema Operativo Linux disfrutan más, muchos responderán «el poder proporcionado por el entorno de la línea de comandos». Esto es porque hay literalmente miles de comandos disponibles con muchas opciones, resultando en herramientas poderosas.

Sin embargo, con este poder viene la complejidad. La complejidad, a su vez, puede crear confusión. Como resultado, saber encontrar ayuda cuando trabajas en Linux es una habilidad esencial para cualquier usuario. Referirte a la ayuda te permite recordar cómo funciona un comando, además de ser un recurso de información al aprender nuevos comandos.

**93%** de los directores de recursos humano planea contratar a un profesional de Linux en los próximos seis meses. Y casi **90%** mencionó que es difícil encontrar profesionales experimentados en Linux. Esto significa muchas oportunidades de trabajo para aquellos con habilidades de Linux.

### 5.2 Las páginas man (manual)

Las páginas man se utilizan para describir las características de los comandos(programas)

Para ver una página man de un comando, ejecuta el `man comando`

El comando `man` utiliza un «localizador» para mostrar documentos. Este localizador es normalmente el comando `less`, pero en algunas distribuciones puede ser el comando `more`.

Si quieres ver los diferentes comandos de movimiento que están disponibles, puedes introducir la letra **h** mientras visualizas una página man. Esto mostrará una página de ayuda:

Si tu distribución usa el comando `less`, podría estar un poco abrumado por la gran cantidad de «comandos» que están disponibles. La tabla siguiente proporciona un resumen de los comandos más útiles:

Tabla 1 - resumen de los comandos

| Comando                  | Función                                     |
|--------------------------|---|
| <b>Return (o Intro)</b>  | Bajar una línea                             |
| <b>Space (o Espacio)</b> | Bajar una página                            |
| <b>/term(o /término</b>  | Buscar un término                           |
| <b>n</b>                 | Buscar el siguiente elemento de la búsqueda |
| <b>1G</b>                | Ir a Inicio                                 |
| <b>G</b>                 | Ir a la final                               |
| <b>h</b>                 | Mostrar ayuda                               |
| <b>q</b>                 | Cerrar página man                           |

### 5.2.3 Las secciones de las paginas MAN

Cada sección está diseñada para proporcionar información específica acerca de un comando.

| Nombre de la Sección         | Propósito  |
|------------------------------|--|
| NAME<br>(Nombre)             | Proporciona el nombre del comando y una breve descripción.<br><br><div data-bbox="553 302 1934 420"> <pre>NAME ls - list directory contents</pre> </div>   |
| SYNOPSIS<br>(Sinopsis)       | Proporciona ejemplos de cómo se ejecuta el comando. Información detallada a continuación.<br><br><div data-bbox="553 529 1934 647"> <pre>SYNOPSIS ls [OPTION]... [FILE]...</pre> </div>  |
| DESCRIPTION<br>(Descripción) | Proporciona una descripción más detallada del comando.<br><br><div data-bbox="553 756 1934 972"> <pre>DESCRIPTION  List information about the FILES (the current directory by default). Sort entries alphabetically if none of -cftuvSUX nor --sort is speci- fied.</pre> </div>   |
| OPTIONS<br>(Opciones)        | Muestra las opciones para el comando, así como una descripción de cómo se utilizan. A menudo esta información se encontrará en la sección DESCRIPTION (o «DESCRIPCIÓN» en español) y no en una sección separada de OPTIONS (o «OPCIONES» en español).<br><br><div data-bbox="553 1146 1934 1471"> <pre>-a, --all do not ignore entries starting with .  -A, --almost-all do not list implied . and ..</pre> </div> |



| Nombre de la Sección                      | Propósito   |
|---|---|
|   | <pre> --author     with -l, print the author of each file  -b, --escape     print C-style escapes for nongraphic characters  --block-size=SIZE     scale sizes by SIZE before printing them; e.g., '--block-size=M'     prints sizes in units of 1,048,576 bytes; see SIZE format below </pre>    |
| FILES<br>(Archivos)                       | Muestra las opciones para el comando, así como una descripción de cómo se utilizan. Estos archivos pueden utilizarse para configurar las características más avanzadas del comando. A menudo esta información se encontrará en la sección de DESCRIPTION y no en una sección separada de OPTIONS. |
| AUTHOR<br>(Autor)                         | El nombre de la persona que creó la página man y (a veces) la manera de contactar a la persona.   |
|   | <pre> AUTHOR  Written by Richard M. Stallman and David MacKenzie. </pre>  |
| REPORTING<br>BUGS<br>(Reportando Errores) | Proporciona información sobre cómo reportar problemas con el comando.   |
|   | <pre> REPORTING BUGS  GNU coreutils online help: &lt;http://www.gnu.org/software/coreutils/&gt; Report ls translation bugs to &lt;http://translationproject.org/team/&gt; </pre>  |
| COPYRIGHT<br>(Derechos de Autor)          | Proporciona información básica de los derechos de autor.  |
|   | <pre> COPYRIGHT </pre>  |

## Nombre de la Sección

## Propósito

```
Copyright (C) 2017 Free Software Foundation, Inc.  License GPLv3+:  GNU
GPL version 3 or later <http://gnu.org/licenses/gpl.html>.

This is free software: you are free to change and redistribute it.

There is NO WARRANTY, to the extent permitted by law.
```

## SEE ALSO (Ver También)

Proporciona una idea de dónde puedes encontrar información adicional. También suele incluir otros comandos que están relacionados con este comando.

## SEE ALSO

```
Full documentation at: <http://www.gnu.org/software/coreutils/ls>
or available locally via: info '(coreutils) ls invocation'
```

### 5.2.4 La sección SYNOPSIS de la Página man.

La sección de SYNOPSIS (o «SINOPSIS» en español) de una página man puede ser difícil de entender, pero es muy importante porque proporciona un ejemplo conciso de cómo utilizar el comando. Por ejemplo, considera la SYNOPSIS de la página man para el comando `cal`:

#### SYNOPSIS

```
cal [-3h jy] [-A number] [-B number] [[[day] month] year]
```

Los corchetes `[ ]` se utilizan para indicar que esta característica no es necesaria para ejecutar el comando. Por ejemplo, `[-3h jy]` significa que puedes usar las opciones `-h`, `-j`, `-y`, `1` o `3`, pero ninguna de estas opciones son necesarias para el correcto funcionamiento del comando `cal`.

El segundo conjunto de corchetes en la SYNOPSIS del comando `cal` (`[[[day] month] year]`) muestra otra característica. Esto significa que puedes especificar un año por sí mismo, pero si se especifica un mes también se debe especificar un año. Además, si especificas un día entonces también necesitarás especificar un mes y un año.

Otro componente de SYNOPSIS que puede causar cierta confusión puede verse en SYNOPSIS del comando `date`:

#### SYNOPSIS

```
date [OPTION]... [+FORMAT]
date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]
```

En esta SYNOPSIS hay dos sintaxis para el comando `date`. El primero de ellos se utiliza para mostrar la fecha en el sistema mientras que el segundo se utiliza para fijar la fecha.

Las elipses siguientes a `[OPTION]`, . . . , indican que uno o más ítems antes de la opción pueden usarse.

Además, la notación `[-u|--utc|--universal]` significa que puedes utilizar la opción `-u`, la opción `--utc` o la opción `--universal`. Normalmente esto significa que las tres opciones hacen lo mismo, pero a veces este formato (uso del carácter `|`) sirve para indicar que no se utilicen las opciones combinadas, tal como un operador lógico «o».

### 5.2.5 Buscando en la página MAN

Para buscar un término en la página man, pulsa `/`

Si se encuentra una coincidencia y quieres pasar al siguiente término, pulsa `n`. Para volver al término anterior pulsa `N`

### 5.2.6 Las páginas MAN categorizadas por secciones.

Existen varios tipos de comandos (`comandos de usuario, comandos del sistema y comandos de administración`), así como `otras funciones` que requieren documentación, como las `librerías y los componentes del Kernel`.

**Para considerar:**

Por defecto, hay nueve secciones de las páginas man:

- Programas ejecutables o comandos del shell
- Llamadas del sistema (funciones proporcionados por el kernel)
- Llamadas de la librería (funciones dentro de las librerías de los programas)
- Archivos especiales (generalmente se encuentran en `/dev`)
- Formatos de archivo y convenciones, por ejemplo `/etc/passwd`
- Juegos
- Otros (incluyendo paquetes macro y convenciones), por ejemplo, `man(7)`, `groff(7)`
- Comandos de administración de sistema (generalmente sólo para el root)
- Rutinas del kernel [No estándar]

El comando `man <comando>` va analizando cada una de las secciones de las paginas MAN, para encontrar similitudes, desde la primera hasta la ultima en ese orden, si no se encuentra coincidencias el programa te mostrara un mensaje de error.

#### 5.2.6.1 Determinar la sección

Para determinar a qué sección pertenece una página man específica tienes que ver el valor numérico de la primera línea de la salida de la página man. Ej: `man cal`

```
CAL(1)                                BSD General Commands Manual          CAL(1)
```

#### 5.2.6.1 Especificar la sección

En algunos casos, necesitarás especificar la sección para visualizar la página man correcta. Esto es necesario porque `a veces habrá páginas man con el mismo nombre en diferentes secciones.`

Por ejemplo, hay un comando llamado `passwd` que permite cambiar tu contraseña. También hay un archivo llamado `passwd` que almacena la información de la cuenta. Ambos, el comando y el archivo tienen una página man.

El comando `passwd` es un comando de "user" (o «usuario» en español), por lo que el comando `man passwd` mostrará la página man para el comando `passwd` por defecto:

```
PASSWD (1)                                User Commands                                PASSWD (1)
```

Para especificar una sección diferente, proporciona el número de la sección como el primer argumento del comando `man`. Por ejemplo, el comando `man 5 passwd` buscará la página man de `passwd` sólo en la sección 5:

```
PASSWD (5)                                File Formats and Conversions                                PASSWD (5)
```

### 5.2.6.3 Buscar las Secciones

A veces no es claro en qué sección se almacena una página man. En estos casos, puedes buscar una página man por nombre.

La opción `-f` para el comando `man` mostrará páginas que coinciden, o parcialmente coinciden, con un nombre específico y provee una breve descripción de cada página man:

```
sysadmin@localhost:~$ man -f passwd
passwd (5)          - the password file
passwd (1)          - change user password
passwd (1ssl)       - compute password hashes
sysadmin@localhost:~$
```

Ten en cuenta que en la mayoría de las distribuciones de Linux, el comando `whatis` hace lo mismo que el comando `man -f`. En esas distribuciones, ambos comandos producen la misma salida.

### 5.2.7 Buscar Páginas man por una Palabra Clave

Desafortunadamente, no siempre te acordarás del nombre exacto de la página man que quieres ver. En estos casos puedes buscar páginas man que coincidan con una palabra clave mediante el uso de la opción `-k` del comando `man`.

Por ejemplo, ¿qué pasa si quieres ver una página que muestra cómo cambiar la contraseña, pero no recuerdas el nombre exacto? Puedes ejecutar el comando `man -k password`:

```
sysadmin@localhost:~$ man -k passwd
chgpaswd (8)        - update group passwords in batch mode
chpasswd (8)        - update passwords in batch mode
fgetpwent_r (3)     - get passwd file entry reentrantly
getpwent_r (3)     - get passwd file entry reentrantly
gpaswd (1)          - administer /etc/group and /etc/gshadow
pam_localuser (8)   - require users to be listed in /etc/passwd
```

```
passwd (1)      - change user password
passwd (1ssl)   - compute password hashes
passwd (5)      - the password file
passwd2des (3)  - RFS password encryption
update-passwd (8) - safely update /etc/passwd, /etc/shadow and /etc/group

sysadmin@localhost:~$
```

Al utilizar esta opción puedes recibir una gran cantidad de salidas. El comando anterior, por ejemplo, dió salida a 60 resultados.

Recuerda que hay miles de páginas man, así que cuando buscas por una palabra clave, sé tan específico como sea posible. Usando una palabra genérica, como "the" (o «el/la» en español), podría resultar en cientos o incluso miles de resultados.

Ten en cuenta que en la mayoría de las distribuciones de Linux, el comando `apropos` hace lo mismo que el comando `man -k`. En esas distribuciones, ambas producen la misma salida.

### 5.3 Comando info

Las páginas man son unas fuentes extensas de información, pero suelen tener algunas desventajas. Un ejemplo de una desventaja es que cada página man es un documento independiente y no está relacionado con ninguna otra página man. Aunque algunas páginas man tienen una sección `SEE ALSO` (o «Ver También» en español) que puede hacer referencia a otras páginas man, en realidad tienden a ser relacionadas con las fuentes de documentación.

El comando `info` también proporciona documentación sobre funciones y comandos del sistema operativo. El objetivo de este comando es ligeramente diferente de las páginas man: proporcionar un recurso de documentación que proporciona una estructura lógica, facilitando la lectura de la documentación.

En los documentos info la información se desglosa en categorías que funcionan de una manera parecida que una tabla de contenido en un libro. Se proporcionan hipervínculos hacia páginas con la información sobre los temas individuales para un comando específico o función. De hecho, toda la documentación se combina en un solo "book" (o «libro» en español) en el que puedes ir a un nivel superior de la documentación y ver la tabla de contenido que representa toda la documentación disponible.

Otra ventaja del comando info sobre las páginas man es que el estilo de escritura de los documentos info es típicamente más propicio para aprender un tema. Considera que las páginas man son un recurso de referencias y los documentos info sirven más como una guía de aprendizaje.

#### 5.3.1 Visualizar la Documentación Info para un Comando

Para visualizar la documentación `info` de un comando, ejecuta el comando `info command` (reemplaza `command` con el nombre del comando sobre cuál buscas la información). Por ejemplo, la siguiente pantalla muestra la salida del comando `info ls`:

```
File: coreutils.info, Node: ls invocation, Next: dir invocation, Up: Directo\ry listing

10.1 `ls': List directory contents
=====
```

The ``ls'` program lists information about files (of any type, including directories). Options and file arguments can be intermixed arbitrarily, as usual.

For non-option command-line arguments that are directories, by default ``ls'` lists the contents of directories, not recursively, and omitting files with names beginning with ``.``. For other non-option arguments, by default ``ls'` lists just the file name. If no non-option argument is specified, ``ls'` operates on the current directory, acting as if it had been invoked with a single argument of ``.``.

By default, the output is sorted alphabetically, according to the locale settings in effect. (1) If standard output is a terminal, the output is in columns (sorted vertically) and control characters are output as question marks; otherwise, the output is listed one per line and control characters are output as-is.

```
--zz-Info: (coreutils.info.gz)ls invocation, 58 lines --Top-----
Welcome to Info version 5.2. Type h for help, m for menu item.
```

Observa que la primera línea proporciona información que te indica dónde dentro de la documentación info estás ubicado. Esta documentación se divide en **nodes** (o «nodos» en español) y en el ejemplo anterior estás actualmente en el **nodo `ls invocation`**. Si pasaras al siguiente nodo (tal como ir al siguiente capítulo en un libro), estarías en el **nodo `dir invocation`**. Si te pasaras un nivel hacia arriba estarías en el nodo **Directory listing**.

### 5.3.2 Cambiando de Posición mientras se Visualiza un Documento **info**

Igual que el comando **man**, puedes obtener un listado de comandos de movimiento escribiendo la letra **h** al leer la documentación info:

```
Basic Info command keys

l          Close this help window.
q          Quit Info altogether.
H          Invoke the Info tutorial.
Up         Move up one line.
Down       Move down one line.
DEL        Scroll backward one screenful.
SPC        Scroll forward one screenful.
```



```

Home      Go to the beginning of this node.
End        Go to the end of this node.
TAB        Skip to the next hypertext link.
RET        Follow the hypertext link under the cursor.
l          Go back to the last node seen in this window.
[          Go to the previous node in the document.
]          Go to the next node in the document.
p          Go to the previous node on this level.
n          Go to the next node on this level.
u          Go up one level.
-----Info: *Info Help*, 466 lines --Top-----

```

Ten en cuenta que si quieres **cerrar** la pantalla de ayuda, debes introducir la letra **I**. Esto te regresa a tu documento y te permite a continuar leyendo. Para **salir completamente**, introduce la letra **q**.

La tabla siguiente proporciona un resumen de los comandos útiles:

| Comando               | Función                |
|-----------------------|------------------------|
| <b>Flecha abajo ↓</b> | Bajar una línea        |
| <b>Espacio</b>        | Bajar una página       |
| <b>s</b>              | Buscar un término      |
| <b>[</b>              | Ir al nodo anterior    |
| <b>]</b>              | Vaya al siguiente nodo |

| Comando          | Función                          |
|------------------|----------------------------------|
| <b>u</b>         | Subir un nivel                   |
| <b>TABULADOR</b> | Saltar al siguiente hipervínculo |
| <b>INICIO</b>    | Ir a inicio                      |
| <b>FIN</b>       | Ir al final                      |
| <b>h</b>         | Mostrar ayuda                    |
| <b>L</b>         | Cerrar la página de ayuda        |
| <b>q</b>         | Cerrar el comando info           |

Si te desplazas en el documento, verás el menú para el comando `ls`:

```
* Menu:

* Which files are listed::
* What information is listed::
* Sorting the output::
* Details about version sort::
* General output formatting::
* Formatting file timestamps::
* Formatting the file names::
```

----- Footnotes -----

(1) If you use a non-POSIX locale (e.g., by setting `LC\_ALL' to `en\_US'), then `ls' may produce output that is sorted differently than you're accustomed to. In that case, set the `LC\_ALL' environment variable to `C'.

--zz-Info: (coreutils.info.gz)ls invocation, 58 lines --Top-----

Los **elementos bajo el menú son hipervínculos** que pueden llevarte a los nodos que describen más del comando `ls`. Por ejemplo, si colocas tu cursor en la línea **«\*Sorting the output:»** (o «clasificando la salida» en español) y presionas la tecla **Entrar**, pasarás al **nodo** que describe la clasificación de la salida del comando `ls`:

File: coreutils.info, Node: Sorting the output, Next: Details about version s\ort, Prev: What information is listed, Up: ls i  
nvocation

### 10.1.3 Sorting the output

-----

These options change the order in which `ls' sorts the information it outputs. By default, sorting is done by character code (e.g., ASCII order).

`-c'

`--time=ctime'

`--time=status'

If the long listing format (e.g., `-l', `-o') is being used, print the status change time (the `ctime' in the inode) instead of the modification time. When explicitly sorting by time (`--sort=time' or `-t') or when not using a long listing format, sort according to the status change time.

`-f'

Primarily, like `-U'--do not sort; list the files in whatever order they are stored in the directory. But also enable `-a' (lis

```
--zz-Info: (coreutils.info.gz)Sorting the output, 68 lines --Top-----
```

Ten en cuenta que **entrando al nodo de clasificación**, prácticamente entras a un **subnodo del nodo** del que originalmente partiste. **Para regresar a tu nodo anterior, puedes utilizar la tecla u**. Mientras que **u** te llevará un nivel arriba hasta el inicio del nodo, también puedes utilizar la tecla **l** para volver exactamente a la ubicación anterior en la que te encontrabas antes de entrar al nodo de clasificación.

### 5.3.3 Explorar la Documentación info

En lugar de utilizar la documentación info para buscar la información sobre un comando específico o función, **puedes considerar la posibilidad de explorar las capacidades de Linux mediante la lectura a través de la documentación info**. Si ejecutas el comando **info** sin ningún argumento, pasarás a un nivel superior de la **documentación**. Desde allí puedes explorar muchas características:

```
File: dir,      Node: Top      This is the top of the INFO tree

This (the Directory node) gives a menu of major topics.

Typing "q" exits, "?" lists all Info commands, "d" returns here,
"h" gives a primer for first-timers,
"mEmacs" visits the Emacs manual, etc.

In Emacs, you can click mouse button 2 on a menu item or cross reference to select it.

* Menu:

Basics

* Common options: (coreutils)Common options.
* Coreutils: (coreutils).      Core GNU (file, text, shell) utilities.
* Date input formats: (coreutils)Date input formats.
* File permissions: (coreutils)File permissions.
                                Access modes.
* Finding files: (find).      Operating on files matching certain criteria.

C++ libraries

* autosprintf: (autosprintf).  Support for printf format strings in C+

-----Info: (dir)Top, 211 lines --Top-----

Welcome to Info version 5.2. Type h for help, m for menu item.
```

## 5.4 Otras Fuentes de Ayuda

En muchos casos verás que las páginas man o documentación info te proporcionarán las respuestas que necesitas. Sin embargo, en algunos casos, puede que necesites buscar en otras ubicaciones.

#### 5.4.1 Utilizar la opción `--help`

Muchos comandos te proporcionan información básica, muy similar a la sección SYNOPSIS que aparece en las páginas man, al aplicar la opción `--help` (o «ayuda» en español) al comando. Esto es útil para aprender el uso básico de un comando:

```
sysadmin@localhost:~$ ps --help
***** simple selection *****      ***** selection by list *****
-A all processes                      -C by command name
-N negate selection                  -G by real group ID (supports names)
-a all w/ tty except session leaders -U by real user ID (supports names)
-d all except session leaders        -g by session OR by effective group name
-e all processes                     -p by process ID
T all processes on this terminal      -s processes in the sessions given
a all w/ tty, including other users   -t by tty
g OBSOLETE -- DO NOT USE              -u by effective user ID (supports names)
r only running processes              U processes for specified users
x processes w/o controlling ttys      t by tty
***** output format *****          ***** long options *****
-o,o user-defined  -f full              --Group --User --pid --cols --ppid
-j,j job control   s signal              --group --user --sid --rows --info
-O,O preloaded -o v virtual memory      --cumulative --format --deselect
-l,l long          u user-oriented       --sort --tty --forest --version
-F extra full      X registers           --heading --no-headi

***** misc options *****
-V,V show version      L list format codes  f ASCII art forest
-m,m,-L,-T,H threads  S children in sum    -y change -l format
-M,Z security data    c true command name  -c scheduling class
-w,w wide output      n numeric WCHAN,UID  -H process hierarchy

sysadmin@localhost:~$
```

### 5.4.2 Documentación Adicional del Sistema

En la mayoría de los sistemas, existe un directorio donde se encuentra la documentación adicional. A menudo se trata de una ubicación donde los proveedores que crean software adicional (de terceros) almacenan sus archivos de documentación.

Por lo general, **se trata de una ubicación donde los administradores del sistema irán a aprender cómo configurar servicios de software más complejos.** Sin embargo, los usuarios regulares a veces también encuentran esta documentación útil.

Estos **archivos de documentación** se suelen llamar archivos "**readme**" (o «leeme» en español), ya que los archivos tienen nombres como `README` o `readme.txt`. La ubicación de estos archivos puede variar según la distribución que estés utilizando. **Ubicaciones típicas incluyen `/usr/share/doc` y `/usr/doc`.**

### 5.5 Búsqueda de los Comandos y la Documentación

Recuerda que el comando `whatis` (o `man -f`) te dirá en qué sección se almacena la página man. **Si utilizas este comando con suficiente frecuencia, probablemente te topes con una salida inusual, como la siguiente:**

```
sysadmin@localhost:~$ whatis ls
ls (1)          - list directory contents
ls (lp)         - list directory contents
sysadmin@localhost:~$
```

Según esta salida, hay dos comandos que listan el contenido del directorio. La respuesta simple a la pregunta por qué hay dos comandos `ls` es que UNIX tuvo dos variantes principales, lo que dio lugar a que algunos comandos fueran desarrollados «en paralelo». Por lo tanto, algunos comandos se comportan diferentemente en diversas variantes de UNIX. Muchas distribuciones modernas de Linux incluyen comandos de ambas variantes de UNIX.

Esto, sin embargo, conlleva un pequeño problema: Cuando corres el comando `ls`, ¿Cuál de los comandos se ejecuta? Las próximas secciones se centrarán en responder esta pregunta, así como proporcionarte las herramientas para encontrar donde residen estos archivos en el sistema.

#### 5.5.1 ¿Dónde están ubicados estos comandos?

Para **buscar la ubicación de un comando** o de las páginas man para un comando, utiliza el comando `whereis` (o «dónde está» en español). Este comando busca los comandos, archivos de código fuente y las páginas man en las ubicaciones específicas donde estos archivos se almacenan normalmente:

```
sysadmin@localhost:~$ whereis ls
ls: /bin/ls /usr/share/man/man1/ls.1.gz
sysadmin@localhost:~$
```

Las páginas man **se suelen distinguir fácilmente** entre los comandos ya que **normalmente están comprimidos con un comando** llamado `gzip`, **dando por resultado un nombre de archivo que termina** en `.gz`.

Lo interesante es que verás que hay dos páginas man, pero sólo un comando (`/bin/ls`). Esto es porque el comando `ls` **puede utilizarse con las opciones/funciones** que se describen por *cualquiera* de las páginas man. Así que, cuando estás aprendiendo lo que puedes hacer con el comando `ls`, puedes explorar ambas páginas man. Afortunadamente, esto más bien es una excepción ya que la mayoría de los comandos sólo tienen una página man.

#### 5.5.2 Encontrar Cualquier Archivo o Directorio

El comando `whereis` está diseñado para encontrar de manera específica las páginas man y los comandos. Si bien esto es útil, hay veces en las que quieras encontrar un archivo o directorio, no sólo archivos de comandos o páginas man.

Para encontrar cualquier **archivo o directorio**, puede utilizar el comando `locate` (o «localizar» en español). Este comando buscará en una base de datos de todos los archivos y directorios que estaban en el sistema cuando se creó la base de datos. Por lo general, el comando que genera tal base de datos se ejecuta por la noche.

```
sysadmin@localhost:~$ locate gshadow
/etc/gshadow
/etc/gshadow-
/usr/include/gshadow.h
/usr/share/man/cs/man5/gshadow.5.gz
/usr/share/man/da/man5/gshadow.5.gz
/usr/share/man/de/man5/gshadow.5.gz
/usr/share/man/fr/man5/gshadow.5.gz
/usr/share/man/it/man5/gshadow.5.gz
/usr/share/man/man5/gshadow.5.gz
/usr/share/man/ru/man5/gshadow.5.gz
/usr/share/man/sv/man5/gshadow.5.gz
/usr/share/man/zh_CN/man5/gshadow.5.gz
sysadmin@localhost:~$
```

**Los archivos que creaste hoy normalmente no los vas a poder buscar con el comando `locate`.** Si tienes acceso al sistema como usuario `root` (con la cuenta del administrador de sistema), **puede actualizar manualmente la base de datos `locate`** ejecutando el comando `updatedb`. Los usuarios regulares no pueden actualizar el archivo de base de datos.

También ten en cuenta que cuando utilizas el comando `locate` como un usuario normal, tu salida puede ser limitada debido a los permisos. En general, si no tienes acceso a un archivo o directorio en el sistema de ficheros debido a los permisos, el comando `locate` no devolverá esos nombres. Esta es una característica de seguridad diseñada para evitar que los usuarios «exploren» el sistema de ficheros utilizando el comando `locate`. El usuario `root` puede buscar cualquier archivo en la base de datos con el comando `locate`.

### 5.5.3 Contar el Número de Archivos

La salida del comando `locate` puede ser bastante grande. Cuando buscas un nombre de archivo, como `passwd`, el comando `locate` producirá cada archivo que contiene la cadena `passwd`, no sólo los archivos `passwd`.

En muchos casos, puede que quieras empezar listando cuántos archivos coincidirán. Lo puedes hacer mediante la opción `-c` del comando `locate`:

```
sysadmin@localhost:~$ locate -c passwd
97
sysadmin@localhost:~$
```

### 5.5.4 Limitando la Salida

Puedes **limitar la salida producida** por el comando `locate` mediante la opción `-b`. Esta opción sólo incluye los listados que contienen el término de búsqueda en *basename* del archivo. El basename es la parte del nombre de archivo que no incluye los nombres de directorio.

```
sysadmin@localhost:~$ locate -c -b passwd
83
sysadmin@localhost:~$
```

Como puedes ver en la salida anterior, todavía habrá muchos resultados cuando utilices la opción `-b`. Para limitar la salida aún más, coloca un carácter `\` delante del término de búsqueda. Este carácter limita la salida a los nombres de archivo que coincidan exactamente con el término:

```
sysadmin@localhost:~$ locate -b "\passwd"
/etc/passwd
/etc/cron.daily/passwd
/etc/pam.d/passwd
/usr/bin/passwd
/usr/share/doc/passwd
/usr/share/lintian/overrides/passwd
sysadmin@localhost:~$
```

## NDG Linux Essentials: Capítulo 5: Gestion de Archivos y Directorios

### 6.1 Introducción

Cuando trabajes en un sistema operativo Linux, necesitarás saber cómo manipular los archivos y los directorios. Algunas distribuciones de Linux tienen aplicaciones basadas en GUI que permiten gestionar los archivos, pero es importante saber cómo realizar estas operaciones a través de la línea de comandos.

La línea de comandos dispone de una amplia colección de comandos que permiten administrar los archivos. **En este capítulo aprenderás cómo listar los archivos en un directorio, así como, cómo copiar, mover y eliminar los archivos.**

Los conceptos básicos enseñados en este capítulo se ampliarán en los capítulos posteriores al ir cubriendo más comandos de manipulación de archivos, tales como los **comandos para ver archivos, comprimir archivos y establecer permisos de archivo.**

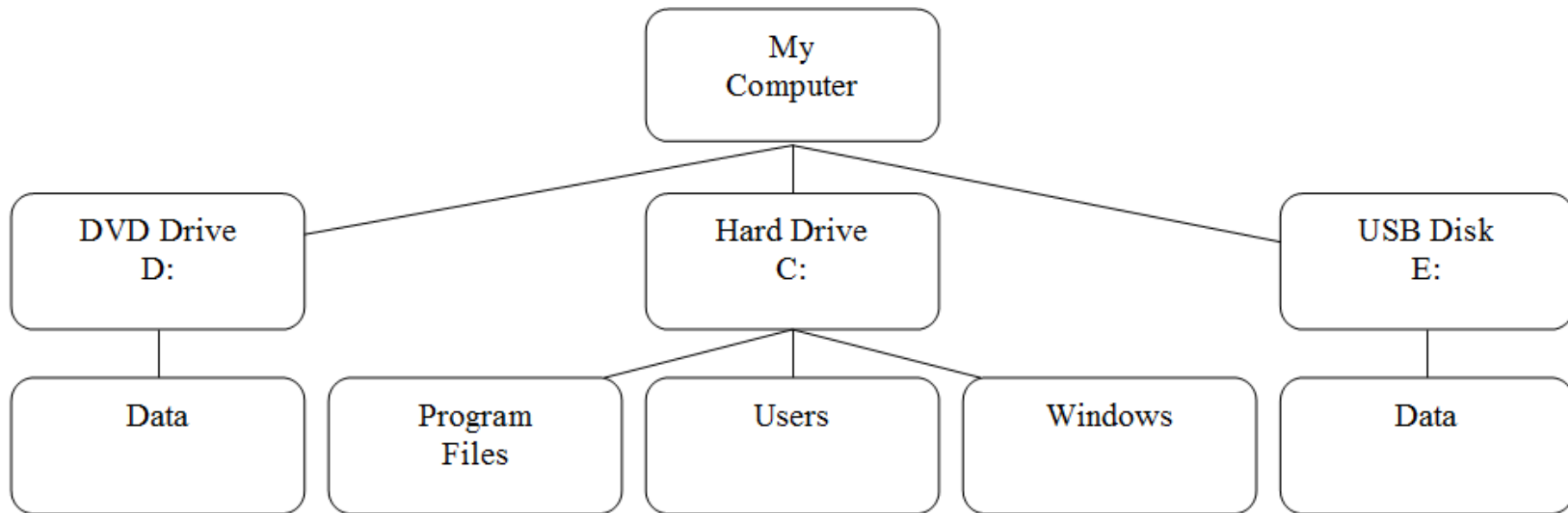
### 6.2 Comprender los Archivos y los Directorios

Los archivos se utilizan para **almacenar datos** tales como **texto, gráficos y programas**. Los directorios (También conocidos como «carpetas») se utilizan para **proporcionar una estructura de organización jerárquica**. Esta estructura es algo diferente a la que puedes estar acostumbrado si previamente trabajaste en los sistemas de Microsoft Windows.

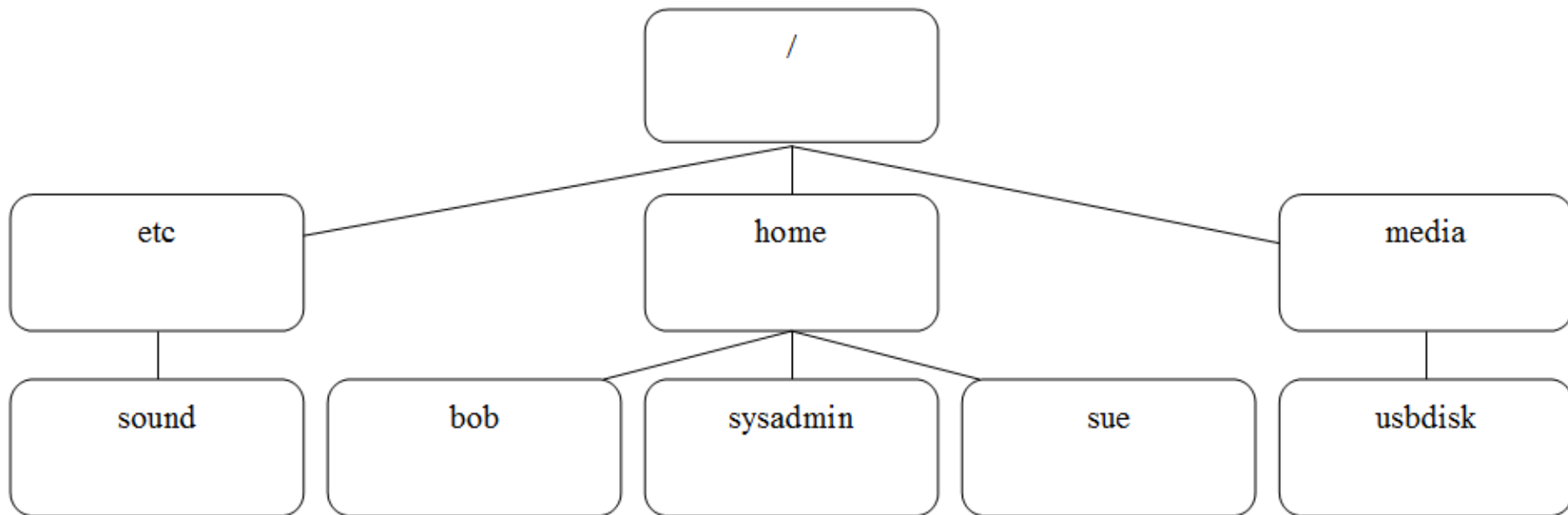


En un sistema Windows, el *nivel superior* de la estructura de directorios se llama *Este Equipo*. Cada dispositivo físico (disco duro, unidad de DVD, unidad USB, unidad de red, etc.) aparece en Este Equipo, cada uno asignado a una letra de unidad como C: o D:. Una representación visual de esta estructura:

Las estructuras de directorio que se muestran a continuación sirven solamente como ejemplos. Estos directorios pueden no estar presentes dentro del entorno de la máquina virtual de este curso.



Igual que Windows, la estructura de directorios de Linux tiene un nivel superior, sin embargo, no se llama Este Equipo, sino *directorio raíz* y su símbolo es el carácter `/`. También, en Linux no hay unidades; **cada dispositivo físico es accesible bajo un directorio, no una letra de unidad**. Una representación visual de una estructura de directorios típica de Linux:



La mayoría de los usuarios de Linux denominan esta estructura de directorios el *sistema de archivos*.

Para ver el sistema de archivos raíz, introduce `ls /`:

```

sysadmin@localhost:~$ ls /
bin  dev  home  lib  media  opt  root  sbin  selinux  sys  usr
boot  etc  init  lib64  mnt  proc  run  sbin???  srv  tmp  var

```

Observa que hay muchos directorios descriptivos incluyendo `/boot`, que contiene los archivos para arrancar la computadora.

### 6.2.1 El Directorio Path

Usando el gráfico en la sección anterior como un punto de referencia, verás que hay un directorio llamado `sound` bajo el directorio llamado `etc`, que se encuentra bajo el directorio `/`. Una manera más fácil de decir esto es refiriéndose a *la ruta*.

#### Para considerar:

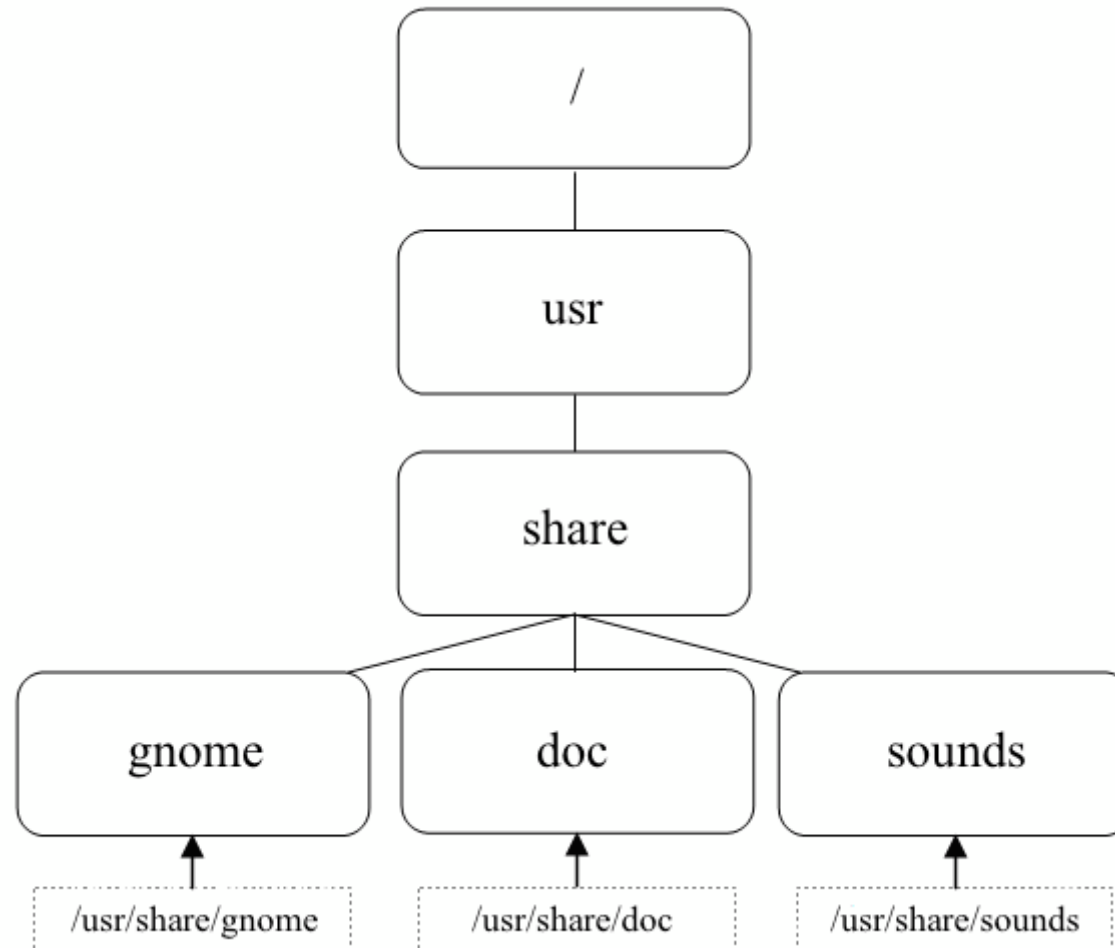
El directorio `/etc` originalmente significaba "etcetera" en la documentación temprana de Bell Labs y solía contener archivos que no pertenecían a ninguna ubicación. En las distribuciones modernas de Linux, el directorio `/etc` por lo general contiene los archivos de configuración estática como lo define por el Estándar de Jerarquía de Archivos (o «FHS», del inglés «Files Hierarchy Standard»).

Una ruta de acceso te permite especificar la ubicación exacta de un directorio. Para el directorio `sound` la ruta de acceso sería `/etc/sound`. El primer carácter `/` representa el directorio `root` (o «raíz» en español), mientras que cada siguiente carácter `/` se utiliza para separar los nombres de directorio.

Este tipo de ruta se llama la *ruta absoluta* (o «absolute path» en inglés). Con una ruta absoluta, siempre proporcionas direcciones a un directorio (o un archivo) a partir de la parte superior de la estructura de directorios, el directorio `root`. Más adelante en este capítulo cubriremos un tipo diferente de la ruta llamada la *ruta relativa* (o «relative path» en inglés).

**Nota:** Las estructuras de directorio que se muestran a continuación sirven solamente como ejemplos. Estos directorios pueden no estar presentes dentro del entorno de la máquina virtual de este curso.

La siguiente gráfica muestra tres rutas absolutas adicionales:



### 6.2.2 El Directorio Home

El término **home directory** (o «directorio de inicio» en español) a menudo causa confusión a los usuarios principiantes de Linux. Para empezar, en la mayoría de las distribuciones de Linux hay un directorio llamado **home** bajo el directorio **root: /home**.

Bajo de este directorio **/home** hay un directorio para cada **usuario** del sistema. El **nombre del directorio será el mismo que el nombre del usuario**, por lo que **un usuario llamado «bob» tendría un directorio home llamado /home/bob**.

Tu directorio home es un directorio muy importante. Para empezar, cuando abres un shell automáticamente te ubicarás en tu directorio home, en donde harás la mayor parte de tu trabajo.

Además, el directorio home es uno de los pocos directorios donde tienes el control total para crear y eliminar los archivos adicionales. La mayor parte de otros directorios en un sistema de archivos de Linux están protegidos con **file permissions** (o «permisos de archivos» en español), un tema que se tratará a detalle en un capítulo posterior.

En la mayoría de las distribuciones de Linux, los únicos usuarios que pueden acceder a los archivos en tu directorio home eres tú y el administrador del sistema (el usuario `root`). Esto se puede cambiar utilizando los permisos de archivo.

Tu **directorio** tiene incluso un **símbolo especial** que puedes usar para representarlo: `~`. Si tu **directorio** home es `/home/sysadmin`, puedes **simplemente introducir** `~` en la línea de comandos en lugar de `/home/sysadmin`. También puedes **referirte** al directorio home de otro **usuario usando la notación** `~usuario`, donde `usuario` es el nombre de la cuenta de usuario cuyo directorio home quieres consultar. Por ejemplo, `~bob` sería igual a `/home/bob`. Aquí, vamos a cambiar al directorio home del usuario:

```
sysadmin@localhost:~$ cd ~
sysadmin@localhost:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates
Videos
sysadmin@localhost:~$
```

Ten en cuenta que una lista revela los subdirectorios contenidos en el directorio home. Cambiar directorios requiere atención al detalle:

```
sysadmin@localhost:~$ cd downloads
-bash: cd: downloads: No such file or directory
sysadmin@localhost:~$
```

¿Por qué el anterior comando resultó en un error? Eso es porque los entornos de Linux son sensibles a mayúsculas y minúsculas. Cambiarnos al directorio `Downloads` requiere que la ortografía sea correcta - incluyendo la letra `D` mayúscula:

```
sysadmin@localhost:~$ cd Downloads
sysadmin@localhost:~/Downloads$
```

### 6.2.3 Directorio Actual

Tu **directorio actual** es el directorio donde estás **trabajando actualmente** en una terminal. Cuando primero abres una terminal, **el directorio actual debe ser tu directorio home**, pero esto **puede cambiar** mientras vas explorando el sistema de archivos y cambias a otros directorios.

Mientras estás en un entorno de línea de comandos, puedes determinar el directorio actual mediante el comando `pwd`:

```
sysadmin@localhost:~$ pwd
/home/sysadmin
sysadmin@localhost:~$
```

Adicionalmente, la mayoría de los sistemas tiene el prompt que visualiza el directorio actual del usuario por defecto:

```
[sysadmin@localhost ~]$
```

En el gráfico anterior, el carácter ~ indica el directorio actual. Como se mencionó anteriormente, el carácter ~ representa el directorio home.

Normalmente el sistema sólo muestra el nombre del directorio actual, no la ruta completa del directorio raíz hacia abajo. En otras palabras, si estuvieras en el directorio `/usr/share/doc`, tu prompt probablemente te proporcionará solamente el nombre `doc` el directorio actual. Si quieres la ruta completa, utiliza el comando `pwd`.

#### 6.2.4 Cambio de Directorios

Si te quieres cambiar a un directorio diferente, utiliza el comando `cd` (cambiar directorio). Por ejemplo, el siguiente comando cambiará el directorio actual a un directorio llamado `/etc/sound/events`:

```
sysadmin@localhost:~$ cd /etc/sound/events
sysadmin@localhost:/etc/sound/events$
```

Ten en cuenta que no vas a obtener salida si el comando `cd` tiene éxito. Este caso es uno de los de "ninguna noticia es buena noticia". Si tratas de cambiar a un directorio que no existe, recibirás un mensaje de error:

```
sysadmin@localhost:/etc/sound/events$ cd /etc/junk
-bash: cd: /etc/junk: No such file or directory
sysadmin@localhost:/etc/sound/events$
```

Si quieres volver a tu directorio home, puedes introducir el comando `cd` sin argumentos o usar el comando `cd` con el carácter ~ como argumento:

```
sysadmin@localhost:/etc/sound/events$ cd
sysadmin@localhost:~$ pwd
/home/sysadmin
sysadmin@localhost:~$ cd /etc
sysadmin@localhost:/etc$ cd ~
sysadmin@localhost:~$ pwd
/home/sysadmin
sysadmin@localhost:~$
```

#### 6.2.5 Nombres de Ruta Absoluta versus Relativa

Hay que recordar que una ruta de acceso es esencialmente una descripción de la ubicación de un archivo o un directorio en el sistema de archivos. Una ruta de acceso también se puede entender como las direcciones que indican al sistema donde se encuentra un archivo o un directorio. Por ejemplo, el comando `cd /etc/perl/Net` significa "cambiar al directorio `Net`, que encontrarás bajo el directorio `perl`, que encontrarás bajo el directorio `etc`, que encontrarás bajo el directorio `/`".

Cuando des un nombre de ruta que comienza en el directorio raíz, se llamará *ruta absoluta*. En muchos casos, proporcionar una ruta de acceso absoluta tiene sentido. Por ejemplo, si estás en tu directorio `home` y quieres ir al directorio `/etc/perl/Net`, entonces proporcionar una ruta de acceso absoluta al comando `cd` tiene sentido:

```
sysadmin@localhost:~$ cd /etc/perl/Net
sysadmin@localhost:/etc/perl/Net$
```

Sin embargo, ¿Qué pasa si estás en el directorio `/etc/perl` y quieres ir al directorio `/etc/perl/Net`? Sería tedioso introducir la ruta completa para llegar a un directorio que es sólo un nivel más abajo de tu ubicación actual. En una situación como ésta, vas a utilizar una *ruta relativa*:

```
sysadmin@localhost:/etc/perl$ cd Net
sysadmin@localhost:/etc/perl/Net$
```

Una ruta de acceso relativa proporciona direcciones usando tu **ubicación actual** como un punto de referencia. Recuerda que esto es diferente de las rutas absolutas, que siempre requieren que utilices el **directorio raíz** como punto de referencia.

Existe una técnica útil de ruta de acceso relativa que se puede utilizar para subir un nivel en la estructura de directorios: el directorio `..`. Sin importar en qué directorio estás, el comando `..` siempre representa un directorio arriba que el directorio actual (con la excepción de cuando estás en el directorio `/`):

```
sysadmin@localhost:/etc/perl/Net$ pwd
/etc/perl/Net
sysadmin@localhost:/etc/perl/Net$ cd ..
sysadmin@localhost:/etc/perl$ pwd
/etc/perl
sysadmin@localhost:/etc/perl$
```

A veces usar las rutas de acceso relativas es una mejor opción que rutas de acceso absolutas, sin embargo esto no siempre es el caso. ¿Qué pasa si estás en el directorio `/etc/perl/Net` y quieres ir al directorio `/usr/share/doc`? Utilizando una ruta absoluta, se ejecutaría el comando `cd /usr/share/doc`. Utilizando una ruta relativa, se ejecutaría el comando `cd ../../../../usr/share/doc`:

```
sysadmin@localhost:/etc/perl/Net$ cd
sysadmin@localhost:~$ cd /etc/perl/Net
sysadmin@localhost:/etc/perl/Net$ cd ../../../../usr/share/doc
sysadmin@localhost:/usr/share/doc$ pwd
/usr/share/doc
sysadmin@localhost:/usr/share/doc$
```

**Nota:** Las rutas relativas y absolutas no sólo sirven para el comando `cd`. Siempre cuando especificas un archivo o un directorio, puedes utilizar las rutas de acceso relativas o absolutas.

Mientras que el doble punto (..) se utiliza para referirse al directorio arriba del directorio actual, el punto solo (.) se usa para referirse al directorio actual. No tendría sentido para un administrador moverse al directorio actual introduciendo `cd .` (aunque en realidad funciona). Es más útil referirse a un elemento en el directorio actual usando la notación `./`. Por ejemplo:

```
sysadmin@localhost:~$ pwd
/home/sysadmin
sysadmin@localhost:~$ cd ./Downloads/
sysadmin@localhost:~/Downloads$ pwd
/home/sysadmin/Downloads
sysadmin@localhost:~/Downloads$ cd ..
sysadmin@localhost:~$ pwd
/home/sysadmin
sysadmin@localhost:~$
```

Nota: Este uso del punto solo (.), como punto de referencia, no se debe confundir con su uso al principio de un nombre de archivo. Leer más sobre los archivos ocultos en la sección 6.4.2.

### 6.3 Listado de los Archivos en un Directorio

Ahora que te puedes mover de un directorio a otro, querrás visualizar el contenido de estos directorios. El comando `ls` (`ls` es la abreviatura para listar) puede utilizarse para mostrar el contenido de un directorio, así como toda la información sobre los archivos que están dentro de un directorio.

Por sí mismo, el comando `ls` listará los archivos en el directorio actual:

```
sysadmin@localhost:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates
Videos
sysadmin@localhost:~$
```

#### 6.3.1 Lista de Colores

Hay muchos tipos de archivos en Linux. Según vayas aprendiendo más sobre Linux, descubrirás muchos de estos tipos. A continuación tenemos un breve resumen de algunos de los tipos de archivo más comunes:

| Tipo                                       | Descripción  |
|--|--|
| plain file (o «archivo simple» en español) | Un archivo que no es un tipo de archivo especial; también se llama un archivo normal |
| directory (o «directorio» en español)      | Un directorio de archivos (contiene otros archivos)                                  |
| executable (o «ejecutable» en español)     | Un archivo que se puede ejecutar como un programa                                    |
| symbolic link                              | Un archivo que apunta a otro archivo (o «enlace simbólico» en español)               |

En muchas distribuciones de Linux, las cuentas de usuario regulares son modificadas de tal manera que el comando `ls` muestre los nombres de archivo, codificados por colores según el tipo de archivo. Por ejemplo, los directorios pueden aparecer en azul, archivos ejecutables pueden verse en verde, y enlaces simbólicos pueden ser visualizados en cian (azul claro).

Esto no es un comportamiento normal para el comando `ls`, sino algo que sucede cuando se utiliza la opción `--color` para el comando `ls`. La razón por la que el comando `ls` parece realizar automáticamente estos colores, es que hay un alias para el comando `ls` para que se ejecute con la opción `--color`:

```
sysadmin@localhost:~$ alias
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -aLF'
alias ls='ls --color=auto'
sysadmin@localhost:~$
```



Como puedes ver en la salida anterior, cuando se ejecuta el comando `ls`, en realidad se ejecuta el comando `ls --color=auto`.

En algunos casos, puede que no quieras ver todos los colores (a veces te pueden distraer un poco). Para evitar el uso de los alias, coloca un carácter de barra invertida `\` antes de tu comando:

```
sysadmin@localhost:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates
Videos

sysadmin@localhost:~$ \ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates
Videos

sysadmin@localhost:~$
```

### 6.3.2 Lista de los Archivos Ocultos

Cuando utilizas el comando `ls` para mostrar el contenido de un directorio, no todos los archivos se muestran automáticamente. El comando `ls` no muestra los *archivos ocultos* de manera predeterminada. Un archivo oculto es cualquier archivo (o directorio) que comienza con un punto `.`.

Para mostrar todos los archivos, incluyendo los archivos ocultos, utiliza la opción `-a` para el comando `ls`:

```
sysadmin@localhost:~$ ls -a
.          .bashrc    .selected_editor  Downloads  Public
..         .cache     Desktop           Music      Templates
.bash_logout .profile  Documents         Pictures   Videos
```

¿Por qué los archivos están ocultos en primer lugar? La mayoría de los archivos ocultos son *archivos de personalización*, diseñados para personalizar la forma en la que Linux, el shell o los programas funcionan. Por ejemplo, el archivo `.bashrc` en tu directorio home personaliza las características del shell, tales como la creación o modificación de las variables y los alias.

Estos archivos de personalización no son con los que regularmente trabajas. Hay muchos de ellos, como puedes ver, y visualizarlos hará más difícil encontrar los archivos con los que regularmente trabajas. Así que, el hecho de que están ocultos es para tu beneficio.

### 6.3.3 Listado con Visualización Larga

Existe información sobre cada archivo, llamada metadata (o «metadatos» en español), y visualizarla a veces resulta útil. Esto puede incluir datos de quién es el dueño de un archivo, el tamaño de un archivo y la última vez que se modificó el contenido de un archivo. Puedes visualizar esta información mediante el uso de la opción `-l` para el comando `ls`:

```
sysadmin@localhost:~$ ls -l
total 0

drwxr-xr-x 1 sysadmin sysadmin 0 Jan 29  2015 Desktop
drwxr-xr-x 1 sysadmin sysadmin 0 Jan 29  2015 Documents
```

```
drwxr-xr-x 1 sysadmin sysadmin 0 Jan 29 2015 Downloads
drwxr-xr-x 1 sysadmin sysadmin 0 Jan 29 2015 Music
drwxr-xr-x 1 sysadmin sysadmin 0 Jan 29 2015 Pictures
drwxr-xr-x 1 sysadmin sysadmin 0 Jan 29 2015 Public
drwxr-xr-x 1 sysadmin sysadmin 0 Jan 29 2015 Templates
drwxr-xr-x 1 sysadmin sysadmin 0 Jan 29 2015 Videos
sysadmin@localhost:~$
```

En la salida anterior, cada línea describe metadatos sobre un solo archivo. A continuación se describe cada uno de los campos de datos que verás en la salida del comando `ls -l`:

#### File type

```
d-rwxr-xr-x. 2 sysadmin sysadmin 4096 Aug 7 13:33 Desktop
```

The first character of each output line indicated the type of file. Common file types include:

- d** = directory
- = plain file
- l** = symbolic link

«**Tipo de archivo.** El primer carácter de cada línea de salida indica el tipo de archivo. Tipos de archivo comunes incluyen: **d**= directorio, **-**= archivo simple, **l**= enlace simbólico»

#### Permissions

```
d-rwxr-xr-x. 2 sysadmin sysadmin 4096 Aug 7 13:33 Desktop
```

The next ten characters will demonstrate the *permissions* of the file. The permissions are used to determine who has access to the file. They will be covered in detail in a later chapter.

«**Permisos.** Los próximos diez caracteres demostrarán *los permisos* del archivo. Los permisos se utilizan para determinar quién tiene acceso al archivo. Esto será cubierto a detalle en un capítulo posterior.»

**Hard link count**

```
drwxr-xr-x. 2 sysadmin sysadmin 4096 Aug  7 13:33 Desktop
```

The hard link count of a file is used to demonstrate how many hard links there are to this file. Links are more of an administrator topic and not covered in this course.

«Conteo de enlaces físicos. El conteo de enlaces físicos de un archivo se usa para demostrar cuantos enlaces físicos hacia este archivo existen. Los enlaces son más que nada un tema de administrador por lo que no son cubiertos en este curso.»

**User owner**

```
drwxr-xr-x. 2 sysadmin sysadmin 4096 Aug  7 13:33 Desktop
```

Every file is owned by a user account. This is important because the owner has the rights to set permissions on a file and the owner has his/her own permissions on the file. Permissions will be covered in detail in a later chapter.

«**Usuario propietario.** Cada archivo es propiedad de una cuenta de usuario. Esto es importante porque el propietario tiene los derechos para establecer permisos en un archivo y el propietario tiene sus propios permisos en el archivo. Los permisos se cubrirán a detalle en un capítulo posterior.»

**Group owner**

```
drwxr-xr-x. 2 sysadmin sysadmin 4096 Aug  7 13:33 Desktop
```

Every file is owned by a group account. This is important because any member of this group will have special access to this file based on the group permissions on the file. Permissions will be covered in detail in a later chapter.

«**Grupo propietario.** Cada archivo es propiedad de un grupo. Esto es importante porque cualquier miembro de este grupo tendrá acceso especial al archivo basado en los permisos de grupo del archivo. Los permisos se cubrirán a detalle en un capítulo posterior.»

**File size**

```
drwxr-xr-x. 2 sysadmin sysadmin 4096 Aug  7 13:33 Desktop
```

This field describes the size of the file in bytes. Note: For directories, this value does not describe the total size of the directory, but rather how many bytes are reserved to keep track of the filenames in the directory (in other words, ignore this field for directories).

«Tamaño de archivo. Este campo describe el tamaño de un archivo en bytes. Nota: En el caso de los directorios, este valor no describe el tamaño total del directorio, más bien, cuántos bytes están reservados para mantenerse al corriente con los nombres de archivo en el directorio (en otras palabras, ignora este campo en los directorios).»

**Modification timestamp**

```
drwxr-xr-x. 2 sysadmin sysadmin 4096 Aug  7 13:33 Desktop
```

This field indicates the last time the file contents were modified. For directories this timestamp indicates the last time a file was added or deleted from the directory.

«Hora de modificación. Este campo indica la última hora en la que el contenido del archivo fue modificado. En el caso de los directorios, indica la última vez que se agregó o eliminó un archivo dentro del directorio.»

**Name**

```
drwxr-xr-x. 2 sysadmin sysadmin 4096 Aug  7 13:33 Desktop
```

The last field is the name of the file or directory.

«Nombre. El último campo es el nombre del archivo o directorio.»

**6.3.3.1 Tamaños Legibles**

Cuando visualizas los tamaños de los archivos con la opción `-l` del comando `ls` obtienes los tamaños de los archivo en bytes. Para archivos de texto, un byte es 1 carácter.

Para archivos más pequeños, los tamaños en byte están bien. Sin embargo, para los archivos más grandes es difícil comprender qué tan grande es el archivo. Por ejemplo, considera la salida del siguiente comando:

```
sysadmin@localhost:~$ ls -l /usr/bin/omshell
-rwxr-xr-c 1 root root 1561400 Oct 9 2012 /usr/bin/omshell
sysadmin@localhost:~$
```

Como puedes ver, es difícil de determinar el tamaño del archivo en bytes. ¿Un archivo 1561400 es grande o pequeño? Parece bastante grande, pero es difícil de determinar su tamaño utilizando los bytes.

Piénsalo de esta manera: si alguien diera la distancia entre Boston y Nueva York utilizando centímetros, ese valor esencialmente no tendría sentido porque una distancia como ésta la piensas en términos de kilómetros.

Sería mejor si el tamaño del archivo fuese presentado en un tamaño más fácilmente legible, tal como megabytes o gigabytes. Para lograr esto, añade la opción `-h` al comando `ls`:

```
sysadmin@localhost:~$ ls -lh /usr/bin/omshell
-rwxr-xr-c 1 root root 1.5M Oct 9 2012 /usr/bin/omshell
sysadmin@localhost:~$
```

**Importante:** Debes utilizar la opción `-h` junto con la opción `-l`.

### 6.3.4 Lista de Directorios

Cuando se utiliza el comando `ls -d`, se refiere al directorio actual y no al contenido dentro de él. Sin otras opciones, es algo sin sentido, aunque es importante tener en cuenta que al directorio actual siempre se refiere con un solo punto (`.`):

```
sysadmin@localhost:~$ ls -d
.
```

Para utilizar el comando `ls -d` de una manera significativa tienes que añadir la opción `-l`. En este caso, ten en cuenta que el primer comando muestra los detalles de los contenidos en el directorio `/home/sysadmin`, mientras que el segundo lista el directorio `/home/sysadmin`.

```
sysadmin@localhost:~$ ls -l
total 0
drwxr-xr-x 1 sysadmin sysadmin  0 Apr 15  2015 Desktop
drwxr-xr-x 1 sysadmin sysadmin  0 Apr 15  2015 Documents
drwxr-xr-x 1 sysadmin sysadmin  0 Apr 15  2015 Downloads
drwxr-xr-x 1 sysadmin sysadmin  0 Apr 15  2015 Music
drwxr-xr-x 1 sysadmin sysadmin  0 Apr 15  2015 Pictures
drwxr-xr-x 1 sysadmin sysadmin  0 Apr 15  2015 Public
drwxr-xr-x 1 sysadmin sysadmin  0 Apr 15  2015 Templates
drwxr-xr-x 1 sysadmin sysadmin  0 Apr 15  2015 Videos
drwxr-xr-x 1 sysadmin sysadmin 420 Apr 15  2015 test

sysadmin@localhost:~$ ls -ld
drwxr-xr-x 1 sysadmin sysadmin 224 Nov  7 17:07 .
```

```
sysadmin@localhost:~$
```

Observa **el punto solo** al final de la segunda lista larga. Esto **indica que el directorio actual está en la lista** y no el **contenido**.

### 6.3.5 Listado Recursivo

Habrás momentos cuando quieras visualizar **todos los archivos en un directorio**, así como **todos los archivos en todos los subdirectorios bajo un directorio**. Esto se llama **listado recursivo**.

Para realizar un listado recursivo, utiliza la opción **-R** para el comando **ls**:

**Nota:** La salida que se muestra a continuación variará de los resultados que verás si ejecutas el comando en el entorno de la máquina virtual de este curso.

```
sysadmin@localhost:~$ ls -R /etc/ppp
/etc/ppp:
chap-secrets  ip-down.ipv6to4  ip-up.ipv6to4  ipv6-up  pap-secrets
ip-down       ip-up            ipv6-down      options  peers

/etc/ppp/peers:
sysadmin@localhost:~$
```

Ten en cuenta que en el ejemplo anterior, los archivos en el directorio `/etc/ppp` se listaron primero. Después de eso, se listan los archivos en el directorio `/etc/ppp/peers` (no hubo ningún archivo en este caso, pero si hubiera encontrado cualquier archivo en este directorio, se habría visualizado).

Ten cuidado con esta opción; por ejemplo, ejecutando el comando **ls -R /** se listarían todos los archivos del sistema de archivos, incluyendo todos los archivos de cualquier dispositivo USB y DVD en el sistema. Limita el uso de la opción **-R** para estructuras de directorio más pequeñas.

### 6.3.6 Ordenar un Listado

De forma predeterminada, el comando **ls** ordena los archivos alfabéticamente por nombre de archivo. A veces, puede ser útil ordenar los archivos utilizando diferentes criterios.

Para ordenar los **archivos por tamaño**, podemos utilizar la opción **-S**. Observa la diferencia en la salida de los dos siguientes comandos:

```
sysadmin@localhost:~$ ls /etc/ssh
moduli          ssh_host_dsa_key.pub  ssh_host_rsa_key  sshd_config
ssh_config      ssh_host_ecdsa_key    ssh_host_rsa_key.pub
ssh_host_dsa_key ssh_host_ecdsa_key.pub ssh_import_id

sysadmin@localhost:~$ ls -S /etc/ssh
moduli          ssh_host_dsa_key      ssh_host_ecdsa_key
sshd_config     ssh_host_dsa_key.pub  ssh_host_ecdsa_key.pub
ssh_host_rsa_key ssh_host_rsa_key.pub
```

```
ssh_config      ssh_import_id
sysadmin@localhost:~$
```

Aparecen los mismos archivos y directorios, pero en un orden diferente. Mientras que la opción `-S` trabaja por sí misma, realmente no puedes decir que la salida está ordenada por tamaño, por lo que es más útil cuando se utiliza con la opción `-l`. El siguiente comando listará los archivos del mayor al menor y mostrará el tamaño real del archivo.

```
sysadmin@localhost:~$ ls -lS /etc/ssh
total 160
-rw-r--r-- 1 root root 125749 Apr 29 2014 moduli
-rw-r--r-- 1 root root 2489 Jan 29 2015 sshd_config
-rw----- 1 root root 1675 Jan 29 2015 ssh_host_rsa_key
-rw-r--r-- 1 root root 1669 Apr 29 2014 ssh_config
-rw----- 1 root root 668 Jan 29 2015 ssh_host_dsa_key
-rw-r--r-- 1 root root 607 Jan 29 2015 ssh_host_dsa_key.pub
-rw-r--r-- 1 root root 399 Jan 29 2015 ssh_host_rsa_key.pub
-rw-r--r-- 1 root root 302 Jan 10 2011 ssh_import_id
-rw----- 1 root root 227 Jan 29 2015 ssh_host_ecdsa_key
-rw-r--r-- 1 root root 179 Jan 29 2015 ssh_host_ecdsa_key.pub
sysadmin@localhost:~$
```

También puede ser útil usar la opción `-lh` para mostrar los tamaños de los archivos de una manera legible:

```
sysadmin@localhost:~$ ls -lSh /etc/ssh
total 160K
-rw-r--r-- 1 root root 123K Apr 29 2014 moduli
-rw-r--r-- 1 root root 2.5K Jan 29 2015 sshd_config
-rw----- 1 root root 1.7K Jan 29 2015 ssh_host_rsa_key
-rw-r--r-- 1 root root 1.7K Apr 29 2014 ssh_config
-rw----- 1 root root 668 Jan 29 2015 ssh_host_dsa_key
-rw-r--r-- 1 root root 607 Jan 29 2015 ssh_host_dsa_key.pub
-rw-r--r-- 1 root root 399 Jan 29 2015 ssh_host_rsa_key.pub
```

```
-rw-r--r-- 1 root root 302 Jan 10 2011 ssh_import_id
-rw----- 1 root root 227 Jan 29 2015 ssh_host_ecdsa_key
-rw-r--r-- 1 root root 179 Jan 29 2015 ssh_host_ecdsa_key.pub

sysadmin@localhost:~$
```

También es posible ordenar los archivos según el momento en que se modificaron. Puedes hacer esto mediante la opción `-t`.

La opción `-t` listará los archivos modificados más recientemente en primer lugar. Esta opción puede utilizarse sola, pero otra vez, es generalmente más útil cuando se combina con la opción `-l`:

```
sysadmin@localhost:~$ ls -tl /etc/ssh
total 160
-rw----- 1 root root 668 Jan 29 2015 ssh_host_dsa_key
-rw-r--r-- 1 root root 607 Jan 29 2015 ssh_host_dsa_key.pub
-rw----- 1 root root 227 Jan 29 2015 ssh_host_ecdsa_key
-rw-r--r-- 1 root root 179 Jan 29 2015 ssh_host_ecdsa_key.pub
-rw----- 1 root root 1675 Jan 29 2015 ssh_host_rsa_key
-rw-r--r-- 1 root root 399 Jan 29 2015 ssh_host_rsa_key.pub
-rw-r--r-- 1 root root 2489 Jan 29 2015 sshd_config
-rw-r--r-- 1 root root 125749 Apr 29 2014 moduli
-rw-r--r-- 1 root root 1669 Apr 29 2014 ssh_config
-rw-r--r-- 1 root root 302 Jan 10 2011 ssh_import_id

sysadmin@localhost:~$
```

**Es importante recordar que la fecha de modificación de los directorios representa la última vez que un archivo se agrega o se elimina del directorio.**

Si los archivos en un directorio se modificaron hace muchos días o meses, puede ser más difícil de decir exactamente cuándo fueron modificados, ya que para los archivos más antiguos sólo se proporciona la fecha. Para una información más detallada de la hora de modificación puedes utilizar la opción `--full-time` que visualiza la fecha y la hora completas (incluyendo horas, segundos, minutos...):

```
sysadmin@localhost:~$ ls -t --full-time /etc/ssh
total 160
-rw----- 1 root root 668 2015-01-29 03:17:33.000000000 +0000 ssh_host_dsa_key
-rw-r--r-- 1 root root 607 2015-01-29 03:17:33.000000000 +0000 ssh_host_dsa_key.pub
-rw----- 1 root root 227 2015-01-29 03:17:33.000000000 +0000 ssh_host_ecdsa_key
```



```

-rw-r--r-- 1 root root    179 2015-01-29 03:17:33.000000000 +0000 ssh_host_ecdsa_key.pub
-rw----- 1 root root   1675 2015-01-29 03:17:33.000000000 +0000 ssh_host_rsa_key
-rw-r--r-- 1 root root    399 2015-01-29 03:17:33.000000000 +0000 ssh_host_rsa_key.pub
-rw-r--r-- 1 root root   2489 2015-01-29 03:17:33.000000000 +0000 sshd_config
-rw-r--r-- 1 root root 125749 2014-04-29 23:58:51.000000000 +0000 moduli
-rw-r--r-- 1 root root    1669 2014-04-29 23:58:51.000000000 +0000 ssh_config
-rw-r--r-- 1 root root    302 2011-01-10 18:48:29.000000000 +0000 ssh_import_id

sysadmin@localhost:~$

```

La opción `--full-time` asumirá automáticamente la opción `-l`.

Es posible realizar una **ordenación inversa** con las opciones `-s` o `-t` mediante la opción `-r`. El siguiente comando ordena los archivos por tamaño, de menor a mayor:

```

sysadmin@localhost:~$ ls -lrS /etc/ssh

total 160

-rw-r--r-- 1 root root    179 Jan 29  2015 ssh_host_ecdsa_key.pub
-rw----- 1 root root    227 Jan 29  2015 ssh_host_ecdsa_key
-rw-r--r-- 1 root root    302 Jan 10  2011 ssh_import_id
-rw-r--r-- 1 root root    399 Jan 29  2015 ssh_host_rsa_key.pub
-rw-r--r-- 1 root root    607 Jan 29  2015 ssh_host_dsa_key.pub
-rw----- 1 root root    668 Jan 29  2015 ssh_host_dsa_key
-rw-r--r-- 1 root root   1669 Apr 29  2014 ssh_config
-rw----- 1 root root   1675 Jan 29  2015 ssh_host_rsa_key
-rw-r--r-- 1 root root   2489 Jan 29  2015 sshd_config
-rw-r--r-- 1 root root 125749 Apr 29  2014 moduli

sysadmin@localhost:~$

```

El siguiente comando listará los archivos por fecha de modificación, de la más antigua a la más reciente:

```

sysadmin@localhost:~$ ls -lrt /etc/ssh

total 160

-rw-r--r-- 1 root root    302 Jan 10  2011 ssh_import_id
-rw-r--r-- 1 root root   1669 Apr 29  2014 ssh_config

```

```
-rw-r--r-- 1 root root 125749 Apr 29 2014 moduli
-rw-r--r-- 1 root root 2489 Jan 29 2015 sshd_config
-rw-r--r-- 1 root root 399 Jan 29 2015 ssh_host_rsa_key.pub
-rw----- 1 root root 1675 Jan 29 2015 ssh_host_rsa_key
-rw-r--r-- 1 root root 179 Jan 29 2015 ssh_host_ecdsa_key.pub
-rw----- 1 root root 227 Jan 29 2015 ssh_host_ecdsa_key
-rw-r--r-- 1 root root 607 Jan 29 2015 ssh_host_dsa_key.pub
-rw----- 1 root root 668 Jan 29 2015 ssh_host_dsa_key

sysadmin@localhost:~$
```

### 6.3.7 Listado con Globs

En un capítulo anterior, vimos el uso de los globs para los archivos para buscar coincidencias de los nombres de archivo utilizando los caracteres comodines. Por ejemplo, hemos visto que puedes listar todos los archivos en el directorio `/etc` que comienzan con la letra `e` utilizando el siguiente comando:

```
sysadmin@localhost:~$ echo /etc/e*
/etc/encript.cfg /etc/environment /etc/ethers /etc/event.d /etc/exports

sysadmin@localhost:~$
```

Ahora que sabes que el comando `ls` se utiliza normalmente para listar los archivos en un directorio, el uso del comando `echo` puede parecer una elección extraña. Sin embargo, hay algo sobre el comando `ls` que pudo haber causado confusión mientras hablamos sobre los globs. Esta «función» también puede causar problemas cuando intentas listar los archivos utilizando los patrones glob.

Ten en cuenta que **es el shell**, no los comandos `echo` o `ls`, el que expande el patrón glob a los nombres de archivo correspondientes. En otras palabras, cuando introduces el comando `echo /etc/e*`, lo que el shell hizo **antes** de ejecutar el comando `echo` fue reemplazar el `e*` por todos los archivos y directorios dentro del directorio `/etc` que coinciden con el patrón.

Por lo tanto, si ejecutaras el comando `ls /etc/e*`, lo que el shell realmente haría, sería lo siguiente:

```
ls /etc/encript.cfg /etc/environment /etc/ethers /etc/event.d /etc/exports
```

Cuando el comando `ls` ve varios argumentos, realiza una operación de listado en cada elemento por separado. En otras palabras, el comando `ls /etc/encript.cfg /etc/environment` es esencialmente igual a `ls /etc/encript.cfg; ls /etc/environment`.

Ahora considera lo que sucede cuando se ejecuta el comando `ls` en un archivo, tal como `encript.cfg`:

```
sysadmin@localhost:~$ ls /etc/encript.cfg
/etc/encript.cfg

sysadmin@localhost:~$
```

Como puedes ver, ejecutando el comando `ls` en un solo archivo se imprime el nombre del archivo. Generalmente esto es útil si quieres ver los detalles acerca de un archivo mediante la opción `-l` del comando `ls`:

```
sysadmin@localhost:~$ ls -l /etc/enscript.cfg
-r--r--r--. 1 root root 4843 Nov 11 2010 /etc/enscript.cfg
sysadmin@localhost:~$
```

Sin embargo, ¿Qué ocurre si el comando `ls` recibe un nombre de directorio como un argumento? En este caso, la salida del comando es diferente a que si el argumento es un nombre de archivo:

```
sysadmin@localhost:~$ ls /etc/event.d
ck-log-system-restart ck-log-system-start ck-log-system-stop
sysadmin@localhost:~$
```

Si proporcionas un nombre de directorio como argumento del comando `ls`, el comando mostrará el contenido del directorio (los nombres de los archivos en el directorio), y no sólo proporcionará el nombre del directorio. Los nombres de los archivos, que se ven en el ejemplo anterior, son los nombres de los archivos en el directorio `/etc/event.d`.

¿Por qué ésto es un problema al utilizar los globs? Considera el siguiente resultado:

```
sysadmin@localhost:~$ ls /etc/e*
/etc/encrypt.cfg /etc/environment /etc/ethers /etc/event.d /etc/exports
/etc/event.d:
ck-log-system-restart ck-log-system-start ck-log-system-stop
sysadmin@localhost:~$
```

Como puedes ver, cuando el comando `ls` ve un nombre de archivo como argumento, sólo muestra el nombre del archivo. Sin embargo, para cualquier directorio, mostrará el contenido del directorio, y no sólo el nombre del directorio.

Esto se vuelve aún más confuso en una situación como la siguiente:

```
sysadmin@localhost:~$ ls /etc/ev*
ck-log-system-restart ck-log-system-start ck-log-system-stop
sysadmin@localhost:~$
```

En el **ejemplo anterior**, parece que el comando `ls` es **simplemente incorrecto**. Pero lo que realmente sucedió es que lo único que coincide con el glob `etc/ev *` es el directorio `/etc/event.d`. Por lo tanto, el comando `ls` muestra sólo los archivos en ese directorio.

Hay una solución simple a este problema: al utilizar los argumentos glob con el comando `ls`, utiliza siempre la opción `-d`. Cuando utilizas la opción `-d`, el comando `ls` no muestra el contenido de un directorio, sino más bien el nombre del directorio:

```
sysadmin@localhost:~$ ls -d /etc/e*
/etc/encrypt.cfg /etc/environment /etc/ethers /etc/event.d /etc/exports
```

```
sysadmin@localhost:~$
```

## 6.4 Copiar los Archivos

El comando `cp` se utiliza para **copiar los archivos**. Requiere especificar un **origen y un destino**. La estructura del comando es la siguiente:

```
cp [fuente] [destino]
```

La *fuentes* («source» en inglés) es el archivo que quieres copiar. El *destino* («destination» en inglés) es la ubicación en donde quieres poner la copia. Cuando el comando es exitoso, el comando `cp` no tendrá ninguna salida (ninguna noticia es buena noticia). El siguiente comando copiará el archivo `/etc/hosts` a tu directorio home:

```
sysadmin@localhost:~$ cp /etc/hosts ~
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures   Templates  hosts
Documents  Music      Public     Videos
sysadmin@localhost:~$
```

**Recuerda:** El carácter `~` representa el directorio home.

### 6.4.1 El Modo Verbose

La opción `-v` hará que el comando `cp` produzca la salida en caso de ser exitoso. La opción `-v` se refiere al comando verbose:

```
sysadmin@localhost:~$ cp -v /etc/hosts ~
'/etc/hosts' -> '/home/sysadmin/hosts'
sysadmin@localhost:~$
```

Cuando el destino es un directorio, el nuevo archivo resultante tendrá el mismo nombre que el archivo original. Si quieres que el nuevo archivo tenga un nombre diferente, debes proporcionar el nuevo nombre como parte del destino:

```
sysadmin@localhost:~$ cp /etc/hosts ~/hosts.copy
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures   Templates  hosts
Documents  Music      Public     Videos     hosts.copy
sysadmin@localhost:~$
```

### 6.4.1 El Modo Verbose

La opción `-v` hará que el comando `cp` produzca la salida en caso de ser exitoso. La opción `-v` se refiere al comando verbose:

```
sysadmin@localhost:~$ cp -v /etc/hosts ~
`/etc/hosts' -> `/home/sysadmin/hosts'
sysadmin@localhost:~$
```

Cuando el destino es un directorio, el nuevo archivo resultante tendrá el mismo nombre que el archivo original. Si quieres que el nuevo archivo tenga un nombre diferente, debes proporcionar el nuevo nombre como parte del destino:

```
sysadmin@localhost:~$ cp /etc/hosts ~/hosts.copy
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures   Templates  hosts
Documents  Music      Public     Videos    hosts.copy
sysadmin@localhost:~$
```

#### 6.4.2 Evitar Sobrescribir los Datos

El comando `cp` puede ser destructivo para los datos si el archivo de destino ya existe. En el caso donde el archivo de destino existe, el comando `cp` sobrescribe el contenido del archivo existente con el contenido del archivo fuente. Para ilustrar este problema, primero se crea un nuevo archivo en el directorio home `sysadmin` copiando un archivo existente:

```
sysadmin@localhost:~$ cp /etc/skel/.bash_logout ~/example.txt
sysadmin@localhost:~$
```

Visualiza la salida del comando `ls` para ver el archivo y visualiza el contenido del archivo utilizando el comando `more`:

```
sysadmin@localhost:~$ cp /etc/skel/.bash_logout ~/example.txt
sysadmin@localhost:~$ ls -l example.txt
-rw-rw-r--. 1 sysadmin sysadmin 18 Sep 21 15:56 example.txt
sysadmin@localhost:~$ more example.txt
# ~/.bash_logout: executed by bash(1) when login shell exits.

sysadmin@localhost:~$ cp -i /etc/hosts ~/example.txt
cp: overwrite `/home/sysadmin/example.txt'? n
sysadmin@localhost:~$ ls -l example.txt
-rw-rw-r--. 1 sysadmin sysadmin 18 Sep 21 15:56 example.txt
sysadmin@localhost:~$ more example.txt
```

```
# ~/.bash_logout: executed by bash(1) when login shell exits.
```

```
sysadmin@localhost:~$
```

En el siguiente ejemplo verás que el comando `cp` destruye el contenido original del archivo `ejemplo.txt`. Observa que una vez finalizado el comando `cp`, el tamaño del archivo es diferente (158 bytes en lugar de 18) del original y los contenidos también son diferentes:

```
sysadmin@localhost:~$ cp /etc/hosts ~/example.txt
```

```
sysadmin@localhost:~$ ls -l example.txt
```

```
-rw-rw-r--. 1 sysadmin sysadmin 158 Sep 21 14:11 example.txt
```

```
sysadmin@localhost:~$ cat example.txt
```

```
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
```

```
:::1        localhost localhost.localdomain localhost6 localhost6.localdomain6
```

```
sysadmin@localhost:~$
```

Hay dos opciones que pueden utilizarse para asegurarnos contra sobreescrituras accidentales de los archivos. Con la opción `-i` (interactivo), el comando `cp` emitirá un prompt antes de sobrescribir un archivo. En el siguiente ejemplo demostraré esta opción, primero restaurando el contenido del archivo original:

```
sysadmin@localhost:~$ cp /etc/skel/.bash_logout ~/example.txt
```

```
sysadmin@localhost:~$ ls -l example.txt
```

```
-rw-r--r-- 1 sysadmin sysadmin 18 Sep 21 15:56 example.txt
```

```
sysadmin@localhost:~$ more example.txt
```

```
# ~/.bash_logout: executed by bash(1) when login shell exits.
```

```
sysadmin@localhost:~$ cp -i /etc/hosts ~/example.txt
```

```
cp: overwrite `/home/sysadmin/example.txt'? n
```

```
sysadmin@localhost:~$ ls -l example.txt
```

```
-rw-r--r-- 1 sysadmin sysadmin 18 Sep 21 15:56 example.txt
```

```
sysadmin@localhost:~$ more example.txt
```

```
# ~/.bash_logout: executed by bash(1) when login shell exits.
```

```
sysadmin@localhost:~$
```

Observa que puesto que el valor de `n` (no) se dió al emitir un prompt de sobrescritura del archivo, no se hicieron cambios en el archivo. Si se da un valor de `y` (sí), entonces resultará en un proceso de copiado.

La opción `-i` requiere respuesta `y` o `n` para cada copia que podría sobrescribir el contenido de un archivo existente. Esto puede ser tedioso cuando se sobrescribe un grupo, como se muestra en el siguiente ejemplo:

```
sysadmin@localhost:~$ cp -i /etc/skel/. * ~
cp: omitting directory `/etc/skel/.'
cp: omitting directory `/etc/skel/..'
cp: overwrite `/home/sysadmin/.bash_logout'? n
cp: overwrite `/home/sysadmin/.bashrc'? n
cp: overwrite `/home/sysadmin/.profile'? n
cp: overwrite `/home/sysadmin/.selected_editor'? n
sysadmin@localhost:~$
```

Como puedes ver en el ejemplo anterior, el comando `cp` intentó sobrescribir los cuatro archivos existentes, obligando al usuario a responder a tres prompts. Si esta situación ocurriera para 100 archivos, puede resultar muy molesto rápidamente.

Si quieres contestar automáticamente `n` para cada prompt, utiliza la opción `-n`. En esencia, significa «sin sobrescribir».

#### 6.4.3 Copiar los Directorios

En un ejemplo anterior se dieron mensajes de error cuando el comando `cp` intentó copiar los directorios:

```
sysadmin@localhost:~$ cp -i /etc/skel/. * ~
cp: omitting directory `/etc/skel/.'
cp: omitting directory `/etc/skel/..'
cp: overwrite `/home/sysadmin/.bash_logout'? n
cp: overwrite `/home/sysadmin/.bashrc'? n
cp: overwrite `/home/sysadmin/.profile'? n
cp: overwrite `/home/sysadmin/.selected_editor'? n
sysadmin@localhost:~$
```

Donde la salida dice `...omitting directory...` (o «omitiendo directorio» en español), el comando `cp` está diciendo que no puede copiar este artículo porque el comando no copia los directorios por defecto. Sin embargo, la opción `-r` del comando `cp` copiará ambos, los archivos y los directorios.

Ten cuidado con esta opción: se copiará la estructura completa del directorio. ¡Esto podría resultar en copiar muchos archivos y directorios!

#### 6.5 Mover los Archivos

Para mover un archivo, utiliza el comando `mv`. La sintaxis del comando `mv` es muy parecida al comando `cp`:

```
mv [fuente] [destino]
```

En el ejemplo siguiente, el archivo `hosts` que se generó anteriormente se mueve desde el directorio actual al directorio `Videos`:

```
sysadmin@localhost:~$ ls
Desktop  Downloads  Pictures  Templates  example.txt  hosts.copy
Documents Music      Public    Videos    hosts

sysadmin@localhost:~$ mv hosts Videos
sysadmin@localhost:~$ ls
Desktop  Downloads  Pictures  Templates  example.txt
Documents Music      Public    Videos    hosts.copy

sysadmin@localhost:~$ ls Videos
hosts

sysadmin@localhost:~$
```

Cuando se mueve un archivo, el archivo se elimina de la ubicación original y se coloca en una ubicación nueva. Esto puede ser algo complicado en Linux porque los usuarios necesitan permisos específicos para quitar archivos de un directorio. Si no tienes los permisos correctos, recibirás un mensaje de error «`Permission denied`» (o «Permiso denegado» en español):

```
sysadmin@localhost:~$ mv /etc/hosts .
mv: cannot move `/etc/hosts' to `./hosts': Permission denied
sysadmin@localhost:~$
```

En un capítulo posterior se ofrece una descripción detallada de los permisos.

### 6.6 Mover los Archivos Mientras se Cambia el Nombre

Si el destino del comando `mv` es un directorio, el archivo se moverá al directorio especificado. El nombre del archivo cambiará sólo si también se especifica un nombre de archivo destino.

Si no se especifica un directorio destino, el archivo será renombrado con el nombre de archivo destino y permanece en el directorio origen.

```
sysadmin@localhost:~$ ls
Desktop  Downloads  Pictures  Templates  example.txt
Documents Music      Public    Videos

sysadmin@localhost:~$ mv example.txt Videos/newexample.txt
```



```

sysadmin@localhost:~$ ls
Desktop  Downloads  Pictures  Templates
Documents  Music      Public    Videos

sysadmin@localhost:~$ ls Videos
hosts  newexample.txt

sysadmin@localhost:~$

```

### 6.6.1 Renombrar los Archivos

El comando `mv` no sólo se utiliza para mover un archivo, sino también cambiar el nombre de un archivo. Por ejemplo, los siguientes comandos cambiarán el nombre del archivo `newexample.txt` a `myexample.txt`:

```

sysadmin@localhost:~$ cd Videos
sysadmin@localhost:~/Videos$ ls
hosts  newexample.txt

sysadmin@localhost:~/Videos$ mv newexample.txt myexample.txt
sysadmin@localhost:~/Videos$ ls
hosts  myexample.txt

sysadmin@localhost:~/Videos$

```

Piensa en el ejemplo anterior del `mv` que significa «mover el archivo `newexample.txt` desde el directorio actual de regreso al directorio actual y denominar el nuevo archivo como `myexample.txt`».

### 6.6.2 Opciones Adicionales del `mv`

Igual al comando `cp`, el comando `mv` proporciona las siguientes opciones:

| Opción          | Significado   |
|-----------------|---|
| <code>-i</code> | Movimiento interactivo: pregunta si un archivo debe sobrescribirse. |
| <code>-n</code> | No sobrescribir el contenido de los archivos de destino             |
| <code>-v</code> | Verbose: muestra el movimiento resultante                           |

**Importante:** Aquí no hay ninguna opción `-r`, ya que el comando `mv` moverá los directorios de forma predeterminada.

## 6.7 Crear Archivos

Hay varias maneras de crear un nuevo archivo, incluyendo el uso de un programa diseñado para editar un archivo (un editor de texto). En un capítulo posterior, se cubrirán los editores de texto.

También existe una manera de simplemente crear un archivo que puede rellenarse con datos en un momento posterior. Esto es útil puesto que por algunas características del sistema operativo, la existencia de un archivo podría alterar la forma de funcionamiento de un comando o de un servicio. También es útil crear un archivo como un «indicador» («placeholder» en inglés) para recordarte que debes crear el contenido del archivo en un momento posterior.

Para crear un archivo vacío, utiliza el comando `touch` (o «tocar» en español) como se muestra a continuación:

```
sysadmin@localhost:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates
Videos

sysadmin@localhost:~$ touch sample
sysadmin@localhost:~$ ls -l sample
-rw-rw-r-- 1 sysadmin sysadmin 0 Nov  9 16:48 sample
sysadmin@localhost:~$
```

Fíjate que el tamaño del archivo nuevo es 0 bytes. Como ya se mencionó anteriormente, el comando `touch` no añade ningún dato en al archivo nuevo.

## 6.8 Eliminar los Archivos

Para borrar un archivo, utiliza el comando de `rm`:

```
sysadmin@localhost:~$ ls
Desktop  Downloads  Pictures  Templates  sample
Documents  Music      Public    Videos

sysadmin@localhost:~$ rm sample
sysadmin@localhost:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates
Videos

sysadmin@localhost:~$
```

Ten en cuenta que el archivo fue borrado sin hacer preguntas. Esto podría causar problemas al borrar varios archivos usando los caracteres glob, por ejemplo: `rm *.txt`. Ya que estos archivos son borrados sin proporcionar una pregunta, un usuario podría llegar a borrar archivos que no quería eliminar.

Además, los archivos se eliminan permanentemente. No hay ningún comando para recuperar un archivo y no hay «papelera de reciclaje» («trash can» en inglés) desde la que puedas recuperar los archivos eliminados. Como precaución, los usuarios deben utilizar la opción `-i` al eliminar varios archivos:

```

sysadmin@localhost:~$ touch sample.txt example.txt test.txt
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures   Templates  example.txt  test.txt
Documents  Music      Public     Videos     sample.txt
sysadmin@localhost:~$ rm -i *.txt
rm: remove regular empty file `example.txt'? y
rm: remove regular empty file `sample.txt'? n
rm: remove regular empty file `test.txt'? y
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures   Templates  sample.txt
Documents  Music      Public     Videos
sysadmin@localhost:~$

```

## 6.9 Eliminar los Directorios

Puedes borrar los directorios con el comando `rm`. Sin embargo, si utilizas el comando `rm` por defecto (sin opciones), éste no eliminará un directorio:

```

sysadmin@localhost:~$ rm Videos
rm: cannot remove `Videos': Is a directory
sysadmin@localhost:~$

```

Si quieres eliminar un directorio, utiliza la opción `-r` con el comando `rm`:

```

sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures   Templates  sample.txt
Documents  Music      Public     Videos
sysadmin@localhost:~$ rm -r Videos
sysadmin@localhost:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates
sample.txt

```

```
sysadmin@localhost:~$
```

**Importante:** Cuando un usuario elimina un directorio, todos los archivos y subdirectorios se eliminan sin proporcionar pregunta interactiva. Lo mejor es utilizar la opción `-i` con el comando `rm`.

También puedes borrar un directorio con el comando `rmdir`, pero sólo si el directorio está vacío.

## 6.10 Crear Directorios

Para crear un directorio, utiliza el comando `mkdir`:

```
sysadmin@localhost:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates
sample.txt

sysadmin@localhost:~$ mkdir test

sysadmin@localhost:~$ ls
Desktop  Downloads  Pictures  Templates  test
Documents  Music      Public    sample.txt

sysadmin@localhost:~$
```

## NDG Linux Essentials: Capítulo 7: Empaquetamiento y Compresión

### 7.1 Introducción

En este capítulo vamos a hablar de cómo **gestionar los archivos en la línea de comandos**.

El **Empaquetamiento de Archivos** se utiliza cuando uno o más archivos se tienen que transmitir o almacenar lo más eficientemente posible. Hay dos aspectos:

- **El Empaquetamiento** - Combina varios archivos en uno solo, lo que elimina la sobrecarga en archivos individuales y los hace más fácil de transmitir
- **Compresión** – hace los archivos más pequeños mediante la eliminación de información redundante

Puedes realizar **empaquetamiento** de **varios archivos en un solo archivo y luego comprimirlo, o puedes comprimir un archivo individual**. La **primera opción se conoce todavía como empaquetamiento de archivos**, mientras la **última se llama solo compresión**. Cuando tomas un archivo empaquetado, lo descomprimes y extraes uno o más archivos, lo estás *desempaquetando*.

A pesar de que el espacio en disco es relativamente barato, el empaquetamiento y la compresión aún tienen su valor:

- Si quieres que un gran número de archivos sea disponible, tales como el código fuente a una aplicación o un conjunto de documentos, es más fácil para las personas descargar un archivo empaquetado que descargar los archivos individuales.
- Los archivos de registro tienen la costumbre de llenar los discos por lo que es útil dividirlos por fecha y comprimir las versiones más antiguas.
- Cuando haces copias de seguridad de los directorios, es más fácil mantenerlos todos en un archivo empaquetado que crear una versión de cada archivo.
- Algunos dispositivos de transmisión, como cintas, se desempeñan mejor enviando una transmisión en secuencia de datos en lugar de los archivos individuales.
- A menudo **puede ser más rápido comprimir un archivo antes de enviarlo a una unidad** de cinta o a una red más lenta y **descomprimir en el otro extremo** en **vez de enviarlo descomprimido**.

Como administrador de Linux, debes familiarizarte con las herramientas para el empaquetamiento y compresión de archivos.

## 7.2 Comprimir los Archivos

*Comprimiendo* los archivos los hace más pequeños **eliminando la duplicación en un archivo y guardándolo de tal manera que el archivo se pueda restaurar**. Un archivo de texto legible podría reemplazar palabras usadas con frecuencia por algo más pequeño, o una imagen con un fondo sólido podría representar manchas de ese color por un código. Generalmente no usas la versión comprimida del archivo, más bien lo descomprimes antes de usar. **El algoritmo de compresión es un procedimiento con el que la computadora codifica el archivo original y como resultado lo hace más pequeño**. Los científicos informáticos investigan estos algoritmos y elaboran mejores, que pueden trabajar más rápido o hacer más pequeño el archivo de entrada.

Cuando se habla de compresión, existen dos tipos:

- **Lossless** (o «sin pérdida» en español): **No se elimina ninguna información del archivo**. Comprimir un archivo y descomprimirlo deja algo idéntico al original.
- **Lossy** (o «con pérdida» en español): **Información podría ser retirada del archivo cuando se comprime para que al descomprimir el archivo de lugar a un archivo que es un poco diferente que el original**. Por ejemplo, una imagen con dos tonos de verde sutilmente diferentes podría hacerse más pequeña por tratar esos dos tonos como uno. De todos modos, el ojo no puede reconocer la diferencia.

Generalmente los ojos y oídos humanos no notan imperfecciones leves en las imágenes y el audio, especialmente cuando se muestran en un monitor o suenan a través de los altavoces. La **compresión con pérdida a menudo beneficia a los medios digitales ya que los tamaños de los archivos son más pequeños y las personas no pueden indicar la diferencia entre el original y la versión con los datos cambiados**. Cosas que deben permanecer intactas, como los documentos, los registros y el software necesitan una compresión sin pérdida.

La mayoría de los **formatos** de imágenes, como GIF, PNG y JPEG, implementan algún tipo de **compresión con pérdida**. Generalmente puedes decidir cuánta calidad quieres conservar. Una calidad inferior resultará en un archivo más pequeño, pero después de la descompresión puedes notar resultados no deseados tales como los bordes ásperos o decoloraciones. Una calidad alta se parecerá mucho a la imagen original, pero el tamaño del archivo será más cercano al original.

Comprimir un archivo ya comprimido no lo hará más pequeño. Esto a menudo se olvida cuando se trata de imágenes, puesto que ya se almacenan en un formato comprimido. Con la compresión sin pérdida esta compresión múltiple no es un problema, **pero si se comprime y descomprime un archivo varias veces mediante un algoritmo con pérdida obtendrás algo que es irreconocible**.

**Linux proporciona varias herramientas para comprimir los archivos**, la más común es **gzip**. A continuación te mostraremos un archivo de registro antes y después de la compresión.

```
bob:tmp $ ls -l access_log*
-rw-r--r-- 1 sean sean 372063 Oct 11 21:24 access_log

bob:tmp $ gzip access_log
```

```
bob:tmp $ ls -l access_log*
-rw-r--r-- 1 sean sean 26080 Oct 11 21:24 access_log.gz
```

En el ejemplo anterior hay un archivo llamado `access_log` que tiene 372,063 bytes. El archivo se comprime invocando el comando `gzip` con el nombre del archivo como el único argumento. Al finalizar el comando su tarea, el archivo original desaparece y una versión comprimida con una extensión de archivo `.gz` se queda en su lugar. El tamaño del archivo es ahora 26,080 bytes, dando una relación de compresión de 14:1, que es común en el caso de los archivos de registro.

El comando `gzip` te dará esta información si se lo pides utilizando el parámetro `-l` tal como se muestra aquí:

```
bob:tmp $ gzip -l access_log.gz

compressed      uncompressed  ratio uncompressed_name
      26080           372063  93.0% access_log
```

Aquí puedes ver que se da el porcentaje de compresión de 93%, lo inverso de la relación 14:1, es decir, 13/14. Además, cuando el archivo se descomprime, se llamará `access_log`.

```
bob:tmp $ gunzip access_log.gz
bob:tmp $ ls -l access_log*
-rw-r--r-- 1 sean sean 372063 Oct 11 21:24 access_log
```

Lo contrario del comando `gzip` es el comando `gunzip`. Por otra parte, `gzip -d` hace la misma cosa (`gunzip` es sólo un script que invoca el comando `gzip` con los parámetros correctos). Cuando el comando `gunzip` termine su tarea, podrás ver que el archivo `access_log` vuelve a su tamaño original.

`Gzip` puede también actuar como un filtro que no lee ni escribe nada en el disco sino recibe datos a través de un canal de entrada y los escribe a un canal de salida. Aprenderás más sobre cómo funciona en el siguiente capítulo, por lo que el ejemplo siguiente sólo te da una idea de lo que puedes hacer al comprimir una secuencia.

```
bob:tmp $ mysqldump -A | gzip > database_backup.gz
bob:tmp $ gzip -l database_backup.gz

compressed      uncompressed  ratio uncompressed_name
      76866           1028003  92.5% database_backup
```

El comando `mysqldump -A` da salidas a los contenidos de las bases de datos de MySQL locales a la consola. El carácter `|` (barra vertical) dice "redirigir la salida del comando anterior en la entrada del siguiente". El programa para recibir la salida es `gzip`, que reconoce que no se dieron nombres de archivo por lo que debe funcionar en modo de barra vertical. Por último, `> database_backup.gz` significa "redirigir la salida del comando anterior en un archivo llamado `database_backup.gz`. La inspección de este archivo con `gzip -l` muestra que la versión comprimida es un 7.5% del tamaño original, con la ventaja agregada de que el archivo más grande jamás tuvo que ser escrito a disco.

Hay otro par de comandos que operan prácticamente idénticamente al `gzip` y `gunzip`. Éstos son el `bzip2` y `bunzip2`. Las utilidades del `bzip` utilizan un algoritmo de compresión diferente (llamado bloque de clasificación de Burrows-Wheeler frente a la codificación Lempel-Ziv que utiliza `gzip`) que puede comprimir los archivos más pequeños que un `gzip` a costa de más tiempo de CPU. Puedes reconocer estos archivos porque tienen una extensión `.bz` o `.bz2` en vez de `.gz`.

### 7.3 Empaquetando Archivos

Si quieres enviar varios archivos a alguien, podrías comprimir cada uno individualmente. Tendrías una cantidad más pequeña de datos en total que si enviaras los archivos sin comprimir, pero todavía tendrías que lidiar con muchos archivos al mismo tiempo.

El **empacamiento de archivos es la solución a este problema**. La utilidad tradicional de UNIX para archivar los ficheros es la llamada **tar**, que es una abreviación de TApe aRchive (o «archivo de cinta» en español). **Tar era utilizado para transmitir muchos archivos a una cinta para copias de seguridad o transferencias de archivos**. Tar toma varios archivos y crea un único archivo de salida que se puede dividir otra vez en los archivos originales en el otro extremo de la transmisión.

Tar tiene 3 modos que deberás conocer:

- **Crear:** hacer un archivo nuevo de una serie de archivos
- **Extraer:** sacar uno o más archivos de un archivo
- **Listar:** mostrar el contenido del archivo sin extraer

Recordar los modos es clave para averiguar las opciones de la línea de comandos necesarias para hacer lo que quieres. Además del modo, querrás asegurarte de que recuerdas dónde especificar el nombre del archivo, ya que podrías estar introduciendo varios nombres de archivo en una línea de comandos.

Aquí, mostramos un *archivo tar*, también llamado un tarball, siendo creado a partir de múltiples registros de acceso.

```
bob:tmp $ tar -cf access_logs.tar access_log*
bob:tmp $ ls -l access_logs.tar
-rw-rw-r-- 1 sean sean 542720 Oct 12 21:42 access_logs.tar
```

La creación de un archivo requiere dos opciones de nombre. La primera, **c**, **especifica el modo**. La segunda, **f**, le dice a **tar** **que espere un nombre de archivo como el siguiente argumento**. El primer argumento en el ejemplo anterior crea un archivo llamado `access_logs.tar`. El resto de los argumentos se toman para ser nombres de los archivo de entrada, ya sea un comodín, una lista de archivos o ambos. En este ejemplo, utilizamos la opción comodín para incluir todos los archivos que comienzan con `access_log`.

El ejemplo anterior hace un listado largo de directorio del archivo creado. El tamaño final es 542,720 bytes que es ligeramente más grande que los archivos de entrada. Los tarballs pueden ser comprimidos para transporte más fácil, ya sea comprimiendo un archivo con **gzip** o diciéndole a **tar** que lo haga con la opción **z** tal como se muestra a continuación:

```
bob:tmp $ tar -czf access_logs.tar.gz access_log*
bob:tmp $ ls -l access_logs.tar.gz
-rw-rw-r-- 1 sean sean 46229 Oct 12 21:50 access_logs.tar.gz
bob:tmp $ gzip -l access_logs.tar.gz
```

| compressed | uncompressed | ratio | uncompressed_name |
|------------|--------------|-------|-------------------|
| 46229      | 542720       | 91.5% | access_logs.tar   |

El ejemplo anterior muestra el mismo comando como en el ejemplo anterior, pero con la adición del parámetro `z`. La salida es mucho menor que el tarball en sí mismo, y el archivo resultante es compatible con `gzip`. Se puede ver en el último comando que el archivo descomprimido es del mismo tamaño, como si hubieras utilizado el `tar` en un paso separado.

Mientras que UNIX no trata las extensiones de archivo especialmente, la convención es usar `.tar` para los archivos tar y `.tar.gz` o `.tgz` para los archivos tar comprimidos. Puedes utilizar `bzip2` en vez de `gzip` sustituyendo la letra `z` con `j` y usando `.tar.bz2`, `.tbz`, o `.tbz2` para una extensión de archivo (por ejemplo `tar -cjf file.tbz access_log*`).

En el archivo `tar`, comprimido o no, puedes ver lo que hay dentro utilizando el comando `t`:

```
bob:tmp $ tar -tjf access_logs.tbz
logs/
logs/access_log.3
logs/access_log.1
logs/access_log.4
logs/access_log
logs/access_log.2
```

Este ejemplo utiliza 3 opciones:

- `t`: listar documentos en el archivo en el archivo empaquetado
- `j`: descomprimir con `bzip2` antes de la lectura
- `f`: operar en el nombre de archivo `access_logs.tbz`

El contenido del archivo comprimido entonces es desplegado. Puedes ver que un directorio fue prefijado a los archivos. Tar se efectuará de manera recursiva hacia los subdirectorios automáticamente cuando comprime y almacenará la información de la ruta de acceso dentro del archivo.

Sólo para mostrar que este archivo aún no es nada especial, vamos a listar el contenido del archivo en dos pasos mediante una barra vertical.

```
bob:tmp $ bunzip2 -c access_logs.tbz | tar -t
logs/
logs/access_log.3
logs/access_log.1
logs/access_log.4
logs/access_log
logs/access_log.2
```



A la izquierda de la barra vertical está `bunzip -c access_logs.tbz`, que descomprime el archivo, pero el `c` envía la salida a la pantalla. La salida es redirigida a `tar -t`. Si no especificas que un archivo con `-f`, entonces el `tar` leerá la entrada estándar, que en este caso es el archivo sin comprimir.

Finalmente, puedes extraer el archivo con la marca `-x`:

```
bob:tmp $ tar -xjf access_logs.tbz
bob:tmp $ ls -l
total 36
-rw-rw-r-- 1 sean sean 30043 Oct 14 13:27 access_logs.tbz
drwxrwxr-x 2 sean sean 4096 Oct 14 13:26 logs
bob:tmp $ ls -l logs
total 536
-rw-r--r-- 1 sean sean 372063 Oct 11 21:24 access_log
-rw-r--r-- 1 sean sean 362 Oct 12 21:41 access_log.1
-rw-r--r-- 1 sean sean 153813 Oct 12 21:41 access_log.2
-rw-r--r-- 1 sean sean 1136 Oct 12 21:41 access_log.3
-rw-r--r-- 1 sean sean 784 Oct 12 21:41 access_log.4
```

El ejemplo anterior utiliza un patrón similar como antes, especificando la operación (eXtract), compresión ( (la opción `j`, que significa `bzip2`) y un nombre de archivo `-f access_logs.tbz`). El archivo original está intacto y se crea el nuevo directorio `logs`. Los archivos están dentro del directorio.

Añade la opción `-v` y obtendrás una salida detallada de los archivos procesados. Esto es útil para que puedas ver lo que está sucediendo:

```
bob:tmp $ tar -xjvf access_logs.tbz
logs/
logs/access_log.3
logs/access_log.1
logs/access_log.4
logs/access_log
logs/access_log.2
```

Es importante mantener la opción `-f` al final, ya que el `tar` asume que lo que sigue es un nombre de archivo. En el siguiente ejemplo, las opciones `f` y `v` fueron transpuestas, llevando al `tar` a interpretar el comando como una operación en un archivo llamado `"v"` (el mensaje relevante viene en *itálica*).

```
bob:tmp $ tar -xjfv access_logs.tbz
tar (child): v: Cannot open: No such file or directory
tar (child): Error is not recoverable: exiting now
tar: Child returned status 2
tar: Error is not recoverable: exiting now
```

Si sólo quieres algunos documentos del archivo empaquetado puedes agregar sus nombres al final del comando, pero por defecto deben coincidir exactamente con el nombre del archivo o puedes utilizar un patrón:

```
bob:tmp $ tar -xjvf access_logs.tbz logs/access_log
logs/access_log
```

El ejemplo anterior muestra el mismo archivo que antes, pero extrayendo solamente el archivo `logs/access_log`. La salida del comando (ya se solicitó el modo detallado con la bandera `v`) muestra que sólo un archivo se ha extraído.

El `tar` tiene muchas más funciones, como la capacidad de utilizar los patrones al extraer los archivos, excluir ciertos archivos o mostrar los archivos extraídos en la pantalla en lugar de un disco. La documentación para el `tar` contiene información a profundidad.

#### 7.4 Archivos ZIP

De hecho, la utilidad del empaquetamiento de archivos en el mundo de Microsoft es el archivo ZIP. No es tan frecuente en Linux pero también es compatible con los comandos `zip` y `unzip`. Con el `tar` y `gzip/gunzip` se pueden utilizar los mismos comandos y las mismas opciones para hacer la creación y extracción, pero éste no es el caso del `zip`. La misma opción tiene diferentes significados para los estos dos diferentes comandos.

El modo predeterminado del `zip` es añadir documentos a un archivo y comprimir.

```
bob:tmp $ zip logs.zip logs/*
adding: logs/access_log (deflated 93%)
adding: logs/access_log.1 (deflated 62%)
adding: logs/access_log.2 (deflated 88%)
adding: logs/access_log.3 (deflated 73%)
adding: logs/access_log.4 (deflated 72%)
```

El primer argumento en el ejemplo anterior es el nombre del archivo sobre el cual se trabajará, en este caso es el `logs.zip`. Después de eso, hay que añadir una lista de archivos a ser agregados. La salida muestra los archivos y la relación de compresión. Debes notar que el `tar` requiere la opción `-f` para indicar que se está pasando un nombre de archivo, mientras que el `zip` y `unzip` requiere un nombre de archivo, por lo tanto no tienes que decir explícitamente que se está pasando un nombre de archivo.

`zip` no se efectuará de manera recursiva hacia los subdirectorios por defecto, lo que es un comportamiento diferente del `tar`. Es decir, simplemente añadiendo logs en vez de `logs/*` sólo añadirá un directorio vacío y no los archivos dentro de él. Si quieres que `zip` se comporte de manera parecida, debes utilizar el comando `-r` para indicar que se debe usar la recursividad:

```
bob:tmp $ zip -r logs.zip logs
adding: logs/ (stored 0%)
adding: logs/access_log.3 (deflated 73%)
adding: logs/access_log.1 (deflated 62%)
adding: logs/access_log.4 (deflated 72%)
adding: logs/access_log (deflated 93%)
adding: logs/access_log.2 (deflated 88%)
```

En el ejemplo anterior, se añaden todos los archivos bajo el directorio `logs` ya que utiliza la opción `-r`. La primera línea de la salida indica que un directorio se agregó al archivo, pero de lo contrario la salida es similar al ejemplo anterior.

El listado de los archivos en el `zip` se realiza por el comando `unzip` y la opción `-l` (listar):

```
bob:tmp $ unzip -l logs.zip
Archive:  logs.zip

  Length      Date    Time    Name
-----
         0  10-14-2013  14:07    logs/
      1136  10-14-2013  14:07  logs/access_log.3
       362  10-14-2013  14:07  logs/access_log.1
       784  10-14-2013  14:07  logs/access_log.4
      90703  10-14-2013  14:07  logs/access_log
     153813  10-14-2013  14:07  logs/access_log.2
-----
      246798                      6 files
```

Extraer los archivos es como crear el archivo, ya que la operación predeterminada es extraer:

```
bob:tmp $ unzip logs.zip
Archive:  logs.zip
creating: logs/
```

```

inflating: logs/access_log.3
inflating: logs/access_log.1
inflating: logs/access_log.4
inflating: logs/access_log
inflating: logs/access_log.2

```

Aquí, extraemos todos los documentos del archivo empaquetado al directorio actual. Al igual que el `tar`, puedes pasar los nombres de archivos a la línea de comandos:

```

bob:tmp $ unzip logs.zip access_log
Archive:  logs.zip
caution: filename not matched:  access_log
bob:tmp $ unzip logs.zip logs/access_log
Archive:  logs.zip
    inflating: logs/access_log
bob:tmp $ unzip logs.zip logs/access_log.*
Archive:  logs.zip
    inflating: logs/access_log.3
    inflating: logs/access_log.1
    inflating: logs/access_log.4
    inflating: logs/access_log.2

```

El ejemplo anterior muestra tres diferentes intentos para extraer un archivo. En primer lugar, se pasa sólo el nombre del archivo sin el componente del directorio. Al igual que con el `tar`, el archivo no coincide.

El segundo intento pasa el componente del directorio junto con el nombre del archivo, que extrae solo ese archivo.

La tercera versión utiliza un comodín, que extrae los 4 archivos que coinciden con el patrón, al igual que el `tar`.

Las páginas man del `zip` y `unzip` describen las otras cosas que puedes hacer con estas herramientas, tales como reemplazar los archivos dentro del archivo empaquetado, utilizar los diferentes niveles de compresión o incluso el cifrado.

**NDG Linux Essentials: Capítulo 8: Las Barras Verticales, Redirección y Las Expresiones Regulares**

## 8.1 Introducción

Un gran número de los archivos en un típico sistema de archivos son *archivos de texto*. Los archivos de texto contienen sólo texto, sin características de formato que puedas ver en un archivo de procesamiento de texto.

Ya que hay muchos de estos archivos en un sistema Linux típico, existe un gran número de comandos para ayudar a los usuarios manipular los archivos de texto. Hay comandos para ver y modificar estos archivos de varias maneras.

Además, existen características disponibles para el shell para controlar la salida de los comandos, así que en lugar de tener la salida en la ventana de la terminal, la salida se puede redirigir a otro archivo u otro comando. Estas características de la redirección ofrecen a los usuarios un entorno más flexible y potente para trabajar.

## 8.2 Las Barras Verticales en la Línea de Comandos

Capítulos anteriores describen la manera de usar los comandos individuales para realizar las acciones en el sistema operativo, incluyendo cómo crear/mover/eliminar los archivos y moverse en todo el sistema. Por lo general, cuando un comando ofrece salida o genera un error, la salida se muestra en la pantalla; sin embargo, esto no tiene que ser el caso.

El carácter barra vertical `|` (o «pipe» en inglés) puede utilizarse para enviar la salida de un comando a otro. En lugar de que se imprima en la pantalla, la salida de un comando se convierte en una entrada para el siguiente comando. Esto puede ser una herramienta poderosa, especialmente en la búsqueda de datos específicos; la implementación de la barra vertical (o «piping» en inglés) se utiliza a menudo para refinar los resultados de un comando inicial.

Los comandos `head` (o «cabeza» en español) y `tail` (o «cola» en español) se utilizarán en muchos ejemplos a continuación para ilustrar el uso de las barras verticales. Estos comandos se pueden utilizar para mostrar solamente algunas de las primeras o las últimas líneas de un archivo (o, cuando se utiliza con una barra vertical, la salida de un comando anterior).

Por defecto, los comandos `head` y `tail` mostrarán diez líneas. Por ejemplo, el siguiente comando muestra las diez primeras líneas del archivo `/etc/sysctl.conf`:

```
sysadmin@localhost:~$ head /etc/sysctl.conf
#
# /etc/sysctl.conf - Configuration file for setting system variables
# See /etc/sysctl.d/ for additional system variables
# See sysctl.conf (5) for information.
#

#kernel.domainname = example.com

# Uncomment the following to stop low-level messages on console
#kernel.printk = 3 4 1 3
sysadmin@localhost:~$
```

En el ejemplo siguiente, se mostrarán las últimas diez líneas del archivo:

```
sysadmin@localhost:~$ tail /etc/sysctl.conf
```

```
# Do not send ICMP redirects (we are not a router)
#net.ipv4.conf.all.send_redirects = 0
#
# Do not accept IP source route packets (we are not a router)
#net.ipv4.conf.all.accept_source_route = 0
#net.ipv6.conf.all.accept_source_route = 0
#
# Log Martian Packets
#net.ipv4.conf.all.log_martians = 1
#
sysadmin@localhost:~$
```

El carácter de la barra vertical permite a los usuarios utilizar estos comandos no sólo en los archivos, sino también en la salida de otros comandos. Esto puede ser útil al listar un directorio grande, por ejemplo el directorio /etc:

|                      |                |               |             |
|----------------------|----------------|---------------|-------------|
| ca-certificates      | insserv        | nanorc        | services    |
| ca-certificates.conf | insserv.conf   | network       | sgml        |
| calendar             | insserv.conf.d | networks      | shadow      |
| cron.d               | iproute2       | nologin       | shadow-     |
| cron.daily           | issue          | nsswitch.conf | shells      |
| cron.hourly          | issue.net      | opt           | skel        |
| cron.monthly         | kernel         | os-release    | ssh         |
| cron.weekly          | ld.so.cache    | pam.conf      | ssl         |
| crontab              | ld.so.conf     | pam.d         | sudoers     |
| dbus-1               | ld.so.conf.d   | passwd        | sudoers.d   |
| debconf.conf         | ldap           | passwd-       | sysctl.conf |
| debian_version       | legal          | perl          | sysctl.d    |
| default              | locale.alias   | pinforc       | systemd     |
| deluser.conf         | localtime      | ppp           | terminfo    |
| depmod.d             | logcheck       | profile       | timezone    |

```

dpkg          login.defs      profile.d     ucf.conf
environment   logrotate.conf  protocols    udev
fstab          logrotate.d     python2.7    ufw
fstab.d        lsbase          rc.local     update-motd.d
gai.conf       lsbase-logging.sh rc0.d         updatedb.conf
groff          lsbase-release  rc1.d         vim
group          magic           rc2.d         wgetrc
group-         magic.mime      rc3.d         xml

sysadmin@localhost:~$

```

Si te fijas en la salida del comando anterior, notarás que ese primer nombre del archivo es `ca-certificates`. Pero hay otros archivos listados "arriba" que sólo pueden verse si el usuario utiliza la barra de desplazamiento. ¿Qué pasa si sólo quieres listas algunos de los primeros archivos del directorio `/etc`?

En lugar de mostrar la salida del comando anterior, poner la barra vertical junto al comando `head` muestra sólo las primeras diez líneas:

```

sysadmin@localhost:~$ ls /etc | head
adduser.conf
adjtime
alternatives
apparmor.d
apt
bash.bashrc
bash_completion.d
bind
bindresvport.blacklist
blkid.conf

sysadmin@localhost:~$

```

La salida del comando `ls` se pasa al comando `head` por el shell en vez de ser impreso a la pantalla. El comando `head` toma esta salida (del `ls`) como "datos de entrada" y luego se imprime la salida del `head` a la pantalla.

Múltiples barras verticales pueden utilizarse consecutivamente para unir varios comandos. Si se unen tres comandos con la barra vertical, la salida del primer comando se pasa al segundo comando. La salida del segundo comando se pasa al tercer comando. La salida del tercer comando se imprime en la pantalla.

Es importante elegir cuidadosamente el orden en que los comandos están unidos con la barra vertical, ya que el tercer comando sólo verá como entrada, la salida del segundo comando. Los siguientes ejemplos ilustran esta situación usando el comando `nl`. En el primer ejemplo, el comando de `nl` se utiliza para numerar las líneas de la salida de un comando anterior:

```
sysadmin@localhost:~$ ls -l /etc/ppp | nl                      1  total 44
 2  -rw----- 1 root root   78 Aug 22  2010 chap-secrets
 3  -rwxr-xr-x 1 root root  386 Apr 27  2012 ip-down
 4  -rwxr-xr-x 1 root root 3262 Apr 27  2012 ip-down.ipv6to4
 5  -rwxr-xr-x 1 root root  430 Apr 27  2012 ip-up
 6  -rwxr-xr-x 1 root root 6517 Apr 27  2012 ip-up.ipv6to4
 7  -rwxr-xr-x 1 root root 1687 Apr 27  2012 ipv6-down
 8  -rwxr-xr-x 1 root root 3196 Apr 27  2012 ipv6-up
 9  -rw-r--r-- 1 root root    5 Aug 22  2010 options
10  -rw----- 1 root root   77 Aug 22  2010 pap-secrets
11  drwxr-xr-x 2 root root 4096 Jun 22  2012 peers
sysadmin@localhost:~$
```

En el ejemplo siguiente, observa que el comando `ls` es ejecutado primero y su salida se envía al comando `nl`, enumerando todas las líneas de la salida del comando `ls`. A continuación, se ejecuta el comando `tail`, mostrando las últimas cinco líneas de la salida del comando `nl`:

```
sysadmin@localhost:~$ ls -l /etc/ppp | nl | tail -5
 7  -rwxr-xr-x 1 root root 1687 Apr 27  2012 ipv6-down
 8  -rwxr-xr-x 1 root root 3196 Apr 27  2012 ipv6-up
 9  -rw-r--r-- 1 root root    5 Aug 22  2010 options
10  -rw----- 1 root root   77 Aug 22  2010 pap-secrets
11  drwxr-xr-x 2 root root 4096 Jun 22  2012 peers
sysadmin@localhost:~$
```

Compara la salida anterior con el siguiente ejemplo:

```
sysadmin@localhost:~$ ls -l /etc/ppp | tail -5 | nl
 1  -rwxr-xr-x 1 root root 1687 Apr 27  2012 ipv6-down
 2  -rwxr-xr-x 1 root root 3196 Apr 27  2012 ipv6-up
```



```

3  -rw-r--r-- 1 root root    5 Aug 22  2010 options
4  -rw----- 1 root root   77 Aug 22  2010 pap-secrets
5  drwxr-xr-x 2 root root 4096 Jun 22  2012 peers

sysadmin@localhost:~$

```

Observa los diferentes números de línea. ¿Por qué sucede esto?

En el segundo ejemplo, la salida del comando `ls` se envía primero al comando `tail` que "capta" sólo las últimas cinco líneas de la salida. El comando `tail` envía esas cinco líneas para al comando `nl`, que los enumera del 1 al 5.

Las barras verticales pueden ser poderosas, pero es importante considerar cómo se unen los comandos con ellas para asegurar que se muestre la salida deseada.

### 8.3 Redirección de E/S

La Redirección de Entrada/Salida (E/S) permite que la información pase de la línea de comandos a las diferentes *secuencias*. Antes de hablar sobre la redirección, es importante entender las secuencias estándar.

#### 8.3.1 STDIN

La entrada estándar STDIN es la información normalmente introducida por el usuario mediante el teclado. Cuando un comando envía un prompt al shell esperando datos, el shell proporciona al usuario la capacidad de introducir los comandos, que a su vez, se envían al comando como STDIN.

#### 8.3.2 STDOUT

Salida estándar o STDOUT es la salida normal de los comandos. Cuando un comando funciona correctamente (sin errores), la salida que produce se llama STDOUT. De forma predeterminada, STDOUT se muestra en la ventana de la terminal (pantalla) donde se ejecuta el comando.

#### 8.3.3 STDERR

Error estándar o STDERR son mensajes de error generados por los comandos. De forma predeterminada, STDERR se muestra en la ventana de la terminal (pantalla) donde se ejecuta el comando.

La redirección de E/S permite al usuario redirigir STDIN para que los datos provengan de un archivo y la salida de STDOUT/STDERR vaya a un archivo. La redirección se logra mediante el uso de los caracteres de la flecha: `<` y `>`.

#### 8.3.4 Redirigir STDOUT

STDOUT se puede dirigir a los archivos. Para empezar, observa la salida del siguiente comando que se mostrará en la pantalla:

```

sysadmin@localhost:~$ echo "Line 1"

Line 1

sysadmin@localhost:~$

```

Utilizando el carácter `>`, la salida puede ser redirigida a un archivo:

```

sysadmin@localhost:~$ echo "Line 1" > example.txt

sysadmin@localhost:~$ ls

Desktop    Downloads  Pictures   Templates  example.txt  test
Documents  Music      Public     Videos    sample.txt

```

```
sysadmin@localhost:~$ cat example.txt
Line 1
sysadmin@localhost:~$
```

Este comando no muestra ninguna salida, porque la STDOUT fue enviada al archivo `example.txt` en lugar de la pantalla. Puedes ver el nuevo archivo con la salida del comando `ls`. El archivo recién creado contiene la salida del comando `echo` cuando se ve el contenido del archivo con el comando `cat`.

Es importante tener en cuenta que la flecha sola sobrescribe cualquier contenido de un archivo existente:

```
sysadmin@localhost:~$ cat example.txt
Line 1
sysadmin@localhost:~$ echo "New line 1" > example.txt
sysadmin@localhost:~$ cat example.txt
New line 1
sysadmin@localhost:~$
```

El contenido original del archivo ha desaparecido y fue reemplazado por la salida del comando `echo` nuevo.

También es posible preservar el contenido de un archivo existente anexando al mismo. Utiliza la «doble flecha» `>>` para anexar a un archivo en vez de sobrescribirlo:

```
sysadmin@localhost:~$ cat example.txt
New line 1
sysadmin@localhost:~$ echo "Another line" >> example.txt
sysadmin@localhost:~$ cat example.txt
New line 1
Another line
sysadmin@localhost:~$
```

En lugar de ser sobrescrito, la salida del comando del comando `echo` reciente se anexa a la parte inferior del archivo.

### 8.3.5 Redirigir la STDERR

Puedes redirigir el STDERR de una manera similar a la STDOUT. STDOUT es también conocida como secuencia o canal («stream» o «channel» en inglés) #1. Al STDERR se asigna la secuencia #2.

Al utilizar las flechas para redirigir, se asumirá la secuencia #1 mientras no venga especificada otra secuencia. Por lo tanto, la secuencia #2 debe especificarse al redirigir el STDERR.

Para demostrar la redirección del STDERR, primero observa el siguiente comando que producirá un error porque el directorio especificado no existe:

```
sysadmin@localhost:~$ ls /fake
ls: cannot access /fake: No such file or directory
```

```
sysadmin@localhost:~$
```

Ten en cuenta que no hay nada en el ejemplo anterior lo que implica que la salida es STDERR. La salida es claramente un mensaje de error, pero ¿cómo podrías saber que se envía al STDERR? Una manera fácil de determinar esto es redirigir al STDOUT:

```
sysadmin@localhost:~$ ls /fake > output.txt
ls: cannot access /fake: No such file or directory
sysadmin@localhost:~$
```

En el ejemplo anterior, el STDOUT fue redirigido al archivo `output.txt`. Por lo tanto, **la salida que se muestra no puede ser STDOUT porque habría quedado en el archivo `output.txt`**. Ya que todos los resultados del comando van a STDOUT o STDERR, la salida mostrada debe ser STDERR.

El STDERR de un comando puede enviarse a un archivo:

```
sysadmin@localhost:~$ ls /fake 2> error.txt
sysadmin@localhost:~$ cat error.txt
ls: cannot access /fake: No such file or directory
sysadmin@localhost:~$
```

En el comando de arriba, `2>` indica que todos los mensajes de error deben enviarse al archivo `error.txt`.

### 8.3.6 Redireccionando Múltiples Secuencias

Es posible dirigir la salida STDOUT y STDERR de un comando a la vez. El siguiente comando produce ambas salidas STDOUT y STDERR porque existe uno de los directorios especificados y el otro no:

```
sysadmin@localhost:~$ ls /fake /etc/ppp
ls: cannot access /fake: No such file or directory
/etc/ppp:
chap-secrets  ip-down  ip-down.ipv6to4  ip-up      ip-up.ipv6to4
ipv6-down    ipv6-up  options          pap-secrets peers
```

Si sólo se envía la salida STDOUT a un archivo, la STDERR todavía se imprimirá a la pantalla:

```
sysadmin@localhost:~$ ls /fake /etc/ppp > example.txt
ls: cannot access /fake: No such file or directory
sysadmin@localhost:~$ cat example.txt
/etc/ppp:
chap-secrets
```

```

ip-down
ip-down.ipv6to4
ip-up
ip-up.ipv6to4
ipv6-down
ipv6-up
options
pap-secrets
peers
sysadmin@localhost:~$

```

Si sólo se envía la salida STDERR a un archivo, la STDOUT todavía se imprimirá a la pantalla:

```

sysadmin@localhost:~$ ls /fake /etc/ppp 2> error.txt
/etc/ppp:
hap-secrets      ip-down    ip-down.ipv6to4  ip-up      ip-up.ipv6to4
ipv6-down        ipv6-up    options          pap-secrets peers
sysadmin@localhost:~$ cat error.txt
ls: cannot access /fake: No such file or directory
sysadmin@localhost:~$

```

Las salidas STDOUT y STDERR pueden enviarse a un archivo mediante el uso de &> un conjunto de caracteres que significan «ambos 1> y 2>»:

```

sysadmin@localhost:~$ ls /fake /etc/ppp &> all.txt
sysadmin@localhost:~$ cat all.txt
ls: cannot access /fake: No such file or directory
/etc/ppp:
chap-secrets
ip-down
ip-down.ipv6to4
ip-up
ip-up.ipv6to4

```

```

ipv6-down
ipv6-up
options
pap-secrets
peers
sysadmin@localhost:~$

```

Ten en cuenta que cuando se utiliza `&>`, la salida aparece en el archivo con todos los mensajes STDERR en la parte superior y todos los mensajes STDOUT debajo de todos los mensajes de STDERR:

```

sysadmin@localhost:~$ ls /fake /etc/ppp /junk /etc/sound &> all.txt
sysadmin@localhost:~$ cat all.txt
ls: cannot access /fake: No such file or directory
ls: cannot access /junk: No such file or directory
/etc/ppp:
chap-secrets
ip-down
ip-down.ipv6to4
ip-up
ip-up.ipv6to4
ipv6-down
ipv6-up
options
pap-secrets
peers

/etc/sound:
events
sysadmin@localhost:~$

```

Si no quieres que las salidas STDERR y STDOUT vayan al mismo archivo, puede redirigirlas a diferentes archivos utilizando `>` and `2>`. Por ejemplo:

```

sysadmin@localhost:~$ rm error.txt example.txt
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures   Templates  all.txt
Documents  Music      Public     Videos
sysadmin@localhost:~$ ls /fake /etc/ppp > example.txt 2> error.txt
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures   Templates  all.txt    example.txt
Documents  Music      Public     Videos    error.txt
sysadmin@localhost:~$ cat error.txt
ls: cannot access /fake: No such file or directory
sysadmin@localhost:~$ cat example.txt
/etc/ppp:
chap-secrets
ip-down
ip-down.ipv6to4
ip-up
ip-up.ipv6to4
ipv6-down
ipv6-up
options
pap-secrets
peers
sysadmin@localhost:~$

```

**No importa el orden** en el vienen las secuencias especificadas.

### 8.3.7 Redirigir la entrada *STDIN*

El concepto de redireccionar la *STDIN* es difícil, ya que es más difícil de entender el por qué querrías redirigir la *STDIN*. Con las salidas *STDOUT* y *STDERR*, la respuesta del por qué es bastante fácil: porque a veces quieres almacenar el resultado en un archivo para su uso futuro.

La mayoría de los usuarios de Linux terminan redirigiendo rutinariamente la STDOUT, en ocasiones la STDERR y la STDIN... bien, muy raramente. Hay muy pocos comandos que requieren de que redirijas la STDIN porque en el caso de la mayoría de los comandos, si quieres pasar los datos desde un archivo a un comando, simplemente puedes especificar el nombre del archivo como un argumento del comando. Después, el comando buscará en el archivo.

En el caso de algunos comandos, si no se especifica un nombre de archivo como argumento, volverán a usar la salida STDIN para obtener los datos. Por ejemplo, considera el siguiente comando `cat`:

```
sysadmin@localhost:~$ cat
hello
hello
how are you?
how are you?
goodbye
goodbye
sysadmin@localhost:~$
```

En el ejemplo anterior, el comando `cat` no recibió el nombre de archivo como argumento. Por lo tanto, pidió los datos a mostrarlos en la pantalla desde la entrada STDIN. El usuario introduce `hello` y luego el comando `cat` muestra `hello` en la pantalla. Tal vez esto es útil para las personas solitarias, pero no es realmente un buen uso del comando `cat`.

Sin embargo, tal vez si la salida del comando `cat` se redirige a un archivo, entonces este método podría utilizarse para agregar datos a un archivo existente o colocar texto en un archivo nuevo:

```
sysadmin@localhost:~$ cat > new.txt
Hello
How are you?
Goodbye
sysadmin@localhost:~$ cat new.txt
Hello
How are you?
Goodbye
sysadmin@localhost:~$
```

Mientras que en el ejemplo anterior se muestra otra de las ventajas de redireccionar la STDOUT, no aborda el por qué o cómo puedes dirigir la STDIN. Para entender esto, consideremos primero un nuevo comando llamado `tr`. Este comando tomará un conjunto de caracteres y los plasmará en otro conjunto de caracteres.

Por ejemplo, supongamos que quieres poner una línea de comandos en mayúsculas. Puede utilizar el comando `tr` de la siguiente manera:

```

sysadmin@localhost:~$ tr 'a-z' 'A-Z'
watch how this works
WATCH HOW THIS WORKS
sysadmin@localhost:~$

```

El comando `tr` tomó la entrada STDIN desde el teclado (`watch how this works`) (*a ver cómo funciona esto*) y convierte todas las letras en minúsculas antes de enviar la salida STDOUT a la pantalla (`WATCH HOW THIS WORKS`).

Parecería que el comando `tr` **serviera más para realizar la traducción en un archivo**, no la entrada del teclado. Sin embargo, el comando `tr` no admite argumentos del nombre de archivo:

```

sysadmin@localhost:~$ more example.txt
/etc/ppp:
chap-secrets
ip-down
ip-down.ipv6to4
ip-up
ip-up.ipv6to4
ipv6-down
ipv6-up
options
pap-secrets
peers
sysadmin@localhost:~$ tr 'a-z' 'A-Z' example.txt
tr: extra operand `example.txt'
Try `tr --help' for more information
sysadmin@localhost:~$

```

Sin embargo, puedes decirle al shell que obtenga la STDIN de un archivo en vez de desde el teclado mediante el uso del carácter `<`:

```

sysadmin@localhost:~$ tr 'a-z' 'A-Z' < example.txt
/ETC/PPP:
CHAP-SECRETS

```



```

IP-DOWN
IP-DOWN.IPV6TO4
IP-UP
IP-UP.IPV6TO4
IPV6-DOWN
IPV6-UP
OPTIONS
PAP-SECRETS
sysadmin@localhost:~$

```

Esto es bastante raro porque la mayoría de los comandos aceptan a los nombres de archivo como argumentos. Sin embargo, para los que no, este método podría utilizarse para que el shell lea desde el archivo en lugar de confiar en el comando que tienen esta capacidad.

Una última nota: En la mayoría de los casos probablemente quieras tomar la salida resultante y colocarla en otro archivo:

```

sysadmin@localhost:~$ tr 'a-z' 'A-Z' < example.txt > newexample.txt
sysadmin@localhost:~$ more newexample.txt
/etc/ppp:
CHAP-SECRETS
IP-DOWN
IP-DOWN.IPV6TO4
IP-UP
IP-UP.IPV6TO4
IPV6-DOWN
IPV6-UP
OPTIONS
PAP-SECRETS
sysadmin@localhost:~$

```

## 8.4 Buscar Archivos Utilizando el Comando Find

Uno de los retos al que se enfrentan los usuarios trabajando con el sistema de archivos, es tratar de recordar la ubicación donde se almacenan los archivos. Hay miles de archivos y cientos de directorios en un típico sistema de archivos Linux, así que recordar donde se encuentran estos archivos, puede plantear desafíos.

Ten en cuenta que la mayoría de los archivos con los que trabajas, son los que tú creas. Como resultado, a menudo buscarás en tu propio directorio local para encontrar los archivos. Sin embargo, a veces puede que necesites buscar en otros lugares en el sistema de archivos para encontrar archivos creados por otros usuarios. El comando `find` es una herramienta muy poderosa que puedes utilizar para buscar archivos en el sistema de archivos. Este comando puede buscar archivos por nombre, incluso usando los caracteres comodín cuando no estás seguro del nombre exacto del archivo. Además, puedes buscar los archivos en función de los metadatos de archivos, tales como tipo de archivo, tamaño de archivo y propiedad de archivo.

La sintaxis del comando `find` es:

```
find [directorio de inicio] [opción de búsqueda] [criterio de búsqueda] [opción de resultado]
```

O en Inglés:

```
find [starting directory] [search option] [search criteria] [result option]
```

Descripción de todos estos componentes:

| Component              | Description  |
|------------------------|--|
| [directorio de inicio] | Aquí el usuario especifica dónde comenzar la búsqueda. El comando <code>find</code> buscará en este directorio y todos sus subdirectorios. Si no hay directorio de partida, el directorio actual se utiliza para el punto de partida |
| [opción de búsqueda]   | Aquí el usuario especifica una opción para determinar qué tipo de metadatos hay que buscar; hay opciones para el nombre de archivo, tamaño de archivo y muchos otros atributos de archivo.   |
| [criterio de búsqueda] | Es un argumento que complementa la opción de búsqueda. Por ejemplo, si el usuario utiliza la opción para buscar un nombre de archivo, el criterio de búsqueda sería el nombre del archivo.   |
| [opción de resultado]  | Esta opción se utiliza para especificar qué acción se debe tomar al encontrar el archivo. Si no se proporciona ninguna opción, se imprimirá el nombre del archivo a STDOUT.  |

8.4.1 Buscar por Nombre de Archivo

Para buscar un archivo por nombre, utiliza la opción `-name` (o «nombre» en español) del comando `find` (o «buscar» en español):

```
sysadmin@localhost:~$ find /etc -name hosts
find: `/etc/dhcp': Permission denied
find: `/etc/cups/ssl': Permission denied
find: `/etc/pki/CA/private': Permission denied
find: `/etc/pki/rsyslog': Permission denied
find: `/etc/audisp': Permission denied
find: `/etc/named': Permission denied
find: `/etc/lvm/cache': Permission denied
find: `/etc/lvm/backup': Permission denied
find: `/etc/lvm/archive': Permission denied
/etc/hosts
find: `/etc/ntp/crypto': Permission denied
find: `/etc/polkit-1/localauthority': Permission denied
find: `/etc/sudoers.d': Permission denied
find: `/etc/sss': Permission denied
/etc/avahi/hosts
find: `/etc/selinux/targeted/modules/active': Permission denied
find: `/etc/audit': Permission denied
sysadmin@localhost:~$
```

Observa que se encontraron dos archivos: `/etc/hosts` y `/etc/avahi/hosts`. El resto de la salida eran los mensajes STDERR porque el usuario que ejecutó el comando no tiene permiso para acceder a ciertos subdirectorios.

Recuerda que puedes redirigir el STDERR a un archivo por lo que no necesitarás ver estos mensajes de error en la pantalla:

```
sysadmin@localhost:~$ find /etc -name hosts 2> errors.txt
/etc/hosts
/etc/avahi/hosts
sysadmin@localhost:~$
```

Mientras que la salida es más fácil de leer, realmente no hay ningún propósito para almacenar los mensajes de error en el archivo `error.txt`. Los desarrolladores de Linux se dieron cuenta de que sería bueno tener un **archivo de «basura»** (o «junk» en inglés) **para enviar los datos innecesarios; se descarta cualquier archivo que envíes al archivo `/dev/null`:**

```
sysadmin@localhost:~$ find /etc -name hosts 2> /dev/null
/etc/hosts
/etc/avahi/hosts
sysadmin@localhost:~$
```

#### 8.4.2 Mostrando los Detalles del Archivo

Puede ser útil obtener información sobre el archivo al utilizar el comando `find`, porque solo el nombre del archivo en sí mismo podría no proporcionar información suficiente para que puedas encontrar el archivo correcto.

Por ejemplo, puede haber siete archivos llamados `hosts`; si supieras que el archivo `host` que necesitas se había modificado recientemente, entonces sería útil ver la hora de modificación del archivo.

Para ver estos detalles del archivo, utiliza la opción `-ls` para el comando `find`:

```
sysadmin@localhost:~$ find /etc -name hosts -ls 2> /dev/null
  41    4 -rw-r--r--    1 root    root      158 Jan 12 2010 /etc/hosts
6549    4 -rw-r--r--    1 root    root     1130 Jul 19 2011 /etc/avahi/hosts
sysadmin@localhost:~$
```

**Nota:** Las dos primeras columnas de la salida anterior son el *número inodo* del archivo y el número de *bloques* que el archivo utiliza para el almacenamiento. Ambos están más allá del alcance del tema en cuestión. El resto de las columnas son la salida típica del comando `ls -l`: tipo de archivo, permisos, cuenta de enlaces físico, usuario propietario, grupo propietario, tamaño del archivo, hora de modificación del archivo y el nombre de archivo.

#### 8.4.3 Buscar Archivos por Tamaño

Una de las muchas opciones útiles de la búsqueda es la opción que le permite **buscar archivos por tamaño**. La opción `-size` (o «tamaño» en español) te permite buscar los archivos que son mayores o menores que un tamaño especificado, así como buscar un tamaño de archivo exacto.

Cuando se especifica un tamaño de archivo, puedes proporcionar el tamaño en bytes (c), kilobytes (k), megabytes (M) o gigabytes (G). Por ejemplo, la siguiente será una búsqueda de archivos en la estructura de directorios `/etc` con el tamaño exacto de 10 bytes:

```
sysadmin@localhost:~$ find /etc -size 10c -ls 2>/dev/null
  432    4 -rw-r--r--    1 root    root          10 Jan 28  2015 /etc/adjtime
 8814    0 drwxr-xr-x    1 root    root          10 Jan 29  2015 /etc/ppp/ip-d
own.d
 8816    0 drwxr-xr-x    1 root    root          10 Jan 29  2015 /etc/ppp/ip-u
p.d
```

```

8921    0 lrwxrwxrwx    1 root    root          10 Jan 29  2015 /etc/ssl/cert
s/349f2832.0 -> EC-ACC.pem
 9234    0 lrwxrwxrwx    1 root    root          10 Jan 29  2015 /etc/ssl/cert
s/aeb67534.0 -> EC-ACC.pem
73468    4 -rw-r--r--    1 root    root          10 Nov 16 20:42 /etc/hostname
sysadmin@localhost:~$

```

Si quieres buscar los archivos que son más grandes que un tamaño especificado, puedes usar el carácter + antes que el tamaño. Por ejemplo, el siguiente ejemplo buscará todos los archivos en la estructura de directorio /usr que su tamaño sea mayor a 100 megabytes:

```

sysadmin@localhost:~$ find /usr -size +100M -ls 2> /dev/null
574683 104652 -rw-r--r--    1 root    root        107158256 Aug  7 11:06 /usr/share/icons/oxygen/icon-theme.cache
sysadmin@localhost:~$

```

Si quieres buscar los archivos que son más pequeños que un tamaño especificado, puedes usar el carácter - antes que el tamaño.

#### 8.4.4 Opciones de Búsqueda Útiles Adicionales

Hay muchas opciones de búsqueda. La siguiente tabla muestra algunas:

| Opción                 | Significado  |
|------------------------|--|
| <code>-maxdepth</code> | Permite al usuario especificar la profundidad en la estructura de los directorios a buscar. Por ejemplo, <code>-maxdepth 1</code> significaría sólo buscar en el directorio especificado y en sus subdirectorios inmediatos. |
| <code>-group</code>    | Devuelve los archivos que son propiedad de un grupo especificado. Por ejemplo, <code>-group payroll</code> devolvería los archivos que son propiedad del grupo payroll (o «nómina» en español).                              |
| <code>-iname</code>    | Devuelve los archivos especificados que coinciden con el nombre de archivo, pero a diferencia del <code>-name</code> , <code>-iname</code> no es sensible a las  |

| Opción             | Significado   |
|--------------------|---|
|                    | mayúsculas y minúsculas. Por ejemplo, <code>-iname hosts</code> coincidiría con los archivos llamados <code>hosts</code> , <code>Hosts</code> , <code>HOSTS</code> , etc.                       |
| <code>-mmin</code> | Devuelve los archivos que fueron modificados según el tiempo de modificación en minutos. Por ejemplo, <code>-mmin 10</code> coincidirá con los archivos que fueron modificados hace 10 minutos. |
| <code>-type</code> | Devuelve los archivos que coinciden con el tipo de archivo. Por ejemplo, <code>-type f</code> devuelve los archivos que son archivos regulares.   |
| <code>-user</code> | Devuelve los archivos que son propiedad de un usuario especificado. Por ejemplo, <code>-user bob</code> devuelve los archivos que son propiedad del usuario <code>bob</code> .                  |

#### 8.4.5 Usar Múltiples Opciones

Si utilizas múltiples opciones, éstas actúan como un operador lógico "y", lo que significa que para que se dé una coincidencia, todos los criterios deben coincidir, no sólo uno. Por ejemplo, el siguiente comando muestra todos los archivos en la estructura de directorio `/etc` con el tamaño de 10 bytes y que son archivos simples:

```
sysadmin@localhost:~$ find /etc -size 10c -type f -ls 2>/dev/null
432    4 -rw-r--r--    1 root    root          10 Jan 28  2015 /etc/adjtime
73468  4 -rw-r--r--    1 root    root          10 Nov 16  20:42 /etc/hostname
sysadmin@localhost:~$
```

#### 8.5 Visualización de los Archivos Utilizando el Comando less

Mientras que visualizar pequeños archivos con el comando `cat` no plantea ningún problema, no es una opción ideal para los archivos grandes. El comando `cat` no proporciona ninguna manera para pausar y reiniciar la pantalla fácilmente, por lo que el contenido del archivo entero es arrojado a la pantalla.

Para archivos más grandes, querrás usar el comando *pager* que permite ver el contenido. Los comandos *pager* mostrarán una página de datos a la vez, permitiéndote moverte hacia adelante y hacia atrás en el archivo utilizando las teclas de movimiento.

Hay dos comandos *pager* comúnmente utilizados:

- El comando `less`: Este comando proporciona una capacidad de paginación muy avanzada. Normalmente es el pager predeterminado utilizado por comandos como el comando `man`.
- El comando `more`: Este comando ha existido desde los primeros días de UNIX. Aunque tenga menos funciones que el comando `less`, tiene una ventaja importante: El comando `less` no viene siempre incluido en todas las distribuciones de Linux (y en algunas distribuciones, no viene instalado por defecto). El comando `more` siempre está disponible.

Cuando utilizas los comandos `more` o `less`, te permiten «moverte» en un documento utilizando los *comandos de tecla*. Ya que los desarrolladores del comando `less` basaron el comando en la funcionalidad del comando `more`, todos los comandos de tecla disponibles en el comando `more` también trabajan en el comando `less`.

En este manual nos centraremos más en el comando más avanzado (`less`). Sin embargo, resulta útil que te acuerdes del comando `more` para cuando el comando `less` no esté disponible. Recuerda que la mayoría de los comandos de tecla proporcionados trabajan para ambos comandos.

### 8.5.1 La Pantalla de Ayuda con el Comando `less`

Al ver un archivo con el comando `less`, puedes utilizar la tecla `h` para mostrar una pantalla de ayuda. La pantalla de ayuda te permite ver que otros comandos están disponibles. En el ejemplo siguiente, se ejecuta el comando `less /usr/share/dict/words`. Cuando se visualiza el documento, se presiona la tecla `h`, y se muestra la pantalla de ayuda:

```

SUMMARY OF LESS COMMANDS

Commands marked with * may be preceded by a number, N.

Notes in parentheses indicate the behavior if N is given.

h  H                Display this help.
q  :q  Q  :Q  ZZ    Exit.

-----

MOVING

e  ^E  j  ^N  CR  *  Forward one line (or N lines).
y  ^Y  k  ^K  ^P  *  Backward one line (or N lines).
f  ^F  ^V  SPACE *  Forward one window (or N lines).
b  ^B  ESC-v      *  Backward one window (or N lines).
z                                *  Forward one window (and set window to N).
w                                *  Backward one window (and set window to N).
ESC-SPACE            *  Forward one window, but don't stop at end-of-file.
d  ^D                *  Forward one half-window (and set half-window to N).
u  ^U                *  Backward one half-window (and set half-window to N).
```

```
ESC-) RightArrow * Left one half screen width (or N positions).
ESC-( LeftArrow * Right one half screen width (or N positions).
HELP -- Press RETURN for more, or q when done
```

### 8.5.2 Los Comandos de Movimiento para less

Hay muchos comandos de movimiento para el comando `less`, cada uno con múltiples teclas o combinaciones de teclas. Si bien esto puede parecer intimidante, recuerda que no necesitas memorizar todos estos comandos de movimiento; siempre puedes utilizar la tecla **h** para obtener la ayuda.

El primer grupo de comandos de movimiento en los que probablemente te quieras enfocar son los que más comúnmente se utilizan. Para hacerlo aún más fácil de aprender, se resumirán las teclas que son idénticas para `more` y `less`. De esta manera, aprenderás cómo moverte en los comandos `more` y `less` al mismo tiempo:

| Movimiento             | Tecla             |
|------------------------|-------------------|
| Ventana hacia adelante | Barra espaciadora |
| Ventana hacia atrás    | b                 |
| Línea hacia adelante   | Entrar            |
| Salir                  | q                 |
| Ayuda                  | h                 |

Cuando se utiliza el comando `less` para moverse entre las páginas, la forma más fácil de avanzar una página hacia adelante es presionando la **barra espaciadora**.

### 8.5.3 Comandos de Búsqueda less

Hay dos formas de buscar con el comando `less`: puedes buscar hacia adelante o hacia atrás desde tu posición actual usando patrones llamados las expresiones regulares. Más detalles en relación con las expresiones regulares se proporcionarán más adelante en este capítulo.

Para iniciar una búsqueda hacia adelante desde tu posición actual, utiliza la tecla **/**. A continuación, escribe el texto o el patrón y presiona la tecla **Entrar**.

Si se encuentra una coincidencia, el cursor se moverá en el documento hasta encontrar una coincidencia. Por ejemplo, en el siguiente gráfico la expresión «`frog`» (o «rana» en español) se buscó en el archivo `/usr/share/dict/words`:

```
bullfrog
bullfrog's
bullfrogs
```



```

bullheaded
bullhorn
bullhorn's
bullhorns
bullied
bullies
bulling
bullion
bullion's
bullish
bullock
bullock's
bullocks
bullpen
bullpen's
bullpens
bullring
bullring's
bullrings
bulls
:

```

Observa que «frog» no tiene que ser una palabra por sí misma. Observa también, que mientras el comando `less` te llevó a la primera coincidencia desde tu posición actual, todas las coincidencias se resaltaron.

Si no se encuentra ninguna coincidencia hacia adelante desde tu posición actual, la última línea de la pantalla reportará «Pattern not found» (o «Patrón no encontrado» en español):

```
Pattern not found (press RETURN)
```

Para iniciar una búsqueda hacia atrás desde tu posición actual, pulsa la tecla `?`, después introduce el texto o el patrón y presiona la tecla **Entrar**. El cursor se moverá hacia atrás hasta encontrar la primera coincidencia o te informará que no se puede encontrar el patrón.

Si la búsqueda encuentra más de una coincidencia, entonces con la tecla **n** te puedes mover a la siguiente coincidencia y la tecla **N** te permitirá ir a la coincidencia anterior.

## 8.6 Revisando los Comandos head y tail

Recordemos que los comandos `head` y `tail` se utilizan para filtrar los archivos y mostrar un número limitado de líneas. Si quieres ver un número de líneas seleccionadas desde la parte superior del archivo, utiliza el comando `head` y si quieres ver un número de líneas seleccionadas en la parte inferior de un archivo, utiliza el comando `tail`.

Por defecto, ambos comandos muestran diez líneas del archivo. La siguiente tabla proporciona algunos ejemplos:

| Comando de Ejemplo                 | Explicación del texto visualizado   |
|------------------------------------|---|
| <code>head /etc/passwd</code>      | Las primeras diez líneas del archivo <code>/etc/passwd</code>   |
| <code>head -3 /etc/group</code>    | Las primeras tres líneas del archivo <code>/etc/group</code>  |
| <code>head -n 3 /etc/group</code>  | Las primeras tres líneas del archivo <code>/etc/group</code>  |
| <code>help   head</code>           | Las primeras diez líneas de la salida del comando <code>help</code> redirigidas por la barra vertical |
| <code>tail /etc/group</code>       | Las últimas diez líneas del archivo <code>/etc/group</code>   |
| <code>tail -5 /etc/passwd</code>   | Las últimas cinco líneas del archivo <code>/etc/passwd</code>   |
| <code>tail -n 5 /etc/passwd</code> | Las últimas cinco líneas del archivo <code>/etc/passwd</code>   |
| <code>help   tail</code>           | Las últimas diez líneas de la salida del comando <code>help</code> redirigidas por la barra vertical  |

Como puedes observar en los ejemplos anteriores, ambos comandos darán salida al texto de un archivo regular o de la salida de cualquier comando enviado mediante la barra vertical. Ambos utilizan la opción `-n` para indicar cuántas líneas debe contener la salida.

#### 8.6.1 El Valor Negativo de la Opción `-n`

Tradicionalmente en UNIX, se especifica el número de líneas a mostrar como una opción con cualquiera de los comandos, así pues `-3` significa mostrar tres líneas. Para el comando `tail`, la opción `-3` o `-n -3` siempre significará mostrar tres líneas. Sin embargo, la versión GNU del comando `head` reconoce `-n -3` como mostrar **todo menos las tres últimas líneas**, y sin embargo el comando `head` siempre reconoce la opción `-3` como muestra las tres primeras líneas.

#### 8.6.2 El Valor Positivo del Comando `tail`

La versión GNU del comando `tail` permite una variación de cómo especificar el número de líneas que se deben imprimir. Si utilizas la opción `-n` con un número precedido por el signo más, entonces el comando `tail` reconoce esto en el sentido de mostrar el contenido a partir de la línea especificada y continuar hasta el final.

Por ejemplo, el siguiente ejemplo muestra la línea #22 hasta el final de la salida del comando `nl`:

```
sysadmin@localhost:~$ nl /etc/passwd | tail -n +22
22  sshd:x:103:65534::/var/run/sshd:/usr/sbin/nologin
23  operator:x:1000:37::/root:/bin/sh
24  sysadmin:x:1001:1001:System Administrator,,,:/home/sysadmin:/bin/bash
sysadmin@localhost:~$
```

#### 8.6.3 Seguimiento de los Cambios en un Archivo

Puedes ver los cambios en vivo en los archivos mediante la opción `-f` del comando `tail`. Esto es útil cuando quieres seguir cambios en un archivo mientras están sucediendo.

Un buen ejemplo de esto sería cuando quieres ver los archivos de registro como el administrador de sistemas. Los archivos de registro pueden utilizarse para solucionar los problemas y a menudo los administradores los verán en una ventana independiente de manera «interactiva» utilizando el comando `tail` mientras van ejecutando los comandos con los cuáles están intentando solucionar los problemas.

Por ejemplo, si vas a iniciar una sesión como el usuario `root`, puedes solucionar los problemas con el servidor de correo electrónico mediante la visualización de los cambios en tiempo real de su archivo de registro utilizando el comando siguiente: `tail -f /var/log/mail.log`

### 8.7 Ordenar Archivos o Entradas

Puede utilizarse el comando `sort` para reorganizar las líneas de archivos o entradas en orden alfabético o numérico basado en el contenido de uno o más campos. Los campos están determinados por un separador de campos contenido en cada línea, que por defecto son espacios en blanco (espacios y tabuladores).

En el ejemplo siguiente se crea un pequeño archivo, usando el comando `head` tomando las 5 primeras líneas del archivo `/etc/passwd` y enviando la salida a un archivo llamado `mypasswd`.

```
sysadmin@localhost:~$ head -5 /etc/passwd > mypasswd
sysadmin@localhost:~$
sysadmin@localhost:~$ cat mypasswd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
```

```
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
sysadmin@localhost:~$
```

Ahora vamos a ordenar (`sort`) el archivo `mypasswd`:

```
sysadmin@localhost:~$ sort mypasswd
bin:x:2:2:bin:/bin:/bin/sh
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
root:x:0:0:root:/root:/bin/bash
sync:x:4:65534:sync:/bin:/bin/sync
sys:x:3:3:sys:/dev:/bin/sh
sysadmin@localhost:~$
```

### 8.7.1 Opciones de Campo y Ordenamiento

En el caso de que el archivo o entrada pueda separarse por otro delimitador como una coma o dos puntos, la opción `-t` permitirá especificar otro separador de campo. Para especificar los campos para `sort` (o en español «ordenar»), utiliza la opción `-k` con un argumento para indicar el número de campo (comenzando con 1 para el primer campo).

Las otras opciones comúnmente usadas para el comando `sort` son `-n` para realizar un `sort` numérico y `-r` para realizar a un `sort` inverso.

En el siguiente ejemplo, se utiliza la opción `-t` para separar los campos por un carácter de dos puntos y realiza un `sort` numérico utilizando el tercer campo de cada línea:

```
sysadmin@localhost:~$ sort -t: -n -k3 mypasswd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
sysadmin@localhost:~$
```

Ten en cuenta que la opción `-r` se podía haber utilizado para invertir el `sort`, causando que los números más altos en el tercer campo aparecieran en la parte superior de la salida:

```
sysadmin@localhost:~$ sort -t: -n -r -k3 mypasswd
sync:x:4:65534:sync:/bin:/bin/sync
sys:x:3:3:sys:/dev:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
root:x:0:0:root:/root:/bin/bash
sysadmin@localhost:~$
```

Por último, puede que quieras ordenarlo de una forma más compleja, como un `sort` por un campo primario y luego por un campo secundario. Por ejemplo, considera los siguientes datos:

```
bob:smith:23
nick:jones:56
sue:smith:67
```

Puede que quieras ordenar primero por el apellido (campo #2), luego el nombre (campo #1) y luego por edad (campo #3). Esto se puede hacer con el siguiente comando:

```
sysadmin@localhost:~$ sort -t: -k2 -k1 -k3n filename
```

## 8.8 Visualización de las Estadísticas de Archivo con el Comando `wc`

El comando `wc` permite que se impriman hasta tres estadísticas por cada archivo, así como el total de estas estadísticas, siempre que se proporcione más de un nombre de archivo. De forma predeterminada, el comando `wc` proporciona el número de líneas, palabras y bytes (1 byte = 1 carácter en un archivo de texto):

```
sysadmin@localhost:~$ wc /etc/passwd /etc/passwd-
 35   56 1710 /etc/passwd
 34   55 1665 /etc/passwd-
 69  111 3375 total
sysadmin@localhost:~$
```

El ejemplo anterior muestra la salida de ejecutar: `wc /etc/passwd /etc/passwd-`. La salida tiene cuatro columnas: el número de líneas en el archivo, el número de palabras en el archivo, el número de bytes en el archivo y el nombre de archivo o `total`.

Si quieres ver estadísticas específicas, puede utilizar `-l` para mostrar sólo el número de líneas, `-w` para mostrar sólo el número de palabras y `-c` para mostrar sólo el número de bytes.

El comando `wc` puede ser útil para contar el número de líneas devueltas por algún otro comando utilizando la barra vertical. Por ejemplo, si desea saber el número total de los archivos en el directorio `/etc`, puedes ejecutar `ls /etc | wc -l`:

```
sysadmin@localhost:~$ ls /etc/ | wc -l
136
sysadmin@localhost:~$
```

## 8.9 Utilizar el Comando cut para Filtrar el Contenido del Archivo

El comando `cut` (o «cortar» en español) puede extraer columnas de texto de un archivo o entrada estándar. El uso primario del comando `cut` es para trabajar con los archivos delimitados de las bases de datos. Estos archivos son muy comunes en los sistemas Linux.

De forma predeterminada, considera que su entrada debe ser separada por el carácter **Tab**, pero la opción `-d` puede especificar delimitadores alternativos como los dos puntos o la coma.

Utilizando la opción `-f` puedes especificar qué campos quieres que se muestren, ya sea como un rango separado por guiones o una lista separada por comas.

En el ejemplo siguiente se visualiza el primero, quinto, sexto y séptimo campo del archivo de la base de datos de `mypasswd`:

```
sysadmin@localhost:~$ cut -d: -f1,5-7 mypasswd
root:root:/root:/bin/bash
daemon:daemon:/usr/sbin:/bin/sh
bin:bin:/bin:/bin/sh
sys:sys:/dev:/bin/sh
sync:sync:/bin:/bin/sync
sysadmin@localhost:~$
```

Con el comando `cut` puedes también extraer las columnas de un texto basado en la posición de los caracteres con la opción `-c`. Esto puede ser útil para extraer los campos de archivos de base de datos con un ancho fijo. El siguiente ejemplo muestra sólo el tipo de archivo (carácter #1), los permisos (caracteres #2-10) y el nombre de archivo (caracteres #50+) de la salida del comando `ls -l`:

```
sysadmin@localhost:~$ ls -l | cut -c1-11,50-
total 12
drwxr-xr-x Desktop
drwxr-xr-x Documents
drwxr-xr-x Downloads
drwxr-xr-x Music
drwxr-xr-x Pictures
```

```
drwxr-xr-x Public
drwxr-xr-x Templates
drwxr-xr-x Videos
-rw-rw-r-- errors.txt
-rw-rw-r-- mypasswd
-rw-rw-r-- new.txt
sysadmin@localhost:~$
```

### 8.10 Utilizar el Comando grep para Filtrar el Contenido del Archivo

Puedes utilizar el comando `grep` para filtrar las líneas en un archivo o en la salida de otro comando basado en la coincidencia de un patrón. El patrón puede ser tan simple como el texto exacto que quieres que coincida o puede ser mucho más avanzado mediante el uso de las expresiones regulares (de las que hablaremos más adelante en este capítulo).

Por ejemplo, puedes querer encontrar todos los usuarios que pueden ingresar al sistema con el shell BASH, por lo que se podría utilizar el comando `grep` para filtrar las líneas del archivo `/etc/passwd` para las líneas que contengan los caracteres `bash`:

```
sysadmin@localhost:~$ grep bash /etc/passwd
root:x:0:0:root:/root:/bin/bash
sysadmin:x:1001:1001:System Administrator,,,:/home/sysadmin:/bin/bash
sysadmin@localhost:~$
```

Para que sea más fácil ver exactamente lo que coincide, utiliza la opción de `--color`. Esta opción resaltará los elementos coincidentes en rojo:

```
sysadmin@localhost:~$ grep --color bash /etc/passwd
root:x:0:0:root:/root:/bin/bash
sysadmin:x:1001:1001:System Administrator,,,:/home/sysadmin:/bin/bash
sysadmin@localhost:~$
```

En algunos casos no te interesan las líneas específicas que coinciden con el patrón, sino más bien cuántas líneas coinciden con el patrón. Con la opción `-c` puedes obtener un conteo del número de líneas que coinciden:

```
sysadmin@localhost:~$ grep -c bash /etc/passwd
2
sysadmin@localhost:~$
```

Cuando estás viendo la salida del comando `grep`, puede ser difícil determinar el número original de las líneas. Esta información puede ser útil cuando regreses al archivo (tal vez para editar el archivo), ya que puedes utilizar esta información para encontrar rápidamente una de las líneas coincidentes.

La opción `-n` del comando `grep` muestra los números de la línea originales:

```
sysadmin@localhost:~$ grep -n bash /etc/passwd
1:root:x:0:0:root:/root:/bin/bash
24:sysadmin:x:1001:1001:System Administrator,,,:/home/sysadmin:/bin/bas
sysadmin@localhost:~$
```

Algunas opciones adicionales del `grep` que pueden resultar útiles:

| Ejemplos                                 | Salida   |
|--|--|
| <code>grep -v nologin /etc/passwd</code> | Todas las líneas que no contengan <code>nologin</code> en el archivo <code>/etc/passwd</code>  |
| <code>grep -l linux /etc/*</code>        | Lista de los archivos en el directorio <code>/etc</code> que contengan <code>linux</code>  |
| <code>grep -i linux /etc/*</code>        | Listado de las líneas de los archivos en el directorio <code>/etc</code> que contengan cualquier tipo de letra (mayúscula o minúscula) del patrón de caracteres <code>linux</code> |
| <code>grep -w linux /etc/*</code>        | Listado de las líneas de los archivos en el directorio <code>/etc</code> que contengan el patrón de la palabra <code>linux</code>  |

## 8.11 Las Expresiones Regulares Básicas

Una **Expresión Regular** es una colección de caracteres «normales» y «especiales» que se **utilizan para que coincida con un patrón simple o complejo**. Los caracteres normales son caracteres alfanuméricos que coinciden con ellos mismos. Por ejemplo, la letra `a` coincide con una `a`.



Algunos caracteres tienen significados especiales cuando se utilizan dentro de los patrones por comandos como el comando `grep`. Existen las *Expresiones Regulares Básicas* (disponible para una amplia variedad de comandos de Linux) y las *Expresiones Regulares Extendidas* (disponibles para los comandos más avanzados de Linux). Las Expresiones Regulares Básicas son las siguientes:

| Expresión Regular | Coincidencias   |
|-------------------|---|
| .                 | Cualquier carácter individual   |
| [ ]               | Una lista o rango de caracteres que coinciden con un carácter, a menos que el primer carácter sea el símbolo de intercalación ^, lo que entonces significa cualquier carácter que no esté en la lista |
| *                 | El carácter previo que se repite cero o más veces   |
| ^                 | El texto siguiente debe aparecer al principio de la línea   |
| \$                | El texto anterior debe aparecer al final de la línea  |

El comando `grep` es sólo uno de los muchos comandos que admiten expresiones regulares. Algunos otros comandos son los comandos `more` y `less`. Mientras que a algunas de las expresiones regulares se les ponen innecesariamente con comillas simples, es una buena práctica utilizar comillas simples con tus expresiones regulares para evitar que el shell trate a interpretar su significado especial.

#### 8.11.1 Expresiones Regulares Básicas - el Carácter .

En el ejemplo siguiente, un simple archivo primero se crea usando la redirección. Después el comando `grep` se utiliza para mostrar una coincidencia de patrón simple:

```
sysadmin@localhost:~$ echo 'abcddd' > example.txt
sysadmin@localhost:~$ cat example.txt
abcddd
sysadmin@localhost:~$ grep --color 'a..' example.txt
abcddd
sysadmin@localhost:~$
```

En el ejemplo anterior, se puede ver que el patrón `a..` coincidió con `abc`. El primer carácter `.` coincidió con `b` y el segundo coincidió con `c`.

En el siguiente ejemplo, el patrón `a..c` no coincide con nada, así que el comando `grep` no produce ninguna salida. Para que la coincidencia tenga éxito, hay que poner dos caracteres entre `a` y `c` en el `example.txt`:

```
sysadmin@localhost:~$ grep --color 'a..c' example.txt
sysadmin@localhost:~$
```

### 8.11.2 Expresiones Regulares Básicas - el Carácter [ ]

Si usas el carácter `[]`, entonces cualquier carácter posible podría coincidir. En algunos casos quieres especificar exactamente los caracteres que quieres que coincidan. Por ejemplo, tal vez quieres que coincida un sólo carácter alfa minúscula o un carácter de número. Para ello, puedes utilizar los caracteres de expresiones regulares `[]` y especificar los caracteres válidos dentro de los caracteres `[]`.

Por ejemplo, el siguiente comando coincide con dos caracteres, el primero es ya sea una `a` o una `b` mientras que el segundo es una `a`, `b`, `c` o `d`:

```
sysadmin@localhost:~$ grep --color '[ab][a-d]' example.txt
abcccc
sysadmin@localhost:~$
```

Ten en cuenta que puedes enumerar cada carácter posible `[abcd]` o proporcionar un rango `[a-d]` siempre que esté en el orden correcto. Por ejemplo, `[d-a]` no funcionaría ya que no es un intervalo válido:

```
sysadmin@localhost:~$ grep --color '[d-a]' example.txt
grep: Invalid range end
sysadmin@localhost:~$
```

El rango se especifica mediante un estándar denominado la tabla ASCII. Esta tabla es una colección de todos los caracteres imprimibles en un orden específico. Puedes ver la tabla ASCII con el comando `man ascii`. Un pequeño ejemplo:

|     |    |    |    |     |     |    |   |
|-----|----|----|----|-----|-----|----|---|
| 041 | 33 | 21 | !  | 141 | 97  | 61 | a |
| 042 | 34 | 22 | "  | 142 | 98  | 62 | b |
| 043 | 35 | 23 | #  | 143 | 99  | 63 | c |
| 044 | 36 | 24 | \$ | 144 | 100 | 64 | d |
| 045 | 37 | 25 | %  | 145 | 101 | 65 | e |
| 046 | 38 | 26 | &  | 146 | 102 | 66 | f |

Puesto que la `a` tiene un valor numérico más pequeño (141) que la `d` (144), el rango `a-d` incluye todos los caracteres de la `a` a la `d`.

¿Qué pasa si quieres un carácter que puede ser cualquier cosa menos una `x`, `y` o `z`? No querrías proporcionar un conjunto de `[]` con todos los caracteres excepto `x`, `y` o `z`.

Para indicar que quieres que coincida un carácter que **no es uno de lo listados**, inicia tu conjunto de `[ ]` con el símbolo `^`. El siguiente ejemplo muestra la coincidencia de un patrón que incluye un carácter que no es un a, b o c seguida de un d:

```
sysadmin@localhost:~$ grep --color '^[abc]d' example.txt
abcddd
sysadmin@localhost:~$
```

### 8.11.3 Expresiones Regulares Básicas - el Carácter `*`

El **carácter** `*` se puede utilizar **para coincidir** con «cero o más de los caracteres previos». El siguiente ejemplo coincidirá con cero o más caracteres d:

```
sysadmin@localhost:~$ grep --color 'd*' example.txt
abcdddd
sysadmin@localhost:~$
```

### 8.11.4 Expresiones Regulares Básicas - los Caracteres `^` y `$`

Cuando quieres que coincida un patrón, tal coincidencia puede ocurrir en cualquier lugar de la línea. Puede que quieras especificar que la coincidencia se presentara al principio de la línea o al final de la línea. **Para que coincida con el principio de la línea**, **comienza** el patrón con el **símbolo** `^`.

En el ejemplo siguiente se agrega otra línea al archivo `example.txt` para mostrar el uso del símbolo `^`:

```
sysadmin@localhost:~$ echo "xyzabc" >> example.txt
sysadmin@localhost:~$ cat example.txt
abcddd
xyzabc
sysadmin@localhost:~$ grep --color "a" example.txt
abcddd
xyzabc
sysadmin@localhost:~$ grep --color "^a" example.txt
abcddd
sysadmin@localhost:~$
```

Ten en cuenta que en la primera salida del `grep`, **ambas líneas coinciden** debido a que **ambas contienen la letra** a. En la segunda salida `grep`, solo coincide con la línea que comienza con la letra a.

Para especificar que **coincida al final de la línea**, termina el patrón con el **carácter** `$`. Por ejemplo, para encontrar sólo las líneas que terminan con la letra c:

```
sysadmin@localhost:~$ grep "c$" example.txt
xyzabc
```

```
sysadmin@localhost:~$
```

### 8.11.5 Expresiones Regulares Básicas - el Carácter \

En algunos casos querrás que la búsqueda coincida con un carácter que resulta ser un carácter especial de la expresión regular. Observa el siguiente ejemplo:

```
sysadmin@localhost:~$ echo "abcd*" >> example.txt
sysadmin@localhost:~$ cat example.txt
abcddd
xyzabc
abcd*
sysadmin@localhost:~$ grep --color "cd*" example.txt
abcddd
xyzabc
abcd*
sysadmin@localhost:~$
```

En la salida del comando `grep` anterior, ves que cada línea corresponde porque estás buscando el carácter `c` seguido de cero o más caracteres `d`. Si quieres buscar un carácter `*`, coloca el carácter `\` antes del carácter `*`:

```
sysadmin@localhost:~$ grep --color "cd\*" example.txt
abcd*
sysadmin@localhost:~$
```

## 8.12 Expresiones Regulares Extendidas

El uso de las Expresiones Regulares Extendidas requiere a menudo una opción especial proporcionada al comando para que éste las reconozca. Históricamente, existe un comando llamado `egrep`, que es similar al `grep`, pero es capaz de entender su uso. Ahora, el comando `egrep` es obsoleto a favor del uso del `grep` con la opción `-E`.

Las siguientes expresiones regulares se consideran «extendidas»:

| RE | Significado   |
|----|---|
| ?  | Coincide con el carácter anterior cero o una vez más, así que es un carácter opcional |

## RE Significado

+ Coincide con el carácter anterior repetido una o más veces

| Alternación o como un operador lógico

Algunos ejemplos de expresiones regulares extendidas:

| Comando                                | Significado   | Coincidencias  |
|--|---|--|
| <code>grep -E 'colou?r' 2.txt</code>   | Haz coincidir <code>colo</code> seguido por cero o un carácter <code>u</code> | <code>color</code><br><code>colour</code>                            |
| <code>grep -E 'd+' 2.txt</code>        | Coincide con uno o más <code>d</code> caracteres                              | <code>d</code> <code>dd</code> <code>ddd</code><br><code>dddd</code> |
| <code>grep -E 'gray grey' 2.txt</code> | Coincide con <code>gray</code> o <code>grey</code>                            | <code>gray</code> <code>grey</code>                                  |

### 8.13 El Command `xargs`

El comando `xargs` se utiliza para **construir y ejecutar líneas de comandos de una entrada estándar**. Este comando es muy útil cuando se necesita ejecutar un comando con una lista de argumentos muy larga, que en algunos casos puede resultar en un error si la lista de argumentos es demasiado larga.

El comando `xargs` tiene una opción `-0`, que **desactiva la cadena** de fin de archivo, **permitiendo el uso de los argumentos que contengan espacios, comillas o barras diagonales inversas**.

El comando `xargs` es útil para permitir que los comandos se ejecuten de manera más eficiente. Su objetivo es construir la línea de comandos para que un comando se ejecute las menos veces posibles con tantos argumentos como sea posible, en lugar de ejecutar el comando muchas veces con un argumento cada vez.

El comando `xargs` funciona separando la lista de los argumentos en sublistas y ejecutando el comando con cada sublista. El número de los argumentos en cada sublista no excederá el número máximo de los argumentos para el comando ejecutado, y por lo tanto, evita el error de «Argument list too long» (o «Lista de argumentos demasiado larga» en español).

En el ejemplo siguiente se muestra un escenario donde el comando `xargs` permite que muchos archivos sean eliminados, y donde el uso de un carácter comodín normal (glob) falló:

```
sysadmin@localhost:~/many$ rm *
```

```
bash: /bin/rm: Argument list too long
sysadmin@localhost:~/many$ ls | xargs rm
sysadmin@localhost:~/many$
```

## NDG Linux Essentials: Capítulo 9: El Scripting básico.

### 9.1 Introducción

En este capítulo, vamos a hablar sobre cómo las herramientas que has aprendido hasta ahora pueden transformarse en scripts reutilizables.

Traducción Imagen: **Linux, la frontera final.** Linux es usado en el espacio por la NASA. El Fénix Mars Rover e incluso en la estación espacial internacional

### 9.2 Scripts Shell en Pocas Palabras

Un *script shell* es un archivo de comandos ejecutables que ha sido almacenado en un archivo de texto. Cuando el archivo se ejecuta, se ejecuta cada comando. Los scripts shell tienen acceso a todos los comandos del shell, incluyendo la lógica. Un script (o «secuencia de comandos» en español), por lo tanto, puede detectar la presencia de un archivo o buscar una salida particular y cambiar su comportamiento en consecuencia. Se pueden crear scripts para automatizar partes repetitivas de tu trabajo, que ahorran tu tiempo y aseguran consistencia cada vez que utilices el script. Por ejemplo, si ejecutas los mismos cinco comandos todos los días, puedes convertirlos en un script shell que reduce tu trabajo a un comando.

Un script puede ser tan simple como un único comando:

```
echo "Hello, World!"
```

El script, test.sh, consta de una sola línea que imprime la cadena Hello, ¡World! (o «¡Hola, Mundo!» en español) en la consola.

Ejecutar un script puede hacerse, ya sea pasándolo como un argumento a tu shell o ejecutándolo directamente:

```
sysadmin@localhost:~$ sh test.sh
Hello, World!
sysadmin@localhost:~$ ./test.sh
-bash: ./test.sh: Permission denied
sysadmin@localhost:~$ chmod +x ./test.sh
sysadmin@localhost:~$ ./test.sh
Hello, World!
```

En el ejemplo anterior, en primer lugar, el script se ejecuta como un argumento del shell. A continuación, se ejecuta el script directamente desde el shell. Es raro tener el directorio actual en la ruta de búsqueda binaria `$PATH`, por lo que el nombre viene con el prefijo `./` para indicar que se debe ejecutar en el directorio actual.

El error `Permission denied` (o «Permiso denegado» en español) significa que el script no ha sido marcado como ejecutable. Un comando `chmod` rápido después y el script funciona. El comando `chmod` se utiliza para cambiar los permisos de un archivo, que se explica en detalle en un capítulo posterior.

Hay varios shells con su propia sintaxis de lenguaje. Por lo tanto, los scripts más complicados indicarán un shell determinado, especificando la ruta absoluta al intérprete como la primera línea, con el prefijo `#!`, tal como lo muestra el siguiente ejemplo:

```
#!/bin/sh
echo "Hello, World!"
```

o

```
#!/bin/bash
echo "Hello, World!"
```

Los dos caracteres `#!` se llaman tradicionalmente el hash y el bang respectivamente, que conduce a la forma abreviada «*shebang*» cuando se utilizan al principio de un script.

Por cierto, el shebang (o crunchbang) se utiliza para los scripts shell tradicionales y otros lenguajes basados en texto como Perl, Ruby y Python. Cualquier archivo de texto marcado como ejecutable se ejecutará bajo el intérprete especificado en la primera línea mientras se ejecuta el script directamente. Si el script se invoca directamente como argumento a un intérprete, como `sh script` o `bash script`, se utilizará el shell proporcionado, independientemente de lo que está en la línea del shebang.

Ayuda sentirse cómodo utilizando un editor de texto antes de escribir los scripts shell, ya que se necesitarás crear los archivos de texto simple. Las herramientas de oficina tradicionales como LibreOffice, que dan salida a archivos que contienen la información de formato y otra información, no son adecuadas para esta tarea.

### 9.3 Editar los Scripts Shell

UNIX tiene muchos editores de texto y las ventajas del uno sobre el otro se debaten muy a menudo. Dos de ellos vienen mencionados específicamente en el programa del curso de los aspectos esenciales de LPI: El editor de GNU `nano` es un editor muy sencillo y bien adaptado para editar pequeños archivos de texto. El Visual Editor, `vi`, o su versión más reciente, VI mejorado (`vim`), es un editor muy potente pero tiene un arduo proceso de aprendizaje. Nosotros nos centraremos en el `nano`.

Introduce `nano test.sh` y verás una pantalla similar a esta:

```
GNU nano 2.2.6      File: test.sh      modified

#!/bin/sh

echo "Hello, World!"
echo -n "the time is "
date
```

```

^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text  ^C Cur Po
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text^T To Spell

```

El editor `nano` tiene algunas características que debes conocer. Simplemente escribes con tu teclado, utiliza las flechas para moverte y el botón **suprimir/retroceso** para borrar texto. A lo largo de la parte inferior de la pantalla puedes ver algunos comandos disponibles, que son sensible al contexto y cambian dependiendo de lo que estás haciendo. Si te encuentras directamente en la máquina Linux, o sea no te conectas a través de la red, puedes utilizar el ratón para mover el cursor y resaltar el texto.

Para familiarizarte con el editor, comienza a escribir un script shell sencillo dentro del editor `nano`:

```

GNU nano 2.2.6      File: test.sh      modified

#!/bin/sh

echo "Hello, World!"
echo -n "the time is "
date

^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text  ^C Cur Po
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text^T To Spell

```

Ten en cuenta que la opción de la parte inferior izquierda es `^X Exit` que significa «presiona **control** y **X** para salir». Presione **Ctrl** con **X** y la parte inferior cambiará:



```
Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ?
```

```
Y Yes
```

```
N No      ^C Cancel
```

En este punto, puedes salir del programa sin guardar los cambios pulsando la tecla **N** o guardar primero pulsando **Y** para guardar. El valor predeterminado es guardar el archivo con el nombre de archivo actual. Puedes presionar la tecla **Entrar** para guardar y salir.

Después de guardar regresarás al shell prompt. Regresa al editor. Esta vez pulsa **Ctrl** y **O** para guardar tu trabajo sin salir del editor. Los prompts son iguales en gran parte, salvo que estás de nuevo en el editor.

Esta vez utiliza las teclas de la flecha para mover el cursor a la línea que contiene el texto «The time is» (o «La hora es» en español). Presiona **Control** y **K** dos veces para cortar las dos últimas líneas al búfer de copia. Mueve el cursor a la línea restante y presiona **Control** y **U** una vez para pegar el búfer de copia a la posición actual. Esto hace que el script muestre la hora actual antes de saludarte y te ahorra tener que volver a escribir las líneas.

Otros comandos útiles que puedas necesitar son:

| Comando                                       | Descripción  |
|---|--|
| <b>Ctrl + W</b>                               | buscar en el documento   |
| <b>Ctrl + W</b> ,<br>luego <b>Control + R</b> | buscar y reemplazar  |
| <b>Ctrl + G</b>                               | mostrar todos los comandos posibles                              |
| <b>Ctrl + Y/V</b>                             | página hacia arriba / abajo                                      |
| <b>Ctrl + C</b>                               | mostrar la posición actual en el archivo y el tamaño del archivo |

## 9.4 El Scripting Básico

Anteriormente en este capítulo tuviste tu primera experiencia de scripting y recibiste una introducción a un script muy básico que ejecuta un comando simple. El script comenzó con la línea shebang, que le dice al Linux que tiene que utilizar el `/bin/bash` (lo que es Bash) para ejecutar un script.

Aparte de ejecutar comandos, hay 3 temas que debes conocer:

Las variables, que contienen información temporal en el script

Las condicionales, que te dejan hacer varias cosas basadas en las pruebas que vas introduciendo

Los Loops, que te permiten hacer lo mismo una y otra vez

#### 9.4.1 Las Variables

Las variables son una parte esencial de cualquier lenguaje de programación. A continuación se muestra un uso muy simple de las variables:

```
#!/bin/bash

ANIMAL="penguin"

echo "My favorite animal is a $ANIMAL"
```

Después de la línea shebang está una directiva para asignar un texto a una variable. El nombre de la variable es `ANIMAL` y el signo de igual asigna la cadena `penguin` (o «pingüino» en español). Piensa en una variable como una caja en la que puedes almacenar cosas. Después de ejecutar esta línea, la caja llamada `ANIMAL` contiene la palabra `penguin`.

Es importante que no haya ningún espacio entre el nombre de la variable, el signo de igual y el elemento que se asignará a la variable. Si le pones espacio, obtendrás un error como «command not found». No es necesario poner la variable en mayúsculas pero es una convención útil para separar las variables de los comandos que se ejecutarán.

A continuación, el script despliega una cadena en la consola. La cadena contiene el nombre de la variable precedido por un signo de dólar. Cuando el intérprete ve el signo de dólar reconoce que va a sustituir el contenido de la variable, lo que se llama interpolación. La salida del script es `My favorite animal is a penguin` (o «My animal favorito es un pingüino» en español.)

Así que recuerda esto: Para asignar una variable, usa el nombre de la variable. Para acceder al contenido de la variable, pon el prefijo del signo de dólar. ¡A continuación, vamos a mostrar una variable a la que se asigna el contenido de otra variable!

```
#!/bin/bash

ANIMAL=penguin

SOMETHING=$ANIMAL

echo "My favorite animal is a $SOMETHING"
```

`ANIMAL` contiene la cadena `penguin` (como no hay espacios, se muestra la sintaxis alternativa sin usar las comillas). A `SOMETHING` se le asigna el contenido de `ANIMAL` (porque a la variable `ANIMAL` le procede el signo de dólar).

Si quieres, puedes asignar una cadena interpolada a una variable. Esto es bastante común en las grandes secuencias de comandos, ya que puedes construir un comando más grande y ejecutarlo!

Otra forma de asignar una variable es utilizar la salida de otro comando como el contenido de la variable incluyendo el comando entre los comillas invertidas:

```
#!/bin/bash

CURRENT_DIRECTORY=`pwd`
```

```
echo "You are in $CURRENT_DIRECTORY"
```

Este patrón a menudo se utiliza para procesar texto. Puedes tomar el texto de una variable o un archivo de entrada y pasarlo por otro comando como `sed` o `awk` para extraer ciertas partes y guardar el resultado en una variable.

Es posible obtener entradas del usuario de su script y asignarlo a una variable mediante el comando `read` (o «leer» en español):

```
#!/bin/bash

echo -n "What is your name? "
read NAME
echo "Hello $NAME!"
```

El comando `read` puede aceptar una cadena directa desde el teclado o como parte de la redirección de comandos tal como aprendiste en el capítulo anterior.

Hay algunas variables especiales además de las establecidas. Puedes pasar argumentos a tu script:

```
#!/bin/bash

echo "Hello $1"
```

El signo de dólar seguido de un número *N* corresponde al argumento *Nth* pasado al script. Si invocas al ejemplo anterior con `./test.sh` el resultado será `Hola Linux`. La variable `$0` contiene el nombre del script.

Después de que un programa se ejecuta, ya sea un binario o un script, devuelve el exit code (o «código de salida» en español) que es un número entero entre 0 y 255. Puedes probarlo con la variable `$?` para ver si el comando anterior se ha completado con éxito.

```
sysadmin@localhost:~$ grep -q root /etc/passwd
sysadmin@localhost:~$ echo $?
0
sysadmin@localhost:~$ grep -q slartibartfast /etc/passwd
sysadmin@localhost:~$ echo $?
1
```

Se utilizó el comando `grep` para buscar una cadena dentro de un archivo con el indicador `-q`, que significa «silencioso» (o «quiet» en inglés). El `grep`, mientras se ejecuta en modo silencioso, devuelve 0, si la cadena se encontró y 1 en el caso contrario. Esta información puede utilizarse en un condicional para realizar una acción basada en la salida de otro comando.

Además puedes establecer el código de salida de tu propio script con el comando `exit`:

```
#!/bin/bash

# Something bad happened!
```

```
exit 1
```

El ejemplo anterior muestra un comentario `#`. Todo lo que viene después de la etiqueta `hash` se ignora, se puede utilizar para ayudar al programador a dejar notas. El comando `exit 1` devuelve el código de salida 1 al invocador. Esto funciona incluso en el shell. Si ejecutas este script desde la línea de comandos y luego introduces `echo $?`, verás que devolverá 1.

Por convención, un código de salida de 0 significa «todo está bien». Cualquier código de salida mayor que 0 significa que ocurrió algún tipo de error, que es específico para el programa. Viste que `grep` utiliza 1 lo que significa que la cadena no se encontró.

#### 9.4.2 Condicionales

Ahora que puedes ver y definir las variables, es hora de hacer que tus propios scripts tengan diferentes funciones basadas en pruebas, llamado *branching* (o «ramificación» español). La instrucción `if` (o «si» en español) es el operador básico para implementar un branching.

La instrucción `if` se ve así:

```
if somecommand; then
    # do this if somecommand has an exit code of 0
fi
```

El siguiente ejemplo ejecutará «somecommand» (en realidad, todo hasta el punto y coma) y si el código de salida es 0, entonces se ejecutará el contenido hasta el cierre `fi`. Usando lo que sabes acerca del `grep`, ahora puedes escribir un script que hace cosas diferentes, basadas en la presencia de una cadena en el archivo de contraseñas:

```
#!/bin/bash

if grep -q root /etc/passwd; then
    echo root is in the password file
else
    echo root is missing from the password file
fi
```

De los ejemplos anteriores podrías recordar que el código de salida del `grep` es 0 si se encuentra la cadena. El ejemplo anterior utiliza esto en una línea para imprimir un mensaje si `root` está en el archivo `passwd`, u otro mensaje si no es así. La diferencia aquí es que en lugar de un `fi` para cerrar, el bloque `if`, hay un `else`. Esto te permite realizar una acción si la condición es verdadera, y otra si la condición es falsa. El bloque `else` siempre debe cerrarse con la palabra clave `fi`.

Otras tareas comunes son buscar la presencia de un archivo o directorio y comparar cadenas y números. Podrías iniciar un archivo de registro si no existe, o comparar el número de líneas en un archivo con la última vez que se ejecutó. El comando `if` es claramente de ayuda aquí, pero ¿qué comando debes usar para hacer la comparación?

El comando `test` te da un acceso fácil los operadores de prueba de comparación y archivos. Por ejemplo:

| Comando                               | Descripción   |
|---------------------------------------|---|
| <code>test -f<br/>/dev/ttyS0</code>   | 0 si el archivo existe  |
| <code>test ! -f<br/>/dev/ttyS0</code> | 0 si el archivo no existe   |
| <code>test -d<br/>/tmp</code>         | 0 si el directorio existe   |
| <code>test -x<br/>`which ls`</code>   | sustituir la ubicación de <code>ls</code> y luego (probar) <code>test</code> , si el usuario puede ejecutar |
| <code>test 1 -eq<br/>1</code>         | 0 si tiene éxito la comparación numérica  |
| <code>test ! 1 -<br/>eq 1</code>      | NO - 0 si la comparación falla  |
| <code>test 1 -ne<br/>1</code>         | Más fácil, ejecutar <code>test</code> (probar) si hay desigualdad numérica                                  |

| Comando                              | Descripción  |
|--------------------------------------|--|
| <code>test "a" = "a"</code>          | 0 si tiene éxito la comparación de cadenas                         |
| <code>test "a" != "a"</code>         | 0 si las cadenas son diferentes                                    |
| <code>test 1 -eq 1 -o 2 -eq 2</code> | <code>-o</code> es OR: cualquiera de las opciones pueden ser igual |
| <code>test 1 -eq 1 -a 2 -eq 2</code> | <code>-a</code> es AND: ambas comparaciones deben ser iguales      |

Es importante tener en cuenta que el `test` ve las comparaciones de números enteros y cadenas de manera distinta. 01 y 1 son iguales por comparación numérica, pero no para comparación de cadenas o strings. Siempre debes tener cuidado y recordar qué tipo de entrada esperas.

Hay muchas más pruebas, como `-gt` para mayor que, formas de comprobar si un archivo es más reciente que los otros y muchos más. Para más información consulta la página [test man](#).

La salida de `test` es bastante largo para un comando que se usa con tanta frecuencia, por lo que tiene un alias llamado `[` (corchete cuadrado izquierdo). Si incluyes tus condiciones en corchetes, es lo mismo que si ejecutaras el `test`. Por lo tanto, estas instrucciones son idénticas.

```
if test -f /tmp/foo; then
if [ -f /tmp/foo]; then
```

Mientras que la última forma es de uso más frecuente, es importante entender que el corchete es un comando en sí que funciona de manera semejante al `test` excepto que requiere el corchete cuadrado de cierre.

La instrucción `if` tiene una forma final que te permite hacer varias comparaciones al mismo tiempo usando `elif` (abreviatura de `elseif`).

```
#!/bin/bash
```

```

if [ "$1" = "hello" ]; then
    echo "hello yourself"
elif [ "$1" = "goodbye" ]; then
    echo "nice to have met you"
    echo "I hope to see you again"
else
    echo "I didn't understand that"
fi

```

El código anterior compara el primer argumento pasado al script. Si es `hello`, se ejecuta el primer bloque. Si no es así, el script de comandos comprueba si es `goodbye` e invoca con el comando `echo` con un diferente mensaje entonces. De lo contrario, se envía un tercer mensaje. Ten en cuenta que la variable `$1` viene entre comillas y se utiliza el operador de comparación de cadenas en lugar de la versión numérica (`-eq`).

Las pruebas `if/elif/else` pueden ser bastante detalladas y complicadas. La instrucción `case` proporciona una forma distinta de hacer las pruebas múltiples más fáciles.

```

#!/bin/bash

case "$1" in
hello|hi)
    echo "hello yourself"
    ;;
goodbye)
    echo "nice to have met you"
    echo "I hope to see you again"
    ;;
*)
    echo "I didn't understand that"
esac

```

La instrucción `case` comienza con la descripción de la expresión que se analiza: `case EXPRESSION in`. La expresión aquí es `$1` entre comillas.

A continuación, cada conjunto de pruebas se ejecutan como una coincidencia de patrón terminada por un paréntesis de cierre. El ejemplo anterior primero busca `hello` o `hi`; múltiples opciones son separadas por la barra vertical `|` que es un operador OR en muchos lenguajes de programación. Después de esto, se ejecutan los comandos si el patrón devuelve verdadero y se terminan con dos signos de punto y coma. El patrón se repite.

El patrón `*` es igual a `else`, ya que coincide con cualquier cosa. El comportamiento de la instrucción `case` es similar a la instrucción `if/elif/else` en que el proceso se detiene tras la primera coincidencia. Si ninguna de las otras opciones coincide, el patrón `*` asegurará que coincida con la última.

Con una sólida comprensión de las condicionales puedes hacer que tus scripts tomen acción sólo si es necesario.

### 9.4.3 Los Loops

Los loops (o «ciclos o bucles» en español) permiten que un código se ejecute repetidas veces. Pueden ser útiles en numerosas situaciones, como cuando quieres ejecutar los mismos comandos sobre cada archivo en un directorio, o repetir alguna acción 100 veces. Hay dos loops principales en los scripts del shell: el loop `for` y el loop `while`.

Los loops `for` se utilizan cuando tienes una colección finita que quieres repetir, como una lista de archivos o una lista de nombres de servidor:

```
#!/bin/bash

SERVERS="servera serverb serverc"

for S in $SERVERS; do
    echo "Doing something to $S"
done
```

Primero, el script establece una variable que contiene una lista de nombres de servidor separada por espacios. La instrucción `for` entonces cicla (realiza «loops») sobre la lista de los servidores, cada vez que establece la variable `S` con el nombre del servidor actual. La elección de la `S` es arbitraria, pero ten en cuenta que la `S` no tiene un signo de dólar, pero en el `$SERVERS` sí lo tiene, lo que muestra que se `$SERVERS` se extenderá a la lista de servidores. La lista no tiene que ser una variable. Este ejemplo muestra dos formas más para pasar una lista.

```
#!/bin/bash

for NAME in Sean Jon Isaac David; do
    echo "Hello $NAME"
done

for S in *; do
    echo "Doing something to $S"
done
```



El primer loop es funcionalmente el mismo que en el ejemplo anterior, excepto que la lista se pasa directamente al loop `for` en lugar de usar una variable. Usar una variable ayuda a que el script sea más claro, ya que una persona fácilmente puede realizar cambios en la variable en lugar de ver el loop.

El segundo loop utiliza comodín `*` que es un *file glob*. El shell expande eso a todos los archivos en el directorio actual.

El otro tipo de loop, un loop `while`, opera en una lista de tamaño desconocido. Su trabajo es seguir ejecutándose y en cada iteración realizar un `test` para ver si se tiene que ejecutar otra vez. Lo puedes ver como «mientras que una condición es verdadera, haz cosas».

```
#!/bin/bash

i=0

while [ $i -lt 10 ]; do
    echo $i
    i=$(( $i + 1 ))
done

echo "Done counting"
```

El ejemplo anterior muestra un loop `while` que cuenta de 0 a 9. Un contador de variables, `i`, arranca en 0. Luego, un loop `while` se ejecuta con un `test` siendo «is `$i` less than 10?» (o «¿es `$i` menor que 10?») ¡Ten en cuenta que el loop `while` utiliza la misma notación que la instrucción `if`.

Dentro del loop `while` el valor actual de `i` es mostrado en pantalla, y luego se le añade 1 a través del comando `$((aritmética))` y se asigna de regreso al `i`. Una vez que `i` llega a 10, la instrucción `while` regresa falso y el proceso continuará después del loop.

**NDG Linux Essentials: Capítulo 10: Compresión de hardware de la computadora.**

## 10.1 Introducción

Una de las muchas ventajas de tener tantas distribuciones distintas de Linux es que algunos de ellas están diseñados para funcionar en plataformas de hardware específicas. De hecho, hay una distribución de Linux diseñada para casi todas las plataformas de hardware modernas.

Cada una de estas plataformas de hardware tiene una gran cantidad de variedad en los componentes de hardware que están disponibles. Además de diferentes tipos de unidades de disco duros, hay muchos diferentes monitores e impresoras. Con la popularidad de los dispositivos USB, tales como dispositivos de almacenamiento USB, cámaras y teléfonos celulares, el número de dispositivos disponibles llega a calcularse en miles.

En algunos casos, esto plantea problemas, ya que estos dispositivos típicamente necesitan algún tipo de software (llamados controladores («drivers» en inglés) o módulos) que les permite comunicarse con el sistema operativo instalado. Los fabricantes de hardware a menudo proporcionan este software, pero típicamente para Microsoft Windows, no para Linux. La mascota de Linux, **Tux**, sin embargo está empezando a aparecer más a menudo en los productos de hardware, indicando el soporte de Linux.

Además del apoyo de los proveedores, hay una gran cantidad de apoyo de la comunidad que se esfuerza por proporcionar controladores para los sistemas Linux. Aunque no todo el hardware tiene los controladores necesarios, hay una buena cantidad que sí los tiene, lo que supone un reto para los usuarios y administradores Linux para encontrar los controladores correctos o elegir el hardware que tiene cierto nivel de soporte en Linux.

En este capítulo aprenderás acerca de los dispositivos de core hardware, incluyendo la manera de utilizar los comandos de Linux para mostrar la información vital de hardware del dispositivo.

## 10.2 Los Procesadores

La *Unidad Central de Procesamiento* (CPU, «*Central Processing Unit*» en inglés o procesador) es uno de los componentes más importantes de hardware en una computadora. Realiza la toma de decisiones, así como los cálculos que deben realizarse para ejecutar correctamente un sistema operativo. El procesador es esencialmente un chip de computadora.

El procesador está conectado con otro hardware a través de una *placa base* (o «*motherboard*» en inglés), también conocida como la placa del sistema. Las placas base están diseñadas para funcionar con determinados tipos de procesadores.

Si un sistema tiene más de un procesador, el sistema se denomina *multiprocesador*. Si se combina más de un procesador en un único chip del procesador, entonces se llama multi-core (o «*multi-núcleo*» en español).

Aunque el apoyo está disponible para más tipos de procesadores en Linux que en cualquier otro sistema operativo, principalmente hay dos tipos de procesadores utilizados en las computadoras de escritorio y en los servidores: x86 y x86\_64. En un x86, el sistema procesa los datos de 32 bits a la vez; en un x86\_64 el sistema procesa datos de 64 bits a la vez. Un sistema x86\_64 también es capaz de procesar los datos de 32 bits a la vez en un modo compatible con las versiones anteriores. Una de las ventajas principales de un sistema de 64 bits es que el sistema es capaz de trabajar con más memoria.

La familia de procesadores de x86 fue creada por Intel en 1978 con el lanzamiento del procesador 8086. Desde entonces, Intel ha producido muchos otros procesadores que son las mejoras del 8086 original; se conocen genéricamente como los procesadores x86. Estos procesadores incluyen el 80386 (también conocido como el i386), 80486 (i486), Pentium (i586) y la serie del Pentium Pro (i686). Además de Intel, hay otras empresas como **AMD** y Cyrix que también han producido procesadores compatibles con los x86. Si bien Linux es capaz de soportar procesadores de la generación del i386, muchas distribuciones limitan su soporte al i686 o posteriores.

La familia de los procesadores x86\_64, incluyendo los procesadores de 64 bits de Intel y AMD, ha estado en la producción desde alrededor del año 2000. Como resultado, la mayoría de los procesadores modernos construidos hoy en día son de x86\_64. Mientras que el hardware ha estado disponible por más de una década hasta ahora, el software de soporte para esta familia de procesadores se ha estado desarrollando mucho más lento. Incluso en el 2013 ya había muchos paquetes de software que estaban disponibles para la arquitectura x86, pero no para x86\_64.

Puedes ver a qué familia pertenece tu CPU usando el comando `arch`:

```
sysadmin@localhost:~$ arch
x86_64
sysadmin@localhost:~$
```

Otro comando que puedes utilizar para identificar el tipo de CPU en el sistema es el comando `lscpu`:

```
sysadmin@localhost:~$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:             Little Endian
CPU(s):                 4
On-line CPU(s) list:   0-3
Thread(s) per core:     1
Core(s) per socket:     4
Socket(s):              1
```

```

NUMA node(s):      1
Vendor ID:         GenuineIntel
CPU family:        6
Model:             44
Stepping:          2
CPU MHz:           2394.000
BogoMIPS:          4788.00
Virtualization:    VT-x
Hypervisor vendor: VMware
Virtualization type: full
L1d cache:         32K
L1i cache:         32K
L2 cache:          256K
L3 cache:          12288K
NUMA node0 CPU(s): 0-3
sysadmin@localhost:~$

```

La primera línea de esta salida muestra que se está utilizando la CPU en modo de 32 bits, ya que la arquitectura reportada es x86\_64. La segunda línea de la salida muestra que la CPU es capaz de operar en modo ya sea de 32 o 64 bits, por lo tanto realmente es un CPU de 64 bits.

La manera más detallada de obtener la información acerca de tu CPU es visualizando el archivo `/proc/cpuinfo` con el comando `cat`:

```

sysadmin@localhost:~$ cat /proc/cpuinfo

processor      : 0
vendor_id     : GenuineIntel
cpu family    : 6
model         : 44
model name    : Intel(R) Xeon(R) CPU           E5620  @ 2.40GHz
stepping      : 2
microcode     : 0x15
cpu MHz       : 2394.000
cache size    : 12288 KB
physical id   : 0
siblings      : 4
core id       : 0
cpu cores     : 4
apicid        : 0
initial apicid : 0

```

```

fpu           : yes
fpu_exception : yes
cpuid level   : 11
wp            : yes
flags         : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
pat pse36 clflush dts mmx fxsr sse sse2 ss ht syscall nx rdtscp lm constant_ts
arch_perfmon pebs bts nopl xtopology tsc_reliable nonstop_tsc aperfmperf pni pcl mulqdq vmx ssse3 cx16 sse4_1 sse4_2 x2apic popcn
t aes hypervisor lahf_lm ida arat dtherm tpr_shadow vnmi ept vpid

```

Mientras que la gran parte de la salida del comando `lscpu` y del contenido del archivo `/proc/cpuinfo` parece ser la misma, una de las ventajas de visualizar el archivo `/proc/cpuinfo` es que aparecen `flags` (o «indicadores» en español) de la CPU. Los `flags` de una CPU son un componentes muy importantes, ya que señalan qué características soporta la CPU y las capacidades de la CPU.

Por ejemplo, la salida del ejemplo anterior contiene el flag `lm` (long mode, o «modo largo» en español), indicando que esta CPU es de 64-bit. También hay flags que indican si la CPU es capaz de soportar máquinas virtuales (la capacidad de tener varios sistemas operativos en un único equipo).

### 10.3 Tarjetas Madre y los Buses

La tarjeta madre o «motherboard» en inglés, es el tablero principal del hardware en la computadora a través de la cuál se conectan la CPU, la Memoria de Acceso Aleatorio (RAM) y otros componentes. Algunos dispositivos se conectan directamente a la tarjeta madre, mientras que otros dispositivos se conectan a la tarjeta madre mediante un *bus*.

#### 10.3.1 dmidecode

La tarjeta madre de muchas computadoras contiene lo que se conoce como *Basic Input and Output System (BIOS)* (o «*BIOS de Administración del Sistema*» en español). *System Management BIOS (SMBIOS)* (o «*El Sistema de Gestión o Administración de BIOS*» en español) es el estándar que define las estructuras de datos y cómo se comunica la información acerca del hardware de la computadora. El comando `dmidecode` es capaz de leer y mostrar la información del SMBIOS.

Un administrador puede utilizar el comando `dmidecode` para ver los dispositivos conectados directamente a la tarjeta madre. Hay una gran cantidad de información proporcionada por la salida de este comando.

Los ejemplos siguientes te proporcionan algunas ideas de lo que se puede saber de la salida del comando `dmidecode`. Este comando no está disponible dentro del entorno de la máquina virtual de este curso.

En el primer ejemplo, se puede ver que el BIOS soporta el arranque directamente desde el CD-ROM. Esto es importante ya que los sistemas operativos a menudo se instalan arrancando directamente desde el CD de instalación:

```

# dmidecode 2.11
SMBIOS 2.4 present.
364 structures occupying 16040 bytes.
Table at 0x000E0010

Handle 0x0000, DMI type 0, 24 bytes

```

**BIOS Information**

Vendor: Phoenix Technologies LTD  
 Version: 6.00  
 Release Date: 06/22/2012  
 Address: 0xEA0C0  
 Runtime Size: 89920 bytes  
 ROM Size: 64 kB

**Characteristics:**

ISA is supported  
 PCI is supported  
 PC Card (PCMCIA) is supported  
 PNP is supported  
 APM is supported  
 BIOS is upgradeable  
 BIOS shadowing is allowed  
 ESCD support is available  
 Boot from CD is supported

--More--

En el siguiente ejemplo puedes ver que un total de 2048 (aproximadamente 2GB) de RAM está instalado en el sistema:

Socket Designation: RAM socket #0

Bank Connections: None

Current Speed: Unknown

Type: EDO DIMM

**Installed Size: 2048 MB** (Single-bank Connection)

Enabled Size: 2048 MB (Single-bank Connection)

Error Status: OK

**10.3.2 Memoria de Acceso Aleatorio (RAM)**

La tarjeta madre normalmente tiene ranuras donde la *Memoria de Acceso Aleatorio* (RAM o «*Random Access Memory*» en inglés) puede conectarse al sistema. Los sistemas de arquitectura de 32 bits pueden utilizar hasta 4 gigabytes (GB) de RAM, mientras que las arquitecturas de 64 bits son capaces de abordar y usar mucha más RAM.

En algunos casos, la RAM que tiene tu sistema podría no ser suficiente para manejar todos los requisitos del sistema operativo. Cada programa necesita almacenar datos en la RAM y los programas mismos se cargan en la RAM cuando se ejecutan.

Para evitar que el sistema falle por falta de RAM, se utiliza una RAM *virtual* (o *espacio de intercambio* «*swap space*» en inglés). La RAM virtual es un espacio en el disco duro que se utiliza para almacenar temporalmente los datos de RAM cuando el sistema se está quedando sin la RAM. Los datos que se almacenan en la RAM y que no se han utilizado recientemente se copian al disco duro los programas utilizados recientemente puedan utilizar la RAM. Si es necesario, los datos intercambiados pueden almacenarse en la RAM en un momento posterior.

Para ver la cantidad de la RAM en tu sistema, incluyendo la RAM virtual, ejecuta el comando `free`. El comando `free` tiene la opción `-m` para forzar la salida a ser redondeada al megabyte más cercano y una opción `-g` para forzar la salida a ser redondeada al gigabyte más cercano:

```
sysadmin@localhost:~$ free -m
```

| total | used | free | shared | buffers | cached |
|-------|------|------|--------|---------|--------|
|-------|------|------|--------|---------|--------|

```
Mem:      1894      356      1537      0      25      177
-/+ buffers/cache:      153      1741
Swap:      4063      0      4063
sysadmin@localhost:~$
```

La salida al ejecutar este comando `free` muestra que el sistema fue ejecutado en un sistema que tiene un total de 1.894 megabytes y está utilizando actualmente 356 megabytes.

La cantidad de swap aparece ser aproximadamente 4 gigabytes, aunque nada de esto parece estar en uso. Esto tiene sentido porque la gran parte de la RAM física está libre, así que en este momento no se necesita utilizar la RAM virtual.

### 10.3.3 Los Dispositivos Periféricos

La tarjeta madre tiene buses que permiten conectar múltiples dispositivos al sistema, incluyendo la Interconexión de los Componentes Periféricos (PCI) y Bus Serie Universal (USB). La tarjeta madre tiene también conectores para monitores, teclados y ratones.

Para ver todos los dispositivos conectados por un bus PCI ejecuta el comando `lspci`. El siguiente ejemplo muestra una salida de este comando. Como se puede ver a continuación en las secciones destacadas, este sistema tiene un controlador VGA (conector de un monitor), un controlador de almacenamiento SCSI (un tipo de disco duro) y un controlador de Ethernet (un conector de red):

Los gráficos a continuación ofrecen ejemplos de uso del comando `lspci`. Este comando no está disponible dentro del entorno de la máquina virtual de este curso.

```
sysadmin@localhost:~$ lspci
00:00.0 Host bridge: Intel Corporation 440BX/ZX/DX - 82443BX/ZX/DX Host bridge (rev 01)
00:01.0 PCI bridge: Intel Corporation 440BX/ZX/DX - 82443BX/ZX/DX AGP bridge (rev 01)
00:07.0 ISA bridge: Intel Corporation 82371AB/EB/MB PIIX4 ISA (rev 08)
00:07.1 IDE interface: Intel Corporation 82371AB/EB/MB PIIX4 IDE (rev 01)
00:07.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 08)
00:07.7 System peripheral: VMware Virtual Machine Communication Interface (rev 10)
00:0f.0 VGA compatible controller: VMware SVGA II Adapter
03:00.0 Serial Attached SCSI controller: VMware PVSCSI SCSI Controller (rev 02)
0b:00.0 Ethernet controller: VMware VMXNET3 Ethernet Controller (rev 01)
```

Ejecutar el comando `lspci` con la opción `-nn` muestra un identificador numérico para cada dispositivo, así como la descripción del texto original:

```
sysadmin@localhost:~$ lspci -nn
00:00.0 Host bridge [0600]: Intel Corporation 440BX/ZX/DX - 82443BX/ZX/DX Host bridge [8086:7190] (rev 01)
00:01.0 PCI bridge [0604]: Intel Corporation 440BX/ZX/DX - 82443BX/ZX/DX AGP bridge [8086:7191] (rev 01)
00:07.0 ISA bridge [0601]: Intel Corporation 82371AB/EB/MB PIIX4 ISA [8086:7110] (rev 08)
00:07.1 IDE interface [0101]: Intel Corporation 82371AB/EB/MB PIIX4 IDE [8086:7111] (rev 01)
00:07.3 Bridge [0680]: Intel Corporation 82371AB/EB/MB PIIX4 ACPI [8086:7113] (rev 08)
00:07.7 System peripheral [0880]: VMware Virtual Machine Communication Interface [15ad:0740] (rev 10)
```

```
00:0f.0 VGA compatible controller [0300]: VMware SVGA II Adapter [15ad:0405]
03:00.0 Serial Attached SCSI controller [0107]: VMware PVSCSI SCSI Controller
[15ad:07c0] (rev 02)
0b:00.0 Ethernet controller [0200]: VMware VMXNET3 Ethernet Controller
[15ad:07b0] (rev 01)
```

La selección resaltada, [15ad:07b0], se refiere a la sección [vendor:device] (o «vendedor:dispositivo» en español).

Utilizar la información [vendor:device] puede ser útil para mostrar la información detallada acerca de un dispositivo específico. Utilizar al la opción `-d vendor:device`, puedes seleccionar ver la información sobre un sólo dispositivo.

También puedes ver información más detallada mediante cualquiera de las dos opciones, la `-v`, `-vv` o la `-vvv`. Cuántos más caracteres `v`, más detallada será la salida. Por ejemplo:

```
sysadmin@localhost:~$ lspci -d 15ad:07b0 -vvv
0b:00.0 Ethernet controller: VMware VMXNET3 Ethernet Controller (rev 01)
    Subsystem: VMware VMXNET3 Ethernet Controller
    Physical Slot: 192
    Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Step
ping- SERR- FastB2B- DisINTx+
    Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort-
<MAbort- >SERR- <PERR- INTx-
    Latency: 0, Cache Line Size: 32 bytes
    Interrupt: pin A routed to IRQ 19
    Region 0: Memory at fd4fb000 (32-bit, non-prefetchable) [size=4K]
    Region 1: Memory at fd4fc000 (32-bit, non-prefetchable) [size=4K]
    Region 2: Memory at fd4fe000 (32-bit, non-prefetchable) [size=8K]
    Region 3: I/O ports at 5000 [size=16]
    [virtual] Expansion ROM at fd400000 [disabled] [size=64K]
    Capabilities: <access denied>
    Kernel driver in use: vmxnet3
    Kernel modules: vmxnet3
sysadmin@localhost:~$
```

El comando `lspci` muestra información detallada sobre los dispositivos conectados al sistema a través del bus PCI. Esta información puede ser útil para determinar si el dispositivo es compatible con el sistema, tal como se indica por un *Kernel driver* o un *Kernel module* en uso, como se muestra en el último par de líneas de la salida anterior.

#### 10.3.4 Los Dispositivos de Bus Serie Universal (USB)

Mientras que el bus PCI se utiliza para muchos dispositivos internos tales como las tarjetas de sonido y red, muchos dispositivos externos (o *periféricos*) están conectados a la computadora vía USB. Los dispositivos conectados internamente son generalmente de *cold-plug* (o «conectables en frío» en español), lo que significa que el sistema debe ser apagado para conectar o desconectar un dispositivo. Los dispositivos USB son *hot-plug* (o «conectables en caliente» en español), lo que significa se conectan o desconectan mientras el sistema está funcionando.

**Nota:** Los gráficos a continuación ofrecen ejemplos de uso del comando `lsusb`. Este comando no está disponible dentro del entorno de la máquina virtual de este curso.

Para mostrar los dispositivos conectados al sistema vía USB, ejecuta el comando `lsusb`:

```
sysadmin@localhost:~$ lsusb
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
sysadmin@localhost:~$
```

La opción detallada, `-v`, del comando `lsusb` muestra una gran cantidad de detalles acerca de cada dispositivo:

```
sysadmin@localhost:~$ lsusb -v

Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Couldn't open device, some information will be missing
Device Descriptor:
  bLength                18
  bDescriptorType         1
  bcdUSB                  1.10
  bDeviceClass             9 Hub
  bDeviceSubClass          0 Unused
  bDeviceProtocol          0 Full speed (or root) hub
  bMaxPacketSize0          64
  idVendor                 0x1d6b Linux Foundation
  idProduct                0x0001 1.1 Linux Foundation
  bcDevice                 2.06
```



```
iManufacturer      3
iProduct           2
iSerial            1
...
```

#### 10.4 La Capa de Abstracción de Hardware

HAL o «Hardware Abstraction Layer» en inglés, es la *Capa de Abstracción de Hardware*. El demonio o programa vigilante (o «daemon» en inglés) de la HAL es `hald`, un proceso que recoge información sobre todos los dispositivos conectados al sistema. Cuando se producen eventos que cambian el estado de los dispositivos conectados, tales como un dispositivo USB es conectado al sistema, el `hald` emite esta nueva información a los procesos que se hayan registrado para ser notificados sobre nuevos eventos.

**Nota:** El gráfico siguiente proporciona un ejemplo de uso del comando `lshal`. Este comando no está disponible dentro del entorno de la máquina virtual de este curso.

El comando `lshal` te permite ver los dispositivos detectados por HAL. Este comando produce una gran cantidad de salidas; a continuación se ofrece una pequeña muestra:

```

sysadmin@localhost:~
File Edit View Search Terminal Help
[sysadmin@localhost ~]$ lshal | more

Dumping 96 device(s) from the Global Device List:
-----
udi = '/org/freedesktop/Hal/devices/computer'
  info.addons = {'hald-addon-acpi'} (string list)
  info.callouts.add = {'hal-storage-cleanup-all-mountpoints'} (string list)
  info.interfaces = {'org.freedesktop.Hal.Device.SystemPowerManagement'} (string
list)
  info.product = 'Computer' (string)
  info.subsystem = 'unknown' (string)
  info.udi = '/org/freedesktop/Hal/devices/computer' (string)
  org.freedesktop.Hal.Device.SystemPowerManagement.method_argnames = {'num_secon
ds_to_sleep', 'num_seconds_to_sleep', '', '', '', 'enable_power_save'} (string l
ist)
  org.freedesktop.Hal.Device.SystemPowerManagement.method_execpaths = {'hal-syst
em-power-suspend', 'hal-system-power-suspend-hybrid', 'hal-system-power-hibernat
e', 'hal-system-power-shutdown', 'hal-system-power-reboot', 'hal-system-power-se
t-power-save'} (string list)
  org.freedesktop.Hal.Device.SystemPowerManagement.method_names = {'Suspend', 'S
uspendHybrid', 'Hibernate', 'Shutdown', 'Reboot', 'SetPowerSave'} (string list)
  org.freedesktop.Hal.Device.SystemPowerManagement.method_signatures = {'i', 'i'
, '', '', '', 'b'} (string list)
  org.freedesktop.Hal.version = '0.5.14' (string)

```

## 10.5 Los Dispositivos de Disco

Los *Dispositivos de Disco* (también conocidos como discos duros) se pueden conectar al sistema de varias maneras; el controlador puede integrarse a la tarjeta madre, a una tarjeta PCI (Interconexión de Componente Periférico) o a un dispositivo USB.

Los discos duros se dividen en *particiones*. Una partición es una división lógica de un disco duro, diseñada para tomar una gran cantidad de espacio de almacenamiento disponible y dividirlo en «trozos» más pequeños. Si bien en Microsoft Windows es común tener una única partición para cada disco duro, en las distribuciones de Linux lo común es tener varias particiones por disco duro.

Algunos discos duros hacen uso de una partición llamada **Registro de Arranque Maestro** (MBR o «Master Boot Record» en inglés), mientras que otros hacen uso de un tipo de partición llamada **Tabla de Particiones GUID** (GPT o «GUID Partitioning Table» en inglés). El tipo MBR de la partición se ha utilizado desde los días tempranos de la computadora personal (PC o Personal Computer) y el tipo GPT ha estado disponible desde el año 2000.

Un viejo término usado para describir un disco duro interno es «disco fijo», ya que el disco es fijo (no extraíble). Este término dio lugar a varios nombres de comando: los comandos `fdisk`, `cfdisk` y `sfdisk` son herramientas para trabajar con las particiones discos MBR.

Los discos GPT usan un tipo de particionado más nuevo, que permite al usuario dividir el disco en más particiones de lo que soporta una MBR. La GPT también permite tener particiones que pueden ser más grandes que dos terabytes (MBR no lo permite). Las herramientas para gestionar los discos GPT se llaman de manera similar a las contrapartes de `fdisk`: `gdisk`, `cgdisk` y `sgdisk`.

También existe una familia de herramientas que intenta apoyar ambos tipos de disco MBR y GPT. Este conjunto de herramientas incluye al comando `parted` y la herramienta gráfica `gparted`.

Las unidades de disco duro están asociadas a los nombres de archivos (llamados archivos de dispositivo) que se almacenan en el directorio `/dev`. Diferentes tipos de unidades de disco duros reciben nombres ligeramente diferentes: `hd` para los discos duros IDE (Intelligent Drive Electronics o «Unidad Electrónica Inteligente» en español) y `sd` para USB, SATA (Serial Advanced Technology Attachment o «Aditamento de Tecnología Serial Avanzada» en español) y los discos duros SCSI (Small Computer System **Interface** o «Interfaz Estándar de Equipos Pequeños» en español).

A cada disco se le asigna una letra, por ejemplo, el primer disco duro IDE tendría un nombre de archivo de dispositivo `/dev/hda` y el segundo disco duro IDE se asociaría al archivo de dispositivo `/dev/hdb`.

Las particiones reciben números únicos para cada dispositivo. Por ejemplo, si un disco duro USB tenía dos particiones, éstas pueden asociarse a los archivos de dispositivo `/dev/sda1` y `/dev/sda2`.

En la salida siguiente puedes ver que este sistema tiene tres dispositivos `sd`: `/dev/sda`, `/dev/sdb` y `/dev/sdc`. También puedes ver que hay dos particiones en el primer dispositivo (como lo demuestran los archivos `/dev/sda1` y `/dev/sda2`) y una partición en el segundo dispositivo (según lo visualiza el archivo `/dev/sdb1`):

```
root@localhost:~$ ls /dev/sd*
/dev/sda  /dev/sda1  /dev/sda2  /dev/sdb  /dev/sdb1  /dev/sdc
root@localhost:~$
```

En el ejemplo siguiente se utiliza el comando `fdisk` para mostrar la información de la partición en el primer dispositivo de `sd`.

**Nota:** El siguiente comando requiere acceso root

```
root@localhost:~# fdisk -l /dev/sda
Disk /dev/sda: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders, total 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000571a2
```

| Device    | Boot | Start    | End      | Blocks   | Id | System               |
|-----------|------|----------|----------|----------|----|----------------------|
| /dev/sda1 | *    | 2048     | 39845887 | 19921920 | 83 | Linux                |
| /dev/sda2 |      | 39847934 | 41940991 | 1046529  | 5  | Extended             |
| /dev/sda5 |      | 39847936 | 41940991 | 1046528  | 82 | Linux swap / Solaris |

```
root@localhost:~#
```

La creación y modificación de particiones está fuera del alcance de este curso.

## 10.6 Los Discos Ópticos

Los discos ópticos, referidos a menudo como CD-ROM, DVD o Blue-Ray son medios de almacenamiento extraíbles. Mientras que algunos dispositivos usados con discos ópticos son de sólo lectura, otros pueden ser grabados (escritos), cuando se utiliza un tipo de disco grabable. Hay varios estándares para los discos grabables y regrabables, como CD-R, CD+R, DVD+RW y DVD-RW. Estos estándares de soporte físico van más allá del alcance del plan de estudios.

La ubicación de estos discos extraíbles en el sistema de archivos es una consideración importante para un administrador de Linux. Las distribuciones modernas a menudo montan los discos bajo la carpeta `/media`, mientras que las distribuciones antiguas suelen montarlos en la carpeta `/mnt`.

Al insertar los discos, la mayoría de las interfaces GUI piden al usuario que tome una acción, así como abrir el contenido del disco en un explorador de archivos o iniciar un programa de multimedia. Cuando el usuario termina de usar el disco, conviene expulsarlo mediante el menú o con el comando `eject` (o «expulsar» en español). Mientras que presionar el botón de **expulsar** se abrirá la bandeja de disco, algunos programas no se darán cuenta que el disco ya no está en el sistema de archivos.

## 10.7 Dispositivos de Visualización de Video

Para visualizar un video (salida al monitor) la computadora debe tener un dispositivo de visualización de vídeo (también conocido como la *tarjeta de video*) y un monitor. Los dispositivos de visualización de video a menudo vienen unidos directamente a la tarjeta madre, aunque también pueden ser conectados a través de las ranuras de bus PCI en la tarjeta madre.

Lamentablemente, desde los primeros días de la PC, los principales proveedores no han aprobado estándares de video, por lo que cada dispositivo de visualización de video generalmente requiere un controlador propietario suministrado por el proveedor. Los drivers o controladores son programas de software que permiten al sistema operativo comunicarse con el dispositivo.

Los drivers deben estar escritos para el sistema operativo específico, algo que se hace comúnmente para Microsoft Windows, pero no siempre para Linux. Afortunadamente, los tres proveedores de visualización de video más grande ahora proporcionan al menos cierto nivel de soporte para Linux.

Hay dos tipos de cables de vídeo de uso general: el cable analógico de 15 pines **Video Graphics Array (VGA)** y el de 29 pines **Digital Visual Interface (DVI)**.

Para que los monitores trabajen correctamente con las tarjetas de video, deben ser capaces de soportar la misma resolución que la tarjeta de video. Normalmente, el software de la tarjeta de video (comúnmente el servidor X.org) normalmente será capaz de detectar automáticamente la máxima resolución que la tarjeta de vídeo y el monitor pueden soportar y establecer la resolución de pantalla a ese valor.

Las herramientas gráficas normalmente sirven para cambiar tu resolución, así como el número máximo de colores que se pueden mostrar (conocido como la *profundidad de color*) con tu distribución de Linux. Para las distribuciones que utilizan el servidor X.org, se puede utilizar el archivo `/etc/X11/xorg.conf` para cambiar la resolución, profundidad de color y otros ajustes.

## 10.8 Gestionar los Dispositivos

Para poder utilizar un dispositivo en Linux puede haber varios tipos de software que se requieren. El primero es el software de driver. El driver puede compilarse como parte del kernel de Linux, cargado al kernel como un módulo o cargado por un comando de usuario o una aplicación. La mayoría de los dispositivos tienen el driver incorporado en el kernel o lo tienen cargado al kernel, ya que el driver puede requerir una clase de acceso de nivel bajo que tiene el kernel con los dispositivos.

Los dispositivos externos, como las impresoras y los escáneres normalmente tienen sus drivers cargados por una aplicación y estos drivers a su vez se comunican a través del dispositivo vía el kernel por una interfaz como USB.

Para activar los dispositivos en Linux con éxito, es mejor consultar la distribución de Linux para ver si el dispositivo está certificado para trabajar con esa distribución. Las distribuciones comerciales como Red Hat y SUSE tienen páginas web con una lista de hardware certificado o aprobado para trabajar con su software.

Consejos adicionales para conectar tus dispositivos de manera exitosa: evitar dispositivos nuevos o altamente especializados y consultar con el proveedor del dispositivo para ver si soportan Linux antes de hacer cualquier compra.

## 10.9 Fuentes de Poder

Las fuentes de poder son los dispositivos que convierten la corriente alterna (120v, 240v) a corriente directa la cual la computadora utiliza en varios voltajes (3.3v, 5v, 12v, etc.). Las fuentes de poder generalmente no son programables, sin embargo su funcionamiento tiene un impacto importante en el resto del sistema.

Aunque no son supresores de picos, estos dispositivos a menudo protegen la computadora de las fluctuaciones en el voltaje que provienen del origen. Es aconsejable que el administrador de red elija una fuente de poder basada en la calidad más que en el precio, ya que una falla de la fuente de poder puede resultar en la destrucción de la computadora.

## NDG Linux Essentials: Capítulo 12: Configuración de la red.

### 12.1 Introducción

Tener acceso a la red es una característica clave de la mayoría de los sistemas Linux. Los usuarios quieren navegar por la red, enviar e recibir correo electrónico y intercambiar archivos con otros usuarios.

Normalmente los programas que realizan estas funciones (navegadores, clientes de correo electrónico, etc.) son bastante fáciles de usar. Sin embargo, cuentan con una característica importante: la capacidad de que tu computadora se comuniquen con otro equipo. Para tener esta comunicación, necesitas saber cómo configurar la red de tu sistema.

Linux te proporciona varias herramientas tanto para configurar tu red, así como para supervisar su rendimiento. En este capítulo aprenderás a utilizar ambas herramientas basadas en GUI, así como las herramientas de línea de comandos.

**¿Principales ventajas de ser un profesional de Linux?** Horarios de trabajo flexibles, teletrabajo, incrementos de salario por arriba de la norma de la compañía, bonos más grandes.

### 12.2 La Terminología Básica de la Red

Antes de configurar una red o acceder a una red existente, es importante conocer algunos términos que están relacionados con las redes. Esta sección explora los términos que debes tener en cuenta. Algunos de los términos son básicos y probablemente ya los conoces, sin embargo otros son más avanzados.

**Host:** Un *host* es básicamente una computadora. Sin embargo, muchas personas tienen una idea más limitada de lo que es una computadora (como una computadora de escritorio o una portátil). En realidad, muchos otros dispositivos también son computadoras, tales como teléfonos celulares, reproductores de música digitales y muchas televisiones modernas. En términos de redes, un host es cualquier dispositivo que se comunica con otro dispositivo.

**Red:** Una *red* es una colección de dos o más hosts (computadoras) que son capaces de comunicarse entre sí. Esta comunicación puede ser a través de una conexión cableada o inalámbrica.

**Internet:** El *Internet* es un ejemplo de una red. Consiste de una red accesible públicamente que conecta millones de hosts en todo el mundo. Mucha gente utiliza el Internet para navegar por páginas web y enviar y recibir correo electrónico, pero el Internet tiene muchas funciones adicionales además de estas actividades.

**Wi-Fi:** El término *Wi-Fi* se refiere a las redes inalámbricas.

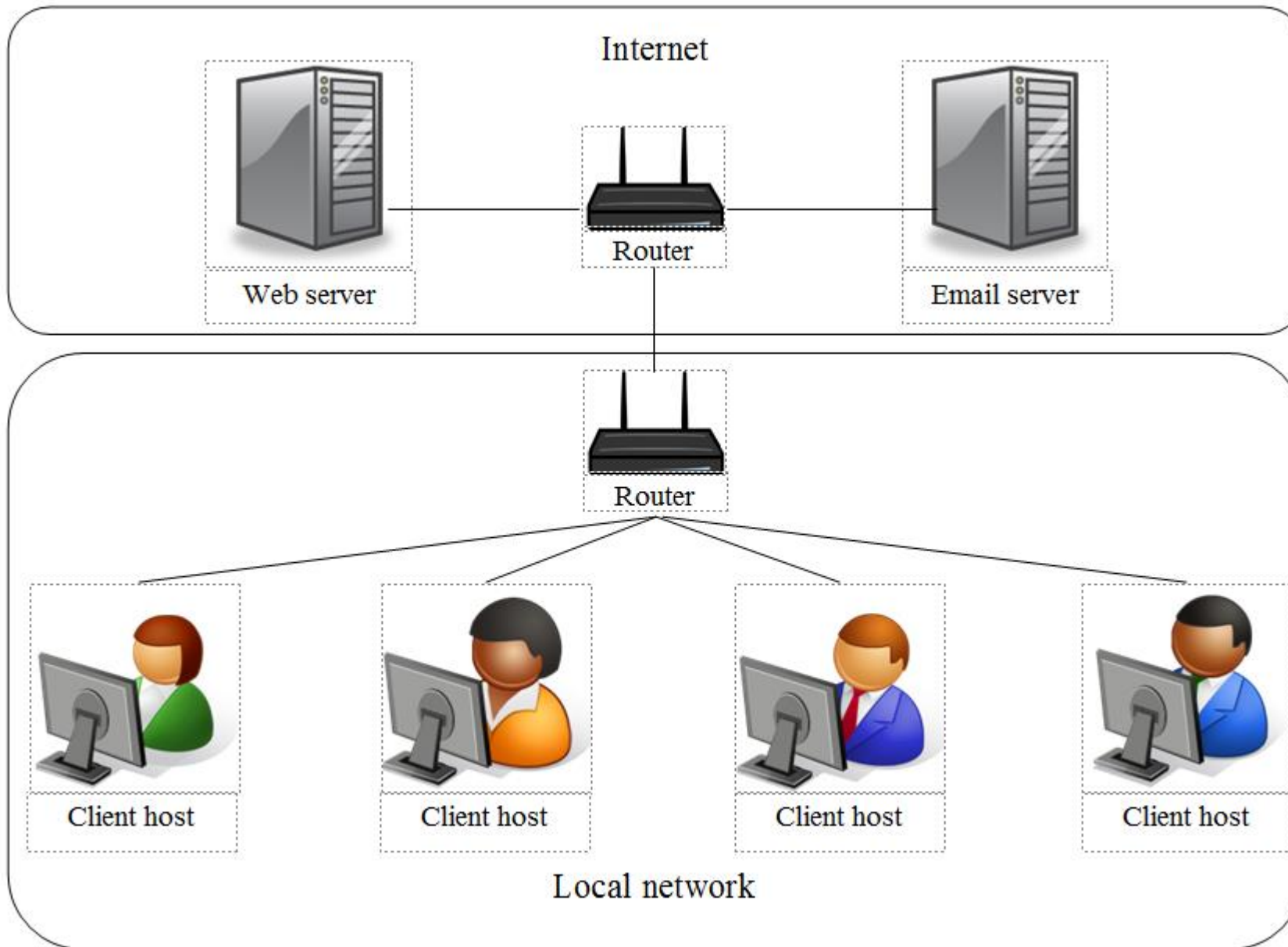
**Servidor:** Un host que proporciona un servicio a otro host o cliente se denomina *servidor*. Por ejemplo, un servidor web almacena, procesa y entrega páginas web. Un servidor de correo recibe correo entrante y entrega correo saliente.

**Servicio:** Una característica que se presta desde un host es un *servicio*. Un ejemplo de un servicio sería cuando un host proporciona páginas web a otro host.

**Cliente:** Un *cliente* es un host que está accediendo a un servidor. Cuando se trabaja en un equipo navegando por Internet, eres un host cliente.

**Router:** También llamado *gateway* (o «puerta de enlace» en español), un router o «enrutador» en español, es una máquina que conecta hosts de una red a otra red. Por ejemplo, si trabajas en un entorno de oficina, las computadoras dentro de la empresa pueden todos comunicarse vía la *red local* (o «local network» en inglés) creada por los administradores. Para acceder al Internet, los equipos tienen que comunicarse con un router que se utiliza para reenviar las comunicaciones de red al Internet. Normalmente cuando te comunicas en una red amplia (como el Internet), hay varios routers que se utilizan antes de que tu comunicación llegue a su destino final.

El siguiente diagrama proporciona una referencia visual para varios de los términos mencionados:



### 12.3 Terminología de las Funciones de Redes

Además de los términos de redes mencionados en la sección anterior, hay algunos términos adicionales que debes conocer. Estos términos se centran más en los diferentes tipos de servicios de redes que se utilizan, así como algunas de las técnicas que se utilizan para la comunicación entre las máquinas.

**Paquete de red:** Un *paquete de red* se utiliza para enviar la comunicación de red entre los hosts. Rompiendo la comunicación en trozos más pequeños (paquetes), el método de entrega de datos es mucho más eficiente.

**Dirección IP:** Una *Dirección de Protocolo de Internet (IP)* o «*Internet Protocol*» en inglés es un número único asignado a un host en una red. Los hosts utilizan estos números para «dirigir» una comunicación de red. Sobre las direcciones IP hablaremos más adelante en este capítulo.

**Máscara de red:** También llamada una *netmask* o *máscara*, una *máscara de red* es un sistema numérico que puede utilizarse para definir cuáles de las direcciones IP se consideran dentro de una única red. Debido a cómo los routers desempeñan sus funciones, las redes tienen que ser definidas claramente.

**Nombre de host:** Cada host en una red puede tener su propio *nombre de host*. Esto le facilita a los usuarios dirigir los paquetes de red a otro host, ya que para los usuarios es más fácil recordar nombres que números. Los nombres de host se traducen a direcciones IP antes de enviar el paquete de red en la red.

**DHCP:** A los hosts se le puede asignar nombres de hosts, direcciones IP y otra información relacionada con la red por un *Servidor DHCP (Dynamic Host Configuration Protocol* o «*Protocolo de Configuración Dinámica de Host*» en español). En el mundo de la informática, un protocolo es un conjunto de reglas bien definido. DHCP define cómo se asigna la información de red a los clientes host y el servidor DHCP es la máquina que proporciona esta información. Mientras que la configuración de un servidor DHCP está fuera del alcance de este capítulo, verás cómo configurar una máquina de cliente DHCP más adelante en este capítulo.

**DNS:** Como ya hemos mencionado anteriormente, los nombres de host se traducen a direcciones IP antes de enviar el paquete en la red. Esto significa que tu host necesita conocer la dirección IP de todos los otros hosts con los cuáles te comunicas. Cuando trabajas en una red amplia (como Internet), esto puede plantear un desafío ya que hay muchos hosts. Un *Servidor DNS (Domain Name Server)* proporciona el servicio de traducción de los nombres de dominio en direcciones IP. Mientras que la configuración de un servidor DNS está fuera del alcance de este capítulo, verás cómo configurar una máquina de cliente DNS más adelante en este capítulo.

**Ethernet:** En un entorno de red por cable, *Ethernet* es la forma más común para conectar físicamente los hosts en una red. Los cables de Ethernet están conectados a las tarjetas de red que soportan las conexiones Ethernet. Los cables de Ethernet y los dispositivos (como routers) están diseñados para soportar diferentes velocidades de comunicación, siendo el más bajo de 10 Mbps (10 Megabits por segundo) y la máxima 100 Gbps (100 gigabits por segundo). Las velocidades más comunes son de 100 Mbps y 1 Gbps.

**TCP/IP:** *Transmission Control Protocol/Internet Protocol (TCP/IP)* (o «*Protocolo de Control de Transmisión/Protocolo de Internet*» en español) es un nombre de adorno para una colección de protocolos (recuerda, protocolo = conjunto de reglas) que se utilizan para definir cómo la comunicación de la red debe ocurrir entre los hosts. Aunque no es la única colección de protocolos utilizados para definir la comunicación de la red, pero es la más utilizada. Por ejemplo, TCP/IP incluye la definición de cómo las direcciones IP y máscaras de red funcionan.

## 12.4 Las Direcciones IP

Como se mencionó anteriormente, los hosts «dirigen» paquetes de red usando la dirección IP de la máquina de destino. El paquete de red también incluye un «remitente», la dirección IP de la máquina origen.

De hecho, hay dos tipos de direcciones IP: IPv4 e IPv6. Para entender por qué hay dos tipos diferentes, es necesario entender un poco la breve historia del direccionamiento IP.

Durante muchos años, la tecnología de las direcciones IP utilizada por todas las computadoras era IPv4 (IP versión 4). En una dirección IPv4, un total de cuatro números de 8 bits (8 bits = números del 0 al 255) se utilizan para definir la dirección. Por ejemplo: 192 . 168 . 10 . 120. Ten en cuenta que esto se considera una dirección de 32 bits (4 x 8 bits = 32).

Cada host en Internet debe tener una dirección IP única. En un entorno IPv4, existe un límite técnico de unos 4.3 billones de direcciones IP. Sin embargo, muchas de estas direcciones IP no son realmente utilizables por varias razones. También, se han asignado direcciones IP a las organizaciones que no han hecho uso completo de todas las direcciones IP que tenían disponibles.

Mientras que parece que debe haber un montón de direcciones IP para ser utilizadas, varios factores (el creciente número de hosts en Internet, direcciones IP privadas reservadas, etc.) han causado un problema: El Internet comenzó a quedarse sin direcciones IP.



Esto, en parte, animó el desarrollo de IPv6. IPv6 fue «creada» oficialmente en 1998. En una red IPv6 las direcciones son mucho más grandes, direcciones de 128 bits que se ven así: 2001:0db8:85a3:0042:1000:8a2e:0370:7334. Esencialmente esto proporciona un grupo de direcciones mucho más grande, tan grande que quedarse sin direcciones en cualquier momento en un futuro cercano es muy poco probable.

Es importante tener en cuenta que la diferencia entre IPv4 e IPv6 no es sólo «más direcciones de IP». IPv6 tiene muchas otras características avanzadas que abordan algunas de las limitaciones de IPv4, como mayor velocidad, administración de paquetes más avanzada y transporte de datos más eficiente.

Teniendo en cuenta todas las ventajas, podrías pensar que ahora todos los hosts estarían usando IPv6. Este no es el caso en absoluto. La mayoría de los dispositivos conectados a la red en el mundo todavía utilizan IPv4 (algo así como 98-99% de todos los dispositivos). Así que, ¿por qué el mundo no ha adoptado la tecnología superior de IPv6?

Principalmente, hay dos razones:

- **La invención de NAT:** Inventado para superar la posibilidad de quedarse sin direcciones IP en un entorno IPv4, la *Network Address Translation (NAT)* (o «Traducción de Direcciones de Red» en español) utiliza una técnica para proporcionar más hosts de acceso a Internet. En resumen, un grupo de hosts se coloca en una red privada sin acceso directo a Internet; un router especial proporciona acceso a Internet y solamente este router necesita una dirección IP para comunicarse en Internet. En otras palabras, un grupo de hosts comparte una única dirección IP, lo que significa que muchas más computadoras se pueden conectar a Internet. Gracias a esta característica la necesidad de pasar a IPv6 es menos crítica que antes de la invención de NAT.
- **Cuestiones de Portabilidad:** La *Portabilidad* es el cambio de una tecnología a otra. IPv6 tiene muchas grandes novedades, pero todos los hosts tienen que ser capaces de utilizar estas características. Conseguir que todos en Internet (o incluso sólo algunos) hagan estos cambios supone un reto.

La mayoría de los expertos están de acuerdo en que IPv6 reemplazará IPv4, así que entender los fundamentos de ambos es importante para las personas que trabajan en la industria de TI.

## 12.5 Configurando los Dispositivos de Red

Cuando estás configurando los dispositivos de red, hay dos preguntas iniciales que debes considerar:

**¿Configuración por cable o inalámbrica?** La manera de configurar un dispositivo inalámbrico será ligeramente diferente a un dispositivo por cable debido a algunas de las características adicionales que se encuentran normalmente en los dispositivos inalámbricos (así como la seguridad).

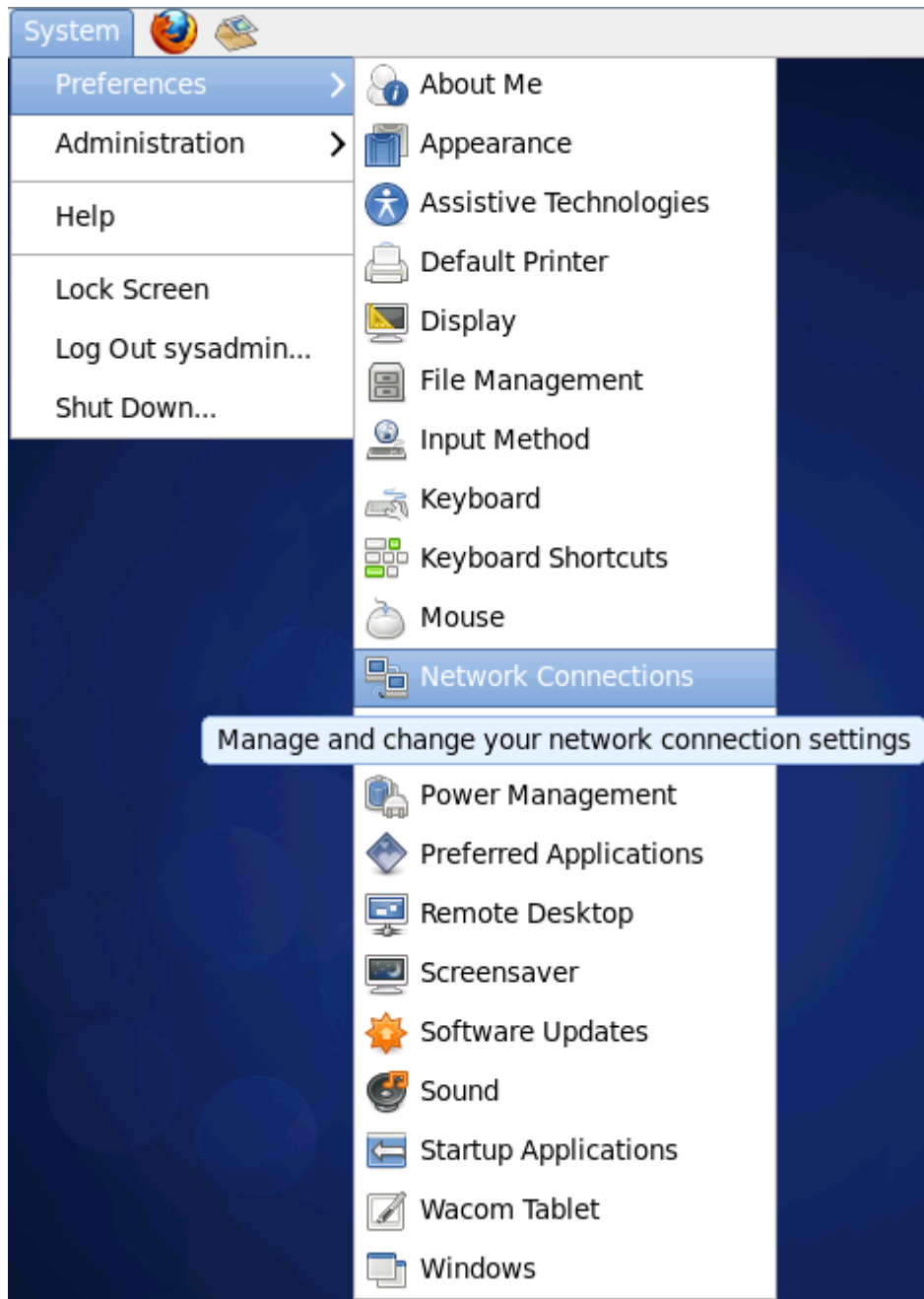
**¿DHCP o dirección estática?** Hay que recordar que un servidor DHCP proporciona información de la red, tal como tu dirección IP y la máscara de subred. Si tú no haces uso de un servidor DHCP, entonces tendrás que proporcionar manualmente esta información a tu host. Esto se llama utilizar una dirección IP estática.

En términos generales, una máquina de escritorio utilizará la red por cable, mientras que una computadora portátil utilizará una red inalámbrica. Normalmente una máquina por cable utiliza una dirección IP estática, pero éstas pueden asignarse también a menudo a través de un servidor DHCP. En casi todos los casos, las máquinas inalámbricas utilizan DHCP ya que son casi siempre móviles y conectadas a diferentes redes.

### 12.5.1 Configurar la Red usando una GUI

Si tienes acceso a un entorno GUI (interfaz gráfica de usuario), probablemente también tendrás acceso a una herramienta basada en GUI que te permitirá configurar tu red. Estas herramientas pueden variar de una distribución a otra. Los ejemplos siguientes se realizaron en una máquina CentOS.

Para iniciar la herramienta de configuración de red, haz clic en **System** (o «sistema» en español) en la barra de menú, luego **Preferences >** (o «preferencias» en español) y **Network Connections** (o «conexiones de red» en español):



La herramienta primero enlista todos los dispositivos de red actuales. En el ejemplo siguiente, hay sólo un dispositivo **Wired** (o «conectado por cable» en español):



El dispositivo de red es `eth0`. Los dispositivos de red se denominan `eth0`, `eth1`, etc. Para modificar este dispositivo de red, haz clic en el nombre del dispositivo y haz clic en el botón **Edit** (o «editar» en español):

**Editing System eth0**

Connection name: System eth0

☒ Connect automatically

☒ Available to all users

Wired 802.1x Security IPv4 Settings IPv6 Settings

Device MAC address: 00:50:56:90:18:18

Cloned MAC address:

MTU: automatic bytes

Cancel Apply...

El tema completo sobre todas las características de red está fuera del alcance de este curso. Esta sección se centrará en cambiar los componentes claves de la red.

Si haces clic en la pestaña **IPv4 Settings** (o «ajustes de IPv4» en español) vas a ver esto:

**Editing System eth0**

Connection name:

☒ Connect automatically

☒ Available to all users

Wired 802.1x Security **IPv4 Settings** IPv6 Settings

Method:

**Addresses**

| Address     | Netmask       | Gateway     |
|-------------|---------------|-------------|
| 192.168.1.2 | 255.255.255.0 | 192.168.1.1 |

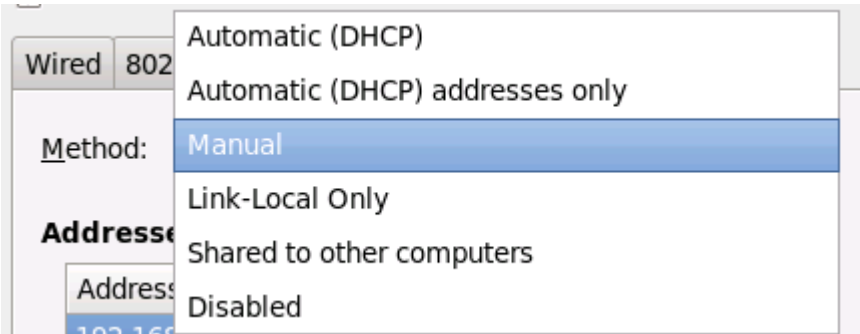
DNS servers:

Search domains:

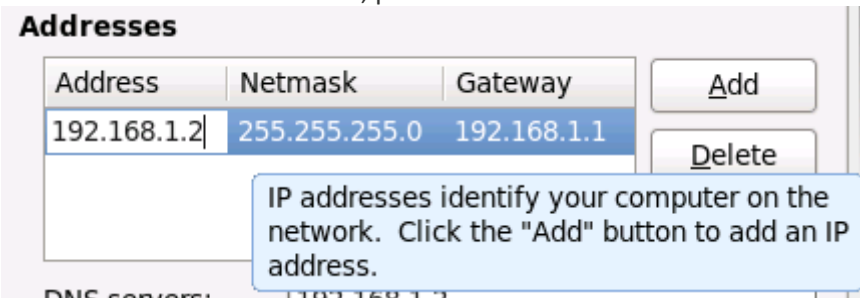
DHCP client ID:

☒ Require IPv4 addressing for this connection to complete

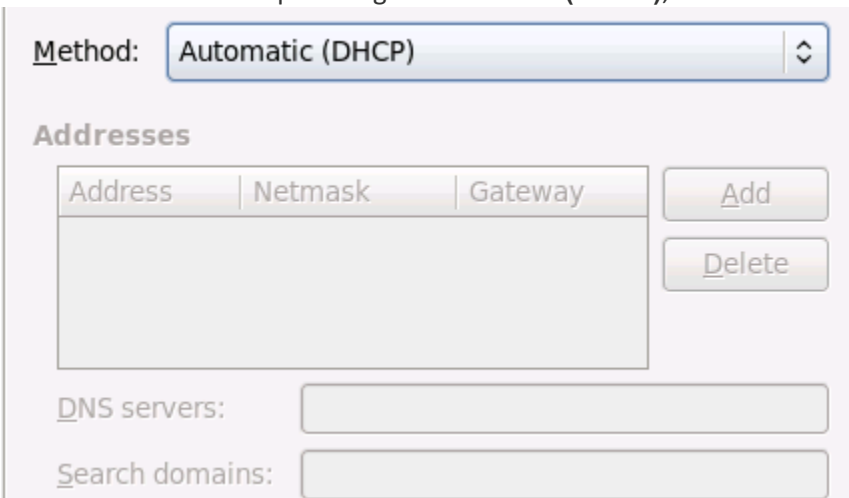
Recuerda que puedes asignar una dirección IP estática o utilizar un servidor DHCP (si está disponible). Este cambio se puede hacer haciendo clic en la lista desplegable situada junto a **Method** (o «método» en español):



Si seleccionas **Manual**, puedes cambiar la dirección actual haciendo clic en el área donde actualmente se especifica la dirección:

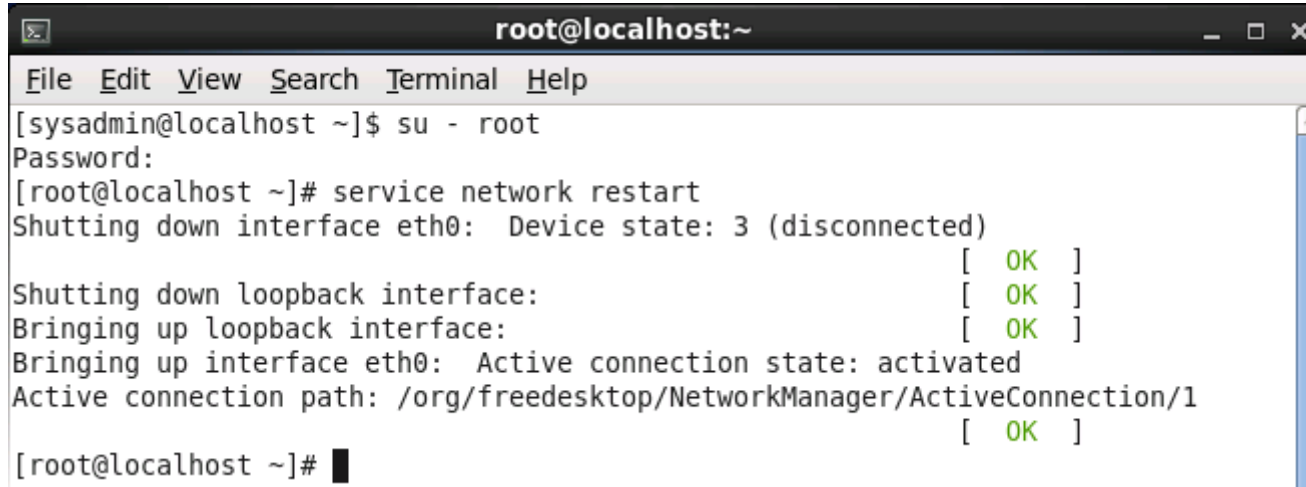


Ten en cuenta que si eliges **Automatic (DHCP)**, la ubicación de las **Direcciones** se verá en gris y no se podrá modificar:



**Importante:** Si cambias del automático (DHCP) a manual, todos los datos anteriores «desaparecen». Haciendo clic en el botón **Cancelar** y editando de nuevo el dispositivo `eth0`, volverán a aparecer los datos.

La mayoría de los cambios realizados con las herramientas basadas en GUI surten efecto inmediatamente después de que se guardan. Sin embargo, en algunos casos, tendrás que reiniciar el equipo o ejecutar un comando como administrador para que los cambios tomen efecto. A continuación se muestra el comando que debe ejecutarse en un sistema CentOS:



```

root@localhost:~
File Edit View Search Terminal Help
[sysadmin@localhost ~]$ su - root
Password:
[root@localhost ~]# service network restart
Shutting down interface eth0: Device state: 3 (disconnected)
[ OK ]
Shutting down loopback interface:
[ OK ]
Bringing up loopback interface:
[ OK ]
Bringing up interface eth0: Active connection state: activated
Active connection path: /org/freedesktop/NetworkManager/ActiveConnection/1
[ OK ]
[root@localhost ~]#

```

### 12.5.2 Configurar la Red usando el Archivo de Configuración

Habrán momentos cuando no haya herramienta gráfica disponible. En esos casos, es útil conocer los archivos de configuración que se utilizan para almacenar y modificar los datos de la red.

Estos archivos pueden variar según la distribución que estés utilizando. Los ejemplos siguientes se proporcionan para sistemas CENTOS.

#### 12.5.2.1 El Archivo Primario de Configuración de IPv4

El archivo primario de configuración para una interfaz de red IPv4 es el archivo `/etc/sysconfig/network-scripts/ifcfg-eth0`. El siguiente ejemplo muestra como se ve un archivo cuando se configura para una dirección IP estática:

```

root@localhost:~# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE="eth0"
BOOTPROTO=none
NM_CONTROLLED="yes"
ONBOOT=yes
TYPE="Ethernet"
UUID="98cf38bf-d91c-49b3-bb1b-f48ae7f2d3b5"
DEFROUTE=yes
IPV4_FAILURE_FATAL=yes
IPV6INOT=no
NAME="System eth0"

```

```

IPADDR=192.168.1.1
PREFIX=24
GATEWAY=192.168.1.1
DNS1=192.168.1.2
HWADDR=00:50:56:90:18:18
LAST_CONNECT=1376319928
root@localhost:~#

```

Si el dispositivo estuviera configurado para ser un cliente DHCP, entonces los valores `IPADDR`, `GATEWAY` y `DNS1` no se establecerían. Además, el valor `BOOTPROTO` se establecería a `dhcp`.

#### 12.5.2.2 El Archivo Primario de Configuración de IPv6

En un sistema CentOS, el archivo primario de configuración de IPv6 es el mismo archivo donde se almacena la configuración de IPv4: `/etc/sysconfig/network-scripts/ifcfg-eth0`. Si quieres que tu sistema tenga una dirección IPv6 estática, agrega lo siguiente al archivo de configuración:

```

IPV6INIT=yes
IPV6ADDR=<IPv6 IP Address>
IPV6_DEFAULTGW=<IPv6 IP Gateway Address>

```

Si quieres que tu sistema sea un cliente DHCP IPv6, agrega la siguiente configuración:

```
DHCPV6C=yes
```

También tienes que ajustar el archivo `/etc/sysconfig/network` de la siguiente manera:

```
NETWORKING_IPV6=yes
```

#### 12.5.2.3 Domain Name Service (DNS) (o «Servicio de Nombres de Dominio» en español)

Cuando a una computadora se le pide que acceda a una página web, como `www.example.com`, no necesariamente sabe qué dirección IP utilizar. Para que la computadora asocie una dirección IP con la solicitud de URL o nombre de host, la computadora depende del servicio DNS de otro equipo. A menudo, la dirección IP del servidor DNS se hace visible durante la solicitud de DHCP, mientras que una computadora recibe información importante para comunicar en la red.

La dirección del servidor DNS se almacena en el archivo `/etc/resolv.conf`. Un archivo `/etc/resolv.conf` típico se genera automáticamente y se ve así:

```

sysadmin@localhost:~$ cat /etc/resolv.conf
nameserver 127.0.0.1
sysadmin@localhost:~$

```



La configuración del servidor de nombres se establece a menudo en la dirección IP del servidor DNS. En el ejemplo siguiente se utiliza el comando `host` que vamos a ver más adelante en este capítulo. Ten en cuenta que el servidor de ejemplo se asocia con la dirección IP 192.168.1.2 por el servidor DNS:

```
sysadmin@localhost:~$ host example.com
example.com has address 192.168.1.2
sysadmin@localhost:~$
```

También es común tener varias opciones de servidor de nombres, si un servidor DNS no responde.

#### 12.5.2.4 Los Archivos Adicionales de Configuración de Red

La tabla siguiente describe los archivos de configuración de red adicionales que debes conocer. Aunque no figuran específicamente en los objetivos del examen, los objetivos incluyen el término general de **Configuración de Red**, por lo que estos archivos pueden aparecer en el examen:

| Comando                             | Explicación   |
|-------------------------------------|---|
| <code>/etc/hosts</code>             | Este archivo contiene una tabla de nombres de host para las direcciones IP. Puede utilizarse para complementar un servidor DNS.   |
| <code>/etc/sysconfig/network</code> | Este archivo tiene dos configuraciones. La configuración de <code>NETWORK</code> (o «red» en español) puede determinar si la red está activada ( <code>yes</code> ) o desactivada ( <code>no</code> ). La configuración de <code>HOSTNAME</code> (O «nombre de host» en español) define un nombre de host de la máquina local.  |
| <code>/etc/nsswitch.conf</code>     | Este archivo se puede utilizar para modificar dónde se producen las búsquedas de nombre de host. Por ejemplo, la configuración <code>hosts :</code><br><code>files dns</code> buscaría los nombres de host primero en el archivo <code>/etc/hosts</code> y después en el servidor DNS. Si cambias a <code>hosts: dns files</code> , la búsqueda se lleva a cabo primero en el servidor DNS. |

#### 12.5.2.5 Reiniciar la Red

Después de cambiar un archivo de configuración de red (por ejemplo, el archivo `/etc/sysconfig/network-scripts/ifcfg-eth0` o el archivo `/etc/resolv.conf`), necesitarás reiniciar la máquina o ejecutar un comando como administrador para que los cambios tomen efecto. A continuación se muestra el comando que tienes que ejecutar en un sistema CentOS:

```

root@localhost:~
File Edit View Search Terminal Help
[sysadmin@localhost ~]$ su - root
Password:
[root@localhost ~]# service network restart
Shutting down interface eth0: Device state: 3 (disconnected)
[ OK ]
Shutting down loopback interface:
[ OK ]
Bringing up loopback interface:
[ OK ]
Bringing up interface eth0: Active connection state: activated
Active connection path: /org/freedesktop/NetworkManager/ActiveConnection/1
[ OK ]
[root@localhost ~]#

```

## 12.6 Las Herramientas de Red

Hay varios comandos que puedes utilizar para ver la información de la red. Estas herramientas también pueden ser útiles cuando quieras solucionar problemas de la red.

### 12.6.1 El Comando `ifconfig`

El comando `ifconfig` significa «interface configuration» (o «configuración de la interfaz» en español) y se utiliza para mostrar la información de configuración de red. No todas las configuraciones de red se verán en este curso, pero es importante observar en la salida de abajo que la dirección IP del dispositivo de red principal (`eth0`) es `192.168.1.2` y que el dispositivo está activo (UP):

```

root@localhost:~# ifconfig
eth0      Link encap:Ethernet  HWaddr b6:84:ab:e9:8f:0a
          inet addr:192.168.1.2  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::b484:abff:fee9:8f0a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:95 errors:0 dropped:4 overruns:0 frame:0
          TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:25306 (25.3 KB)  TX bytes:690 (690.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0

```

```
TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:460 (460.0 B) TX bytes:460 (460.0 B)
root@localhost:~#
```

El dispositivo `lo` se conoce como el dispositivo de *loopback* (o «bucle invertido» en español). Es un dispositivo de red especial utilizado por el sistema cuando éste envía datos basados en red a sí mismo.

El comando `ifconfig` también puede ser utilizado para modificar temporalmente la configuración de red. Normalmente estos cambios deben ser permanentes, por lo que utilizar el comando `ifconfig` para hacer tales cambios es algo bastante raro.

El comando `ifconfig` se está convirtiendo obsoleto en algunas distribuciones de Linux (en desuso) y está siendo reemplazado por una forma del comando `ip`, específicamente `ip addr show`. Observa que también puedes encontrar la misma información destacada anteriormente utilizando este comando:

```
root@localhost:~# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
6476: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP qlen 1000
    link/ether b6:84:ab:e9:8f:0a brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.2/24 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::b484:abff:fee9:8f0a/64 scope link
        valid_lft forever preferred_lft forever
root@localhost:~#
```

### 12.6.2 El Comando `route`

Hay que recordar que un router (o puerta de enlace) es una máquina que permitirá que los hosts de una red se comuniquen con otra red. Para ver una tabla que describe donde se envían los paquetes de red utiliza el comando `route`:

```
root@localhost:~# route
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
192.168.1.0      *               255.255.255.0    U          0      0      0 eth0
```

```
default      192.168.1.1      0.0.0.0      UG      0      0      0 eth0
root@localhost:~#
```

El primer cuadro amarillo en el ejemplo anterior indica que cualquier paquete de red enviado a una máquina en la red 192.168.1 no se envía a una puerta de enlace (el \* indica «no hay puerta de enlace»). El segundo cuadro amarillo indica que todos los otros paquetes de red se envían al host con la dirección IP de 192.168.1.1 (el router).

Algunos usuarios prefieren visualizar esta información con sólo datos numéricos, usando la opción `-n` para el comando `route`. Por ejemplo, mira el siguiente ejemplo y enfócate en dónde la salida mostraba `default`:

```
root@localhost:~# route -n
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref      Use Iface
192.168.1.0      0.0.0.0          255.255.255.0    U        0      0          0 eth0
0.0.0.0          192.168.1.1      0.0.0.0          UG       0      0          0 eth0
root@localhost:~#
```

El 0.0.0.0 se refiere a «todas las otras máquinas», o lo mismo que «default».

El comando `route` se está volviendo obsoleto en algunas distribuciones de Linux (en desuso) y está siendo reemplazado por una forma del comando `ip`, específicamente `ip route show`. Observa que también puedes encontrar la misma información destacada anteriormente utilizando este comando:

```
root@localhost:~# ip route show
default via 192.168.1.254 dev eth0 proto static
192.168.1.0/24 dev eth0 proto kernel scope link src 192.168.1.2
root@localhost:~#
```

### 12.6.3 El Comando ping

El comando `ping` se puede utilizar para determinar si otra máquina es «accesible». Si el comando `ping` puede enviar un paquete de red a otra máquina y recibir una respuesta, entonces te deberías poder conectar a esa máquina.

De forma predeterminada, el comando `ping` continuará enviando paquetes una y otra vez. Para limitar cuántos pings se deben enviar, utiliza la opción `-c`.

Si el comando `ping` se realiza correctamente, verás una salida como la siguiente:

```
root@localhost:~# ping -c 4 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_req=1 ttl=64 time=0.051 ms
64 bytes from 192.168.1.2: icmp_req=2 ttl=64 time=0.064 ms
64 bytes from 192.168.1.2: icmp_req=3 ttl=64 time=0.050 ms
64 bytes from 192.168.1.2: icmp_req=4 ttl=64 time=0.043 ms
```

```

--- 192.168.1.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.043/0.052/0.064/0.007 ms
root@localhost:~#

```

Si el comando `ping` falla, recibirás un mensaje que dice `Destination Host Unreachable` (o «Host de destino inalcanzable» en español):

```

root@localhost:~# ping -c 4 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
From 192.168.1.2 icmp_seq=1 Destination Host Unreachable
From 192.168.1.2 icmp_seq=2 Destination Host Unreachable
From 192.168.1.2 icmp_seq=3 Destination Host Unreachable
From 192.168.1.2 icmp_seq=4 Destination Host Unreachable

--- 192.168.1.1 ping statistics ---
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 2999ms
pipe 4
root@localhost:~#

```

Es importante tener en cuenta que sólo porque el comando `ping` falle, no significa que el sistema remoto sea realmente inalcanzable. Algunos administradores configuran sus máquinas para no responder a las solicitudes de `ping`.

Esto suele pasar, porque un servidor puede ser atacado por algo que se llama *ataque por denegación de servicio*. En este tipo de ataque, un servidor es saturado con un número masivo de paquetes de red. Al ignorar las peticiones de `ping`, el servidor es menos vulnerable.

Como resultado, el comando `ping` puede ser útil para comprobar la disponibilidad de máquinas locales, pero no siempre para máquinas fuera de tu propia red.

#### 12.6.4 El Comando `netstat`

El comando `netstat` es una poderosa herramienta que proporciona una gran cantidad de información de la red. Puede utilizarse para mostrar información acerca de conexiones de red, así como para mostrar la tabla de enrutamiento similar al comando `route`.

Por ejemplo, puedes querer mostrar estadísticas acerca del tráfico de red. Esto puede lograrse mediante el uso de la opción `-i` del comando `netstat`:

```

root@localhost:~# netstat -i

Kernel Interface table

Iface    MTU Met    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0      1500 0         137     0       4 0        12     0     0  0 BMRU
lo        65536 0          18     0       0 0        18     0     0  0 LRU

```

```
root@localhost:~#
```

Las estadísticas más importantes de la salida anterior son TX-OK y TX-ERR. Un alto porcentaje de TX-ERR puede indicar un problema en la red, tal como mucho tráfico de red.

Si quieres utilizar el comando `netstat` para mostrar la información de enrutamiento, utiliza la opción `-r`:

```
root@localhost:~# netstat -r
```

Kernel IP routing table

| Destination | Gateway     | Genmask       | Flags | MSS Window | irtt | Iface |
|-------------|-------------|---------------|-------|------------|------|-------|
| 192.168.1.0 | *           | 255.255.255.0 | U     | 0 0        | 0    | eth0  |
| default     | 192.168.1.1 | 0.0.0.0       | UG    | 0 0        | 0    | eth0  |

```
root@localhost:~#
```

El comando `netstat` se utiliza comúnmente para mostrar *puertos* abiertos. Un puerto es un número único que está asociado con un servicio proporcionado por un host. Si el puerto está abierto, el servicio está disponible para otros hosts.

Por ejemplo, puedes iniciar sesión en un host desde otro host utilizando un servicio llamado *SSH*. El servicio SSH tiene asignado el puerto #22. Si el puerto #22 está abierto, el servicio está disponible para otros hosts.

Es importante tener en cuenta que el host mismo también debe tener los servicios en ejecución; esto significa que debe iniciarse el programa que permite a los usuarios remotos conectarse (que por lo general está iniciado en la mayoría de las distribuciones de Linux).

Para ver una lista de todos los puertos actualmente abiertos, puedes utilizar el siguiente comando:

```
root@localhost:~# netstat -tln
```

Active Internet connections (only servers)

| Proto | Recv-Q | Send-Q | Local Address  | Foreign Address | State  |
|-------|--------|--------|----------------|-----------------|--------|
| tcp   | 0      | 0      | 192.168.1.2:53 | 0.0.0.0:*       | LISTEN |
| tcp   | 0      | 0      | 127.0.0.1:53   | 0.0.0.0:*       | LISTEN |
| tcp   | 0      | 0      | 0.0.0.0:22     | 0.0.0.0:*       | LISTEN |
| tcp   | 0      | 0      | 127.0.0.1:953  | 0.0.0.0:*       | LISTEN |
| tcp6  | 0      | 0      | :::53          | :::*            | LISTEN |
| tcp6  | 0      | 0      | :::22          | :::*            | LISTEN |
| tcp6  | 0      | 0      | :::1:953       | :::*            | LISTEN |

```
root@localhost:~#
```

Como se puede ver en la salida anterior, el puerto #22 está "escuchando (LISTEN)", lo que significa que está abierto.

En el ejemplo anterior, la `-t` se refiere a TCP (recuerda que este protocolo lo vimos de anteriormente en este capítulo), `-l` significa «listening» (o «escuchando» en español) (cuáles de los puertos están escuchando) y `-n` significa «mostrar números, no nombres».

A veces, mostrar los nombres puede ser más útil. Sólo elimina la opción `-n`:

```

root@localhost:~# netstat -tl

Active Internet connections (only servers)

Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 cserver.example.:domain *:*                     LISTEN
tcp        0      0 localhost:domain        *:*                     LISTEN
tcp        0      0 *:ssh                   *:*                     LISTEN
tcp        0      0 localhost:953           *:*                     LISTEN
tcp6       0      0 [::]:domain             [::]:*                  LISTEN
tcp6       0      0 [::]:ssh                 [::]:*                  LISTEN
tcp6       0      0 localhost:953           [::]:*                  LISTEN
root@localhost:~#

```

En algunas distribuciones se puede ver el siguiente mensaje en la página man del comando `netstat`:

#### NOTE

This program is obsolete. Replacement for `netstat` is `ss`. Replacement for `netstat -r` is `ip route`. Replacement for `netstat -i` is `ip -s link`. Replacement for `netstat -g` is `ip maddr`.

Aunque el comando `netstat` no se sigue desarrollando, sigue siendo una excelente herramienta para visualizar información de la red. El objetivo es eventualmente reemplazar el comando `netstat` por comandos como `ss` e `ip`. Sin embargo, es importante tener en cuenta que esto puede tomar algún tiempo.

El comando `netstat` viene en este curso ya que está disponible en todas las distribuciones de Linux, todavía es ampliamente utilizado y es un objetivo del examen de Linux Essentials (los comandos `ss` e `ip` no lo son).

#### 12.6.5 El Comando `dig`

Puede haber ocasiones cuando necesites probar la funcionalidad del servidor DNS que tu host está utilizando. Una forma de hacerlo es utilizar el comando `dig`. Este comando realizará consultas en el servidor DNS para determinar si la información necesaria está disponible en el servidor.

En el ejemplo siguiente, se utiliza el comando `dig` para determinar la dirección IP del host `example.com`:

```

root@localhost:~# dig example.com

; <<>> DiG 9.8.1-P1 <<>> example.com

;; global options: +cmd

;; Got answer:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45155

;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:

```

```
;example.com.                IN      A

;; ANSWER SECTION:
example.com.      86400    IN      A      192.168.1.2
;; AUTHORITY SECTION:
example.com.      86400    IN      NS      example.com.

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Tue Dec  8 17:54:41 2015
;; MSG SIZE rcvd: 59
root@localhost:~#
```

Observa que la respuesta incluye la dirección IP de 192.168.1.2, lo que significa que el servidor DNS tiene la dirección IP asociada a la información de traducción del nombre de host en su base de datos.

Si el servidor DNS no tiene la información solicitada, está configurado para mandar solicitud a otros servidores DNS. Si ninguno de ellos tiene la información solicitada, recibirás un mensaje de error:

```
root@localhost:~# dig sample.com

; <<>> DiG 9.8.1-P1 <<>> sample.com
;; global options: +cmd
;; connection timed out; no servers could be reached
root@localhost:~#
```

### 12.6.6 El Comando host

En su forma más simple, el comando `host` trabaja con DNS para asociar un nombre de host a una dirección IP. Tal como en el ejemplo anterior, ejemplo.com se asocia a la dirección IP 192.168.1.2:

```
root@localhost:~# host example.com
example.com has address 192.168.1.2
root@localhost:~#
```

El comando `host` se puede utilizar también en sentido inverso si se conoce una dirección IP, pero no el nombre del dominio.

```
root@localhost:~# host 192.168.1.2
2.1.168.192.in-addr.arpa domain name pointer example.com.
2.1.168.192.in-addr.arpa domain name pointer cserver.example.com.
```



```
root@localhost:~#
```

Existen otras opciones para consultar los diferentes aspectos de un DNS, así como la **CNAME** (nombre canónico -alias):

```
root@localhost:~# host -t CNAME example.com
```

```
example.com has no CNAME record
```

```
root@localhost:~#
```

Puesto que muchos servidores DNS guardan una copia de example.com, los registros **SOA** (Start of Authority) indican el servidor principal para el dominio:

```
root@localhost:~# host -t SOA example.com
```

```
example.com has SOA record example.com. cserver.example.com. 2 604800 86400 2419200 604800
```

```
root@localhost:~#
```

Puedes encontrar una lista completa de información sobre DNS en relación con el `example.com` usando la opción **-a** («all» o «todo» en español):

```
root@localhost:~# host -a example.com
```

```
Trying "example.com"
```

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 3549
```

```
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1
```

```
;; QUESTION SECTION:
```

```
;example.com.                IN      ANY
```

```
;; ANSWER SECTION:
```

```
example.com.                86400   IN      SOA    example.com. cserver.example.com. 2 604800 86400 2419200 604800
```

```
example.com.                86400   IN      NS     example.com.
```

```
example.com.                86400   IN      A      192.168.1.2
```

```
;; ADDITIONAL SECTION:
```

```
example.com.                86400   IN      A      192.168.1.2
```

```
Received 119 bytes from 127.0.0.1#53 in 0 ms
```

```
root@localhost:~#
```

### 12.6.7 El Comando ssh

El comando `ssh` te permitirá conectarte a otra máquina a través de la red, iniciar sesión y luego realizar tareas en el equipo remoto.

Si utilizas el comando `ssh` y sólo proporcionas el nombre de la máquina o la dirección IP para iniciar la sesión, el comando asumirá que quieres iniciar la sesión con el mismo nombre con el que actualmente estás registrado. Si quieres utilizar un nombre de usuario distinto, utiliza la sintaxis:

```
username@hostname (o «NombreDeUsuario@NombreDelHostlocal» en español)
```

```
root@localhost:~# ssh bob@test
```

```
The authenticity of host 'test (127.0.0.1)' can't be established.
```

```
RSA key fingerprint is c2:0d:ff:27:4c:f8:69:a9:c6:3e:13:da:2f:47:e4:c9.
```

```
Are you sure you want to continue connection (yes/no)? yes
```

```
Warning: Permanently added 'test' (RSA) to the list of known hosts.
```

```
bob@test's password:
```

```
bob@test:~$
```

```
Fri Oct 4 16:14:43 CDT 2013
```

```
bob@test:~$
```

#### 12.6.7.1 Algoritmo RSA de Clave Pública

El primer prompt te pide que verifiques la identidad de la máquina en la que inicias sesión. En la mayoría de los casos vas a responder `yes` (o «sí» en español).

Aunque puedas validar con administrador de la máquina remota para asegurarte de que la clave RSA es correcta, este no es realmente el propósito de esta consulta.

Realmente está diseñado para futuros inicios de sesión.

Después de que respondas `yes`, la clave RSA de la máquina remota se almacena en tu sistema local. Cuando intentes hacer un `ssh` a esta misma máquina en el futuro, la clave RSA proporcionada por el equipo remoto se compara con la copia almacenada en el equipo local. Si coinciden, entonces aparece el prompt del nombre de usuario. Si no coinciden, verás un error similar al siguiente:

```
sysadmin@localhost:~$ ssh bob@test
```

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

```
@ WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! @
```

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

```
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
```

```
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
```

```
It is also possible that the RSA host key has just been changed.
```

```
The fingerprint for the RSA key sent by the remote host is
```

```
c2:0d:ff:27:4c:f8:69:a9:c6:3e:13:da:2f:47:e4:c9.
Please contact your system administrator.
Add correct host key in /home/sysadmin/.ssh/known_hosts to get rid of this message.
Offending key in /home/sysadmin/.ssh/known_hosts:1
RSA host key for test has changed and you have requested strict checking.
Host key verification failed.
sysadmin@localhost:~$
```

Este error puede indicar que un host no autorizado ha reemplazado al host correcto. Consulta con el administrador del sistema remoto. Si el sistema fuese recientemente reinstalado, tendría una nueva clave RSA, lo podría estar causando este error.

En caso de que este mensaje de error es debido a que una máquina remota fue reinstalada, puedes eliminar el archivo `~/.ssh/known_hosts` de tu sistema local (o solo quitar la entrada para esa máquina en específico) e intentar a conectarte de nuevo:

```
sysadmin@localhost:~$ cat ~/.ssh/known_hosts
test ssh-rsa AAAAB3NzaC1yc2EAAAAMiWAAQEAklOUUpkDhRfHY17SbrmTIp/RZ0V4DTxgq9wzd+ohy006SWDSGPA+nafz1HDPow7vdI4mZ5ew18KL4JW9jbhUfrviQ
zM7xlELEVf4h9lFX5QVkbPppSrg0cda3Pbv7kOdJ/MTyBlWXFCRH+Cv3FXRitBqxiX1nKhXpHAZsMciLq8V6RjsNAQwdsdMFvSlVK/7BA
t5FaiKoAfnCM1Q8x3+2V0Ww71/eIFmb1zuUF1jHYTprX88XypNDvjYNby6vw/Pb0rwprz/Tn
mZAW3UX+PnTPI89ZPmNBLuxyrD2cE86Z/il8b+gw3r3+1nJotmIkjn2sold01QraTlMqVSsbx
NrRFi9wrf+ghw==
sysadmin@localhost:~$ rm ~/.ssh/known_hosts
sysadmin@localhost:~$ ssh bob@test
The authenticity of host 'test (127.0.0.1)' can't be established.
RSA key fingerprint is c2:0d:ff:27:4c:f8:69:a9:c6:3e:13:da:2f:47:e4:c9.
Are you sure you want to continue connection (yes/no)? yes
Warning: Permanently added 'test' (RSA) to the list of known hosts.
bob@test's password:
Last login: Fri Oct 4 16:14:39 CDT 2013 from localhost
bob@test:~$
```

#### 12.6.7.2 Regresar a la Máquina Local

Para volver a la máquina local utiliza el comando de `exit` (o «salir» en español):

```
bob@test:~$ exit
```

```
logout
Connection to test closed.
sysadmin@localhost:~#
```

**Advertencia:** ¡Ten cuidado al utilizar el comando `exit` muchas veces, ya que se cerrará la ventana de la terminal en la que estás trabajando!

## NDG Linux Essentials: Capítulo 13: Seguridad y sistema del usuario.

### 13.1 Introducción

Las cuentas de usuario están diseñadas para proporcionar seguridad en un sistema operativo Linux. Cada persona en el sistema debe iniciar la sesión utilizando una cuenta de usuario y la cuenta de usuario permite a la persona ya sea acceder a un directorio de archivos específico o niega dicho acceso. Esto se logra mediante los permisos de archivo, un tema del que hablaremos en un capítulo posterior.

Las cuentas de usuario también pertenecen a los grupos que también pueden utilizarse para proporcionar acceso a los archivos o directorios. Cada usuario pertenece al menos a un grupo (a menudo muchos) para permitir más fácilmente que los usuarios compartan los datos almacenados en los archivos con otros usuarios.

Los datos de la cuenta de usuario y de grupo se almacenan en archivos de base de datos. Conocer el contenido de estos archivos es importante, ya que te permitirá entender mejor cuáles de los usuarios tienen acceso a los archivos y directorios en el sistema. Estos archivos de base de datos también contienen información de seguridad vital que puede afectar la capacidad de un usuario a acceder al sistema (login).

Hay varios comandos que te proporcionarán la capacidad de ver información de la cuenta de grupo y usuario, así como te permiten cambiar de una cuenta de usuario a otra (siempre que tengas la autorización para ello). Estos comandos son valiosos para revisar el uso del sistema, solucionar los problemas de sistema y controlar el acceso no autorizado al sistema.

**“No tengo ninguna experiencia de trabajo en Linux, entonces ¿Cómo obtengo un trabajo?** Constatando tus habilidades obteniendo un certificado reconocido por la industria que demuestra a los empleadores que tienes las habilidades de hacer el trabajo.

Puede ser un gran forma de entrar a la primera compañía para que puedas ganar valor con la experiencia en el trabajo.

### 13.2 Las Cuentas de Usuario

Hay varios archivos de texto en el directorio `/etc` que contienen los datos de la cuenta de los usuarios y grupos definidos en el sistema. Por ejemplo, si quisieras ver si está definida una cuenta de usuario específica en el sistema, el lugar para comprobar es el archivo `/etc/passwd`.

El archivo `/etc/passwd` define parte de la información de la cuenta para las cuentas de usuario. Curiosamente, las contraseñas para las cuentas no se almacenan en el archivo `/etc/passwd`, tal como lo indica el nombre del archivo, sino más bien el archivo `/etc/shadow`.

#### 13.2.1 El Archivo `/etc/passwd`

Cada línea del archivo `/etc/passwd` se refiere a una cuenta de usuario. El siguiente gráfico muestra las diez primeras líneas de un archivo `/etc/passwd` típico:

```
sysadmin@localhost:~$ head /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
```

```
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
sysadmin@localhost:~$
```

Cada línea está dividida en campos por dos puntos. Los campos de izquierda a derecha son los siguientes:

```
name:password placeholder:user id:primary group id:comment:home directory:shell
```

La siguiente tabla describe cada uno de estos campos en detalle, usando la primera línea de la salida en el ejemplo anterior (root:x:0:0:root:/root:/bin/bash):

| Campo | Ejemplo | Descripción   |
|-------|---------|---|
| name  | root    | <p>Es el nombre de la cuenta. Este nombre lo utiliza una persona cuando inicia sesión en el sistema y cuando la propiedad del archivo viene proporcionada por el comando <code>ls -l</code>. Por lo general, el sistema utiliza el ID de usuario (véase abajo) internamente y se proporciona el nombre de cuenta para que a los usuarios regulares se les haga más fácil referirse a la cuenta.</p> <p>Normalmente, la cuenta <code>root</code> es una cuenta administrativa especial. Sin embargo, es importante tener en cuenta que no todos los sistemas tienen una cuenta <code>root</code>, y en realidad, el ID de usuario <code>0</code> (cero) proporciona los privilegios administrativos en el sistema.</p> |

| Campo                   | Ejemplo | Descripción   |
|-------------------------|---------|---|
| password<br>placeholder | x       | <p>Antes, la contraseña se guardaba en esta ubicación, ahora se almacena en el archivo <code>/etc/shadow</code>. La <code>x</code> en el campo del marcador de posición de la contraseña indica al sistema que la contraseña no se almacena aquí, sino más bien en el archivo <code>/etc/shadow</code>.</p>   |
| user id                 | 0       | <p>Cada cuenta tiene asignado un Id. de usuario (UID «User ID» en inglés). El UID es lo que realmente define la cuenta, ya que el nombre de usuario normalmente no es utilizado directamente por el sistema. Por ejemplo, los archivos son propiedad de los UID, no de los nombres de usuario.</p> <p>Algunos UID son especiales. Por ejemplo, el UID 0 le da a la cuenta de usuario privilegios administrativos.</p> <p>Los UID por debajo de 500 (en algunas distribuciones de Linux 1.000) están reservados para las cuentas del sistema. Las cuentas del sistema se tratarán con más detalle más adelante en este capítulo.</p> |
| primary<br>group id     | 0       | <p>Cada archivo y directorio es propiedad de una cuenta de usuario. Normalmente la persona que crea la cuenta posee el archivo. Además, cada archivo es propiedad de un grupo, normalmente del <i>grupo primario</i> del usuario.</p> <p>A los grupos se les asignan identificadores numéricos al igual que a los usuarios.</p> <p>Cuando un usuario crea un archivo, el archivo es propiedad del UID del usuario y también de un id de</p>   |

| Campo             | Ejemplo | Descripción   |
|-------------------|---------|---|
|                   |         | <p>grupo (GID «Group ID» en inglés), el GID primario del usuario. Este campo define que GID es el GID primario del usuario.</p> <p>Además, al proporcionar la propiedad del grupo por defecto en un archivo, este campo también indica que el usuario es un miembro del grupo, lo que significa que el usuario tendrá permisos especiales en cualquier archivo que pertenece a este grupo. Los permisos se cubrirán en detalle en un capítulo posterior.</p>  |
| comment           | root    | <p>Este campo puede contener cualquier información sobre el usuario, incluyendo su nombre real (completo) y otra información útil.</p> <p>Este campo también se llama el campo GECOS (General Electric Comprehensive Operating System). El GECOS es un formato predefinido usado raramente para este campo que define una lista de elementos separada por comas, incluyendo el nombre completo del usuario, ubicación de la oficina, número de teléfono e información adicional.</p> <p>El administrador puede modificar la información GECOS con el comando <code>chfn</code> y los usuarios pueden ver esta información con el comando <code>finger</code>.</p> |
| home<br>directory | /root   | <p>Este campo define la ubicación del directorio home del usuario. Para los usuarios regulares, esto sería normalmente <code>/home/username</code> donde <code>username</code> se reemplaza con el nombre de usuario del usuario. Por ejemplo, un nombre de usuario <code>bob</code> tendría un directorio home <code>/home/bob</code>.</p>   |

| Campo              | Ejemplo                | Descripción  |
|--------------------|------------------------|--|
|                    |                        | <p>El usuario <code>root</code> tiene normalmente una ubicación diferente para el directorio home: <code>/root</code>.</p> <p>Las cuentas del sistema raramente tienen directorios, ya que normalmente no se utilizan para crear o guardar los archivos.</p>   |
| <code>shell</code> | <code>/bin/bash</code> | <p>Aquí está ubicado el shell del inicio de la sesión del usuario. De forma predeterminada, el usuario se "ubica en" este shell siempre que el usuario inicia la sesión en un entorno de línea de comandos o abre una ventana de terminal. El usuario luego puede pasar a un shell diferente escribiendo el nombre del shell, por ejemplo: <code>/bin/tcsh</code>.</p> <p>El shell <code>bash (/bin/bash)</code> es el más común para los usuarios de Linux.</p> |

Ten en cuenta que mientras que este capítulo describe el contenido de los archivos de grupo y usuario, el siguiente capítulo describirá los comandos y herramientas que se utilizan para modificar la información de la cuenta del usuario y de grupos.

### 13.2.2 El Archivo `/etc/shadow`

Como se mencionó anteriormente, el archivo `/etc/shadow` contiene información de la cuenta relacionada con la contraseña del usuario. Un archivo `/etc/shadow` típico sería como el siguiente gráfico:

```
root@localhost:~# head /etc/shadow
root:$6$4Yga95H9$8HbxqsMEIBTZ0YomlMffYCV9VE1SQ4T2H3SHXw41M02SQtfAdDVE9mqGp2hr20q
.ZuncJpLyWkYwQdKlSJyS8.:16464:0:99999:7:::
daemon*:16463:0:99999:7:::
bin*:16463:0:99999:7:::
sys*:16463:0:99999:7:::
sync*:16463:0:99999:7:::
games*:16463:0:99999:7:::
man*:16463:0:99999:7:::
```



```
lp:*:16463:0:99999:7:::
mail:*:16463:0:99999:7:::
news:*:16463:0:99999:7:::
root@localhost:~#
```

Los campos del archivo `/etc/shadow` son:

```
name:password:last change:min:max:warn:inactive:expire:reserved
```

La siguiente tabla describe los campos del archivo `/etc/shadow` con más detalle, utilizando la siguiente cuenta la cuál describe a una cuenta de usuario típica:

```
sysadmin:$6$1s6WJ9O/fNmEzrIi$ko9NKRBJLJjTlZD.L1Dc2xwcuUYaYwCTS.gt4elijSQW8ZDp6GLYAx.TRNNpUdAgUXUrzDuAPsYs5YHZNAorI1:15020:5:30:7:60:15050:
```

| Campo    | Ejemplo       | Descripción  |
|----------|---------------|--|
| name     | sysadmin      | Este es el nombre de la cuenta, que coincide con el nombre de cuenta en el archivo <code>/etc/passwd</code> .  |
| password | \$6\$.....rI1 | <p>El campo <code>passwd</code> contiene la contraseña cifrada de la cuenta. Esta cadena muy larga (que está truncada en el ejemplo a la izquierda de esta celda) es un cifrado unidireccional, lo que significa que no puede "revertirse" para determinar la contraseña original.</p> <p>Mientras que los usuarios habituales tienen contraseñas encriptadas en este campo, las cuentas del sistema tendrán un carácter de <code>*</code> en este campo. Verás más detalles sobre las cuentas de sistema más adelante en este capítulo.</p> |

| Campo          | Ejemplo | Descripción   |
|----------------|---------|---|
| last<br>change | 15020   | <p>Este campo contiene un número que representa la última vez que la contraseña fue cambiada. El número 15020 es el número de días desde el 01 de enero de 1970 (llamada la Época) y el último día en el que la contraseña de la cuenta fue cambiada.</p> <p>Este valor se genera automáticamente cuando se modifica la contraseña del usuario. Este valor es importante ya que se utiliza por característica de <i>envejecimiento de la contraseña</i> proporcionado por el resto de los campos de este archivo.</p> |
| min            | 5       | <p>Este es uno de los campos del <i>envejecimiento de la contraseña</i>; un valor distinto de cero en este campo indica que después de que un usuario cambiara su contraseña, la contraseña no se puede cambiar otra vez por un número específico de días (5 en este ejemplo). Este campo es importante cuando se utiliza el campo <code>max</code> (véase abajo).</p> <p>Un valor de cero en este campo significa que el usuario siempre puede cambiar su contraseña.</p>  |
| max            | 5       | <p>Este campo se utiliza para obligar a los usuarios a cambiar regularmente sus contraseñas. Un valor de 30 en este campo significa que el usuario debe cambiar su contraseña al menos cada 30</p>  |

| Campo | Ejemplo | Descripción  |
|-------|---------|--|
|       |         | <p>días para evitar que su cuenta quede «bloqueada»</p> <p>Ten en cuenta que si el campo <code>min</code> se establece en 0, el usuario puede restablecer inmediatamente su contraseña al valor original, anulando el propósito de obligar al usuario a cambiar su contraseña cada 30 días. Así que, si se establece el campo <code>max</code>, normalmente se establece también el campo <code>min</code>.</p> <p>Por ejemplo, un <code>min:max</code> de 5:60 se refiere a que el usuario debe cambiar su contraseña cada 60 días y, después de cambiarla, el usuario debe esperar 5 días antes de que pueda cambiar su contraseña otra vez.</p> <p>Si el campo <code>max</code> se establece a 99999, el valor máximo posible, entonces el usuario esencialmente nunca debe cambiar su contraseña (porque 99999 días son aproximadamente 274 años).</p> |
| warn  | 7       | <p>Si se establece el campo <code>max</code>, el campo <code>warn</code> indica que al usuario se le «advertirá» cuando se acerca el plazo máximo <code>max</code>. Por ejemplo, si se establece <code>warn</code> a 7, entonces en cualquier momento durante los 7 días anteriores al plazo <code>max</code> se le advertirá al usuario que cambie su contraseña durante el proceso del inicio de sesión.</p>   |

| Campo                 | Ejemplo | Descripción  |
|-----------------------|---------|--|
|                       |         | <p>El usuario sólo se advierte en el inicio de sesión, por lo que algunos administradores prefieren establecer el campo de advertencia a un valor alto para proporcionar una mayor probabilidad de tener una advertencia emitida.</p> <p>Si el plazo máximo <code>max</code> se establece al valor de <code>99999</code>, entonces el campo <code>warn</code> es esencialmente inútil</p>  |
| <code>inactive</code> | 60      | <p>Si el usuario hace caso omiso a las advertencias y se excede el plazo <code>max</code> para la contraseña, se bloqueará su cuenta. En tal caso, el campo <code>inactive</code> proporciona al usuario un período de «gracia» en el que puede cambiar su contraseña, pero sólo durante el proceso del inicio de sesión.</p> <p>Si el campo <code>inactive</code> viene establecido a un valor de 60, el usuario tiene 60 días de gracia para cambiar la contraseña a una nueva. Si no lo hace, entonces necesitará que el administrador restablezca la contraseña para el usuario.</p> |
| <code>expire</code>   | 15050   | <p>Este campo representa el número de días a partir del 01 de enero del 1970 y el día en el que la cuenta «vencerá». Una cuenta vencida se bloqueará, no se borra, lo que significa que el administrador puede restablecer la contraseña para desbloquear la cuenta.</p>   |

| Campo | Ejemplo  | Descripción  |
|-------|----------|--|
|       |          | Las cuentas con fechas de vencimiento normalmente se ofrecen a los empleados temporales o contratistas. La cuenta caducará automáticamente después del último día laboral del usuario. |
|       |          | Cuando un administrador establece este campo, se utiliza una herramienta para convertir una fecha real en una fecha de «Época». En Internet hay varios convertidores disponibles.      |
|       | reserved | Actualmente este campo no se utiliza y está reservado para su uso futuro.  |

Los usuarios regulares no pueden ver el contenido del archivo `/etc/shadow` por razones de seguridad. Para ver el contenido de este archivo, debes iniciar la sesión como administrador (la cuenta `root`).

### 13.2.3 Visualizar la Información de la Cuenta

Una buena manera de visualizar la información de la cuenta del archivo `/etc/passwd` es usar el comando `grep` para dar salida solamente a la línea que contiene la cuenta que te interesa ver. Por ejemplo, para ver la información de la cuenta del nombre de usuario llamado `sysadmin`, utiliza el comando `grep sysadmin /etc/passwd`:

```
sysadmin@localhost:~$ grep sysadmin /etc/passwd
sysadmin:x:1001:1001:System Administrator,,,:/home/sysadmin:/bin/bash
sysadmin@localhost:~$
```

Otra técnica para recuperar la información del usuario que se encuentra normalmente en los archivos `/etc/passwd` y `/etc/shadow` es utilizar el comando `getent`. Una ventaja de usar el comando `getent` es que puede recuperar la información de la cuenta definida localmente (`/etc/passwd` y `/etc/shadow`) o en un servidor de directorio en la red.

La sintaxis general de un comando `getent` es: `getent database record` «o `getent registro de base de datos`». Por ejemplo, el comando `getent passwd sysadmin` recuperaría la información `passwd` de la cuenta para el usuario `sysadmin`:

```
sysadmin@localhost:~$ getent passwd sysadmin
sysadmin:x:1001:1001:System Administrator,,,:/home/sysadmin:/bin/bash
```

```
sysadmin@localhost:~$
```

### 13.2.4 Visualización de la Información del Inicio de sesión

Si inicias sesión con diferentes cuentas, puede ser confuso saber cómo quién estás conectado. Para verificar tu identidad (ver qué cuenta estás usando actualmente) puedes ejecutar el comando `id`.

El comando `id` informará la identidad actual, tanto por nombre del usuario como por el identificador de usuario. Además de proporcionar la información de la cuenta de usuario, también se muestra la pertenencia a un grupo. Sin argumentos, el comando `id` mostrará tu identidad. Cuando un nombre de usuario viene como argumento, tal como `id root`, el comando muestra otra información de cuenta:

```
sysadmin@localhost:~$ id
uid=1001(sysadmin) gid=1001(sysadmin) groups=1001(sysadmin),4(adm),27(sudo) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023

sysadmin@localhost:~$
sysadmin@localhost:~$ id root
uid=0(root) gid=0(root) groups=0(root)

sysadmin@localhost:~$
```

La salida incluye un contexto Linux mejorado de seguridad (SELinux), la parte `context=` de la salida, un tema que está fuera del alcance de este curso.

### 13.2.5 Las Cuentas de Sistema

Los usuarios iniciarán sesión en el sistema utilizando cuentas de usuario regulares. Normalmente, estas cuentas tienen valores UID superiores a 500 (en algunos sistemas 1000).

El usuario `root` tiene acceso especial al sistema. Como se mencionó anteriormente, este acceso especial en realidad se proporciona a la cuenta con el UID de 0.

Hay cuentas adicionales que no están diseñadas para que los usuarios inicien sesión. Estas cuentas, normalmente de UID 1 al UID 499 se llaman cuentas del sistema y están diseñadas para proporcionar cuentas para los servicios que se ejecutan en el sistema.

Las cuentas del sistema tienen algunos campos en los archivos `/etc/passwd` y `/etc/shadow` que son diferentes a otras cuentas. Por ejemplo, en el archivo `/etc/passwd`, las cuentas de sistema tendrán un programa de no iniciar sesión en el campo del «login shell»:

```
bin:x:1:1:bin:/bin:/sbin:/sbin/nologin
```

En el archivo `/etc/shadow`, las cuentas del sistema tendrán típicamente el carácter `*` en lugar del campo de contraseña:

```
bin:!:15513:0:99999:7:::
```

Hay algunas cosas importantes que debes recordar sobre las cuentas de sistema:

- La mayoría son necesarias para que el sistema funcione correctamente.
- No debes eliminar una cuenta del sistema a menos que estés absolutamente seguro de que eliminando la cuenta no causará problemas.

- Según vayas teniendo más experiencia, debes aprender lo que hace cada cuenta del sistema. Los administradores del sistema tienen la tarea de garantizar la seguridad en el sistema esto incluye correctamente asegurar las cuentas del sistema.

13.3 Cuentas de Grupo

Tu nivel de acceso a un sistema no está determinado únicamente por tu cuenta de usuario. Cada usuario puede ser miembro de uno o más grupos, lo que también puede afectar tu nivel de acceso al sistema.

Tradicionalmente, los sistemas UNIX limitaban a los usuarios para que pertenecieran a no más de un total de dieciséis grupos, pero los últimos núcleos de Linux soportan usuarios con más de sesenta y cinco mil membresías .

Como ya hemos visto, el archivo `/etc/passwd` define la pertenencia al grupo primario de un usuario. La pertenencia a un grupo suplementario (o pertenencia a un grupo secundario), así como los propios grupos, se definen en el archivo `/etc/group`.

Para ver la información sobre un grupo específico, puede utilizarse el comando `grep` o `getent`. Por ejemplo, los siguientes comandos mostrarán la información de cuenta de grupo `mail`:

```
sysadmin@localhost:~$ grep mail /etc/group
mail:x:12:mail,postfix
sysadmin@localhost:~$ getent group mail
mail:x:12:mail,postfix
```

13.3.1 El Archivo `/etc/group`

El archivo `/etc/group` es un archivo delimitado por dos puntos con los siguientes campos:

```
group_name:password_placeholder:GID:user_list
```

La siguiente tabla describe los campos del archivo `/etc/group` con más detalle, utilizando una línea que describe a una cuenta de grupo típica:

```
mail:x:12:mail,postfix
```

| Campo      | Ejemplo | Descripción   |
|------------|---------|---|
| group_name | mail    | Este campo contiene el nombre del grupo. Igual que en el caso de los nombres de usuario, para las personas es más fácil |

| Campo                             | Ejemplo                   | Descripción   |
|-----------------------------------|---------------------------|---|
|                                   |                           | recordar los nombres de grupo. El sistema utiliza típicamente IDs de grupos (GID) en lugar de nombres de grupo.   |
| <code>password_placeholder</code> | <code>x</code>            | Aunque hay contraseñas para grupos, raramente se utilizan en Linux. Si el administrador creará una contraseña de grupo, se almacenaría en un archivo diferente ( <code>/etc/gshadow</code> ) ya que la contraseña de grupo ya no se almacena en el archivo <code>/etc/group</code> . La <code>x</code> en este campo se utiliza para indicar que la contraseña no se almacena en este archivo. Las contraseñas de grupo están más allá del alcance de este curso. |
| GID                               | 12                        | Cada grupo está asociado con un ID de grupo único (GID) que se coloca en este campo.  |
| <code>user_list</code>            | <code>mail,postfix</code> | Este último campo se utiliza para indicar quién es un miembro del grupo. Mientras que la pertenencia a un grupo primario se define en el archivo <code>/etc/passwd</code> , los usuarios que se asignan a los   |



| Campo | Ejemplo | Descripción  |
|-------|---------|--|
|       |         | grupos adicionales tendrían su nombre de usuario en este campo del archivo <code>/etc/group</code> .   |
|       |         | En este caso, los usuarios <code>mail</code> y <code>postfix</code> son miembros secundarios del grupo <code>mail</code> .                                   |
|       |         | Es muy común para un nombre de usuario aparecer también como un nombre del grupo. También es común que un usuario pertenezca a un grupo con el mismo nombre. |

### 13.3.2 Visualizando la Información del Grupo

Los grupos se utilizan principalmente para controlar el acceso a los archivos. De forma predeterminada, cualquier nuevo archivo que un usuario crea, será propiedad del grupo primario del usuario. El comando `id` se puede utilizar para verificar el grupo primario del usuario:

```
sysadmin@localhost:~$ id -g
1001
```

Los usuarios también pueden acceder a los archivos y dispositivos si son miembros de los grupos secundarios. El comando `id` se puede utilizar también para verificar la pertenencia a los grupos secundarios del usuario:

```
sysadmin@localhost:~$ id -G
1001 4 27
```

Sin pasar opciones al comando, `id` listará tanto la pertenencia a los grupos primarios como a los grupos secundarios:

```
sysadmin@localhost:~$ id
uid=1001(sysadmin) gid=1001(sysadmin) groups=1001(sysadmin),4(adm),27(sudo)
```

Se puede agregar un nombre de usuario para determinar los grupos de un usuario específico:

```
sysadmin@localhost:~$ id sysadmin
uid=1001(sysadmin) gid=1001(sysadmin) groups=1001(sysadmin),4(adm),27(sudo)
```

Esto se alinea con el contenido del archivo `/etc/group`, cuando se revela la búsqueda para el `sysadmin`:

```
sysadmin@localhost:~$ cat /etc/group | grep sysadmin
adm:x:4:sysadmin
sudo:x:27:sysadmin
sysadmin:x:1001:
```

### 13.3.3 Cambiando la Propiedad de Grupo de un Archivo Existente

Para cambiar al grupo propietario de un archivo existente se puede utilizar el comando `chgrp group_name file` «o `chgrp nombre_de_grupo archivo`». Los usuarios sólo pueden cambiar la propiedad de los archivos que poseen. El nuevo grupo propietario del archivo también debe ser un grupo al que pertenece el usuario:

```
sysadmin@localhost:~$ touch sample
sysadmin@localhost:~$ ls -l sample
-rw-rw-r-- 1 sysadmin sysadmin 0 Dec 10 00:44 sample
sysadmin@localhost:~$ chgrp games sample
-rw-rw-r-- 1 sysadmin games 0 Oct 23 22:12 sample
sysadmin@localhost:~$
```

Para cambiar el grupo propietario de todos los archivos en una estructura de directorios, utiliza la opción `-R` para el comando `chgrp`. Por ejemplo, el comando `chgrp -R games test_dir` cambiaría la propiedad del grupo en el directorio `test_dir` y de todos los archivos y subdirectorios del directorio `test_dir`.

También existe un comando `chown` que se puede utilizar para cambiar al usuario propietario de un archivo o directorio. Sin embargo, este comando lo puede utilizar solamente el usuario `root`. Los usuarios normales no pueden «dar» sus archivos a otro usuario.

## 13.4 Iniciar Sesión como Root

Hay muchas formas de ejecutar un comando que requiere privilegios de administrador o `root`. Como ya se mencionó, iniciar sesión en el sistema como usuario `root` te permite ejecutar los comandos como administrador. Esto es potencialmente peligroso porque se te puede olvidar que estás registrado como `root` y ejecutar un comando que podría causar problemas en el sistema. Como resultado, se recomienda no iniciar sesión como usuario `root` directamente.

Porque usar la cuenta `root` es potencialmente peligrosa, sólo debe ejecutar comandos como `root` si se necesitan privilegios administrativos. Si la cuenta `root` está desactivada, tal como sucede en la distribución de Ubuntu, entonces los comandos administrativos pueden ser ejecutados utilizando el comando `sudo`. Si la cuenta de `root` está activada, un usuario normal puede ejecutar el comando `su` para cambiar a la cuenta `root`.

Cuando inicias sesión en el sistema directamente como `root` para ejecutar comandos, todo acerca de tu sesión se ejecuta como usuario `root`. Si utilizas el entorno gráfico, esto es especialmente peligroso ya que el proceso de inicio de sesión gráfico se compone de muchos ejecutables diferentes (programas que se ejecutan durante el

inicio de sesión). Cada programa que se ejecuta como usuario root representa una amenaza mayor que un proceso ejecutado como un usuario estándar, ya que tales programas tendrían permisos para hacer casi cualquier cosa, mientras que los programas de usuario estándar son muy restringidos en lo que pueden hacer.

El otro peligro potencial con el inicio de sesión en el sistema como root es que una persona que hace esto puede olvidarse cerrar la sesión para hacer su trabajo no administrativo. Esto significa que los programas como navegadores, clientes de correo electrónico, etc. se ejecutarían como usuario root sin restricciones en lo que podrían hacer. El hecho de que varias distribuciones de Linux, particularmente Ubuntu, no permiten a los usuarios iniciar sesión como usuario root debe ser una implicación suficiente que ésta no es la forma preferida para realizar las tareas administrativas.

### 13.5 Usando el Comando su

El comando `su` te permite ejecutar el shell como un usuario diferente. Por defecto, si no se especifica una cuenta de usuario, el comando `su` abre un nuevo shell como usuario root. Aunque el comando `su` es más frecuentemente usado para cambiar al usuario root, también puede cambiar a otros usuarios.

Una opción común que se utiliza con el comando `su` es la opción `-l`, que hace que el nuevo shell sea un shell de inicio de sesión. Utilizando el comando `su` con una opción de shell de inicio de sesión es a menudo importante para asegurar que cualquier comando ejecutado se ejecute correctamente, ya que el shell del inicio de sesión configura totalmente el nuevo shell con la configuración del nuevo usuario. Un non-login shell sólo cambia el UID, pero no conecta al usuario completamente. La opción `-l` puede ser abreviada simplemente como `-` o deletreada como `--login`.

Ya que la cuenta root se utiliza de forma predeterminada con el comando `su`, los siguientes dos comandos son maneras equivalentes de iniciar un shell como usuario root:

```
su - root
su -
```

Después de presionar **Entrar** para ejecutar uno de estos comandos, el usuario debe proporcionar la contraseña del usuario root para iniciar el shell como usuario root. Si no sabes la contraseña de la cuenta a la que te estás moviendo, entonces el comando `su` fallará.

Después de usar el shell iniciado con el comando `su` para realizar las tareas administrativas necesarias, vuelve a tu shell original (y a la cuenta de usuario original) mediante el comando `exit`.

```
sysadmin@localhost:~$ su -
Password:
root@localhost:~# id
uid=0(root) gid=0(root) groups=0(root)
root@localhost:~# exit
logout
sysadmin@localhost:~$ id
uid=1001(sysadmin) gid=1001(sysadmin) groups=1001(sysadmin),4(adm),27(sudo)
sysadmin@localhost:~$
```

### 13.6 Utilizando el comando sudo

En las distribuciones que no permiten que el usuario root acceda directamente o a través del comando `su`, el proceso de instalación configura automáticamente una cuenta de usuario para poder utilizar el comando `sudo` para ejecutar los comandos como si fueran ejecutados por el usuario root.

Igual que el comando `su`, el comando `sudo` asume por defecto que la cuenta del usuario root debe utilizarse para ejecutar comandos; para especificar una cuenta de usuario diferente, utiliza la opción `-u`.

Cuando utilizas el comando `sudo` para ejecutar un comando como usuario root, el comando pedirá la contraseña del usuario (no la del usuario root). Esta es una característica de seguridad que impide el acceso root no autorizados si el usuario deja su computadora desatendida. El prompt de la contraseña no aparecerá de nuevo mientras el usuario continúa ejecutando comandos `sudo` con menos de cinco minutos de diferencia.

Utilizar el comando `sudo` para ejecutar un comando de administración resultará en una entrada en un archivo de registro. Esta entrada incluye el nombre de usuario que ejecuta el comando, el comando ejecutado y la fecha y hora cuando se ejecutó el comando. Esto permite mayor rendición de cuentas en comparación con un sistema donde muchos usuarios saben la contraseña de root y pueden entrar directamente como root o usar el comando `su` para ejecutar los comandos como el usuario root.

Puedes utilizar el comando `sudo` para ejecutar un comando que requiera privilegios de root. Por ejemplo, es necesario ser un usuario root para poder ver el archivo `/etc/shadow`. El comando `sudo head /etc/shadow` ejecuta el comando `head` como un usuario root:

```
sysadmin@localhost:~$ head /etc/shadow
head: cannot open `/etc/shadow' for reading: Permission denied
sysadmin@localhost:~$ sudo head /etc/shadow
[sudo] password for sysadmin:
root:$6$4Yga95H9$8HbxqsMEIBTZ0YomlMffYCV9VE1SQ4T2H3SHXw41M02SQtfAdDVE9mqGp2hr20q
.ZuncJpLyWkYwQdKlSJyS8.:16464:0:99999:7:::
daemon*:16463:0:99999:7:::
bin*:16463:0:99999:7:::
sys*:16463:0:99999:7:::
sync*:16463:0:99999:7:::
games*:16463:0:99999:7:::
man*:16463:0:99999:7:::
lp*:16463:0:99999:7:::
mail*:16463:0:99999:7:::
news*:16463:0:99999:7:::
sysadmin@localhost:~$
```

Una gran ventaja por usar `sudo` para ejecutar los comandos administrativos es que reduce el riesgo de que un usuario accidentalmente ejecute un comando como root. La intención de ejecutar un comando es clara; el comando se ejecuta como root si tiene como prefijo el comando `sudo`. De lo contrario, se ejecuta el comando como un usuario normal.

### 13.6.1 Configurar el Comando `sudo`

Si el comando `sudo` no es configurado automáticamente durante la instalación, se puede configurar manualmente después de la instalación. Inicialmente, esto requerirá iniciar sesión como usuario `root` (o usando el comando `su` para cambiar a la cuenta de `root`), pero después de su configuración, los usuarios especificados podrán ejecutar los comandos `sudo`. Puedes configurar el acceso al comando `sudo` ejecutando el comando `visudo`.

El comando `visudo` te pasa al programa editor, por defecto el editor `vi` (o `vim`) en la mayoría de los sistemas, el que puede ser un reto para los usuarios principiantes Linux. Para configurar otro editor, exporta la variable `EDITOR` con el valor del editor que prefieres. Por ejemplo, para usar el editor `gedit` en vez de `vi/vim`, ejecutarías el comando `export EDITOR=gedit` antes de ejecutar el comando `visudo`.

Utilizando el comando `visudo` se abrirá el archivo de configuración `/etc/sudoers` para editar. Configuraciones avanzadas del comando `sudo` se pueden realizar a través de este archivo, pero están más allá del alcance de este curso. Básicamente, en este archivo se realizan las entradas para especificar qué usuario (usuarios) y en cuáles de las máquinas pueden utilizar el comando `sudo` para ejecutar los comandos como otro usuario.

Esta entrada predeterminada...

```
root    ALL=(ALL)    ALL
```

...se puede leer como, «el usuario `root` puede actuar como TODOS los usuarios en TODAS las máquina para ejecutar TODOS los comandos.» Para que un usuario como el `sysadmin` ejecute todos los comandos como todos los usuarios utilizando el comando `sudo`, podría añadirse una entrada similar a la siguiente:

```
sysadmin ALL=(ALL) ALL
```

13.7 Utilizando el Comando `who`

El comando `who` muestra una lista de usuarios que están conectados actualmente en el sistema, desde dónde están conectados y cuándo iniciaron sesión. Mediante el uso de opciones este comando puede mostrar información como el *nivel de ejecución* actual (un estado funcional de la computadora) y la hora del arranque del sistema.

Por ejemplo:

```
sysadmin@localhost:~$ who
root      tty2      2013-10-11 10:00
sysadmin  tty1      2013-10-11 09:58 (:0)
sysadmin  pts/0     2013-10-11 09:59 (:0.0)
sysadmin  pts/1     2013-10-11 10:00 (example.com)
```

La tabla siguiente describe la salida del comando `who`:

| Columna  | Ejemplo | Descripción  |
|----------|---------|--|
| username | root    | Esta columna indica el nombre del usuario que inició la sesión. Ten en cuenta que por "sesión" nos referimos a «cualquier proceso de inicio de |

| Columna  | Ejemplo   | Descripción   |
|----------|---|---|
|          |   | sesión y cualquier ventana de la terminal abierta».   |
| terminal | <code>tty2</code>   | <p>Esta columna indica en qué ventana de terminal está trabajando el usuario.</p> <p>Si el nombre de la terminal inicia con <code>tty</code>, entonces esto es una indicación de un inicio de sesión local, está es una terminal de línea de comandos regular.</p> <p>Si el nombre de la terminal inicia con <code>pts</code>, entonces esto indica que el usuario está usando una pseudo terminal o corriendo un proceso que actúa como la terminal. Esto puede significar que el usuario tiene una aplicación de terminal corriendo en X Windows, como <code>gnome-terminal</code> o <code>xterm</code> o pueden haber usado un protocolo de red para conectarse al sistema, como <code>ssh</code> o <code>telnet</code>.</p> |
| date     | <code>2013-10-11</code><br><code>10:00</code><br><code>(example.com)</code> | <p> </p> <p>Esto indica cuándo inició sesión el usuario.</p> <p>Después de la fecha y la hora en que el usuario inició sesión en el sistema, puede aparecer alguna información de localización.</p> <p>Si la información de localización contiene un nombre de host, nombre de dominio o dirección IP, entonces el usuario inició sesión remotamente.</p>   |

| Columna | Ejemplo | Descripción   |
|---------|---------|---|
|         |         | Si hay dos puntos y un número, entonces esto indica que han hecho un inicio de sesión gráfico local.  |
|         |         | Si no se muestra información de localización en la última columna, entonces esto significa que el usuario inició sesión mediante un proceso de línea de comandos local. |

Si quieres visualizar la información sobre el estado del sistema, el comando `who` puede hacerlo utilizando varias opciones. Por ejemplo, la opción `-b` mostrará la última vez que el sistema se inició (fue arrancado) y la opción `-r` mostrará el tiempo en el cual el sistema haya alcanzado un cierto nivel ejecución:

```
sysadmin@localhost:~$ who -b -r
      system boot    2013-10-11 09:54
      run-level 5    2013-10-11 09:54
```

### 13.8 Usando el Comando w

El comando `w` proporciona una lista más detallada sobre los usuarios que actualmente están en el sistema que el comando `who`. También proporciona un resumen del estado del sistema. Por ejemplo:

```
sysadmin@localhost:~$ w
 10:44:03 up 50 min,  4 users,  load average: 0.78, 0.44, 0.19
USER      TTY      FROM          LOGIN@   IDLE   JCPU   PCPU   WHAT
root      tty2      -             10:00   43:44   0.01s   0.01s   -bash
sysadmin   tty1      :0            09:58   50:02   5.68s   0.16s   pam: gdm-password
sysadmin   pts/0     :0.0          09:59   0.00s   0.14s   0.13s   ssh 192.168.1.2
sysadmin   pts/1     example.com   10:00   0.00s   0.03s   0.01s   w
```

La primera línea de la salida del comando `w` es idéntica a la comando `uptime`. Muestra la hora actual, cuánto tiempo el sistema ha estado funcionando, el número total de inicios de sesión actuales (usuarios) y el promedio de la carga en el sistema en los últimos 1, 5 y 15 minutos. El *promedio de carga* es el uso de la CPU donde un valor de 100 significaría un uso completo de la CPU durante ese periodo de tiempo.

La tabla siguiente describe el resto de la salida del comando `w`:

| Column | Ejemplo     | Descripción   |
|--------|-------------|---|
| USER   | root        | Esta columna indica el nombre del usuario que inició la sesión.   |
| TTY    | tty2        | Esta columna indica en qué ventana de terminal el usuario está trabajando.  |
| FROM   | example.com | Desde dónde inició sesión el usuario.   |
| LOGIN@ | 10:00       | Cuándo inició sesión el usuario.  |
| IDLE   | 43:44       | Cuánto tiempo el usuario ha estado inactivo desde la ejecución del último comando.  |
| JCPU   | 0.01s       | El tiempo total de cpu ( <i>s</i> =segundos) utilizado por todos los procesos (programas) ejecutados desde el inicio de sesión. |
| PCPU   | 0.01s       | El tiempo total de cpu para el proceso actual.  |
| WHAT   | -bash       | El proceso actual que está ejecutando el usuario.   |

**NDG Linux Essentials: Capítulo 14: Crear un nuevo usuario.**

## 14.1 Introducción



En el capítulo anterior viste que la información de la cuenta de usuario se almacena en el archivo `/etc/passwd` y la información de autenticación de usuario (datos de la contraseña) se almacena en el archivo `/etc/shadow`. La creación de un nuevo usuario puede lograrse añadiendo una nueva línea manualmente a cada uno de estos archivos, pero generalmente no es la técnica recomendada. Mediante el uso de un comando adecuado para agregar un nuevo usuario, estos archivos pueden ser modificados automáticamente y de forma segura. Si fueras a modificar manualmente estos archivos, corres el riesgo de cometer un error que podría impedir el acceso normal de todos los usuarios.

En algunas distribuciones, creando una nueva cuenta de usuario automáticamente se crea una cuenta de grupo para el usuario, llamado un *Grupo Privado de Usuario* (UPG «User Private Group» en inglés). En estos sistemas, el grupo y nombre de usuario serían los mismos y el único miembro de este nuevo grupo sería el nuevo usuario.

Para las distribuciones que no crean un UPG, los nuevos usuarios reciben el grupo de "users" («usuarios» en español) como su grupo primario. El administrador puede crear manualmente cuentas de grupo que sean privadas para el usuario, pero es más común que el administrador cree grupos para varios usuarios que deben colaborar. Las cuentas de usuario pueden ser modificadas en cualquier momento para agregar o quitar membresías de cuenta de grupo, pero los usuarios deben pertenecer por lo menos a un grupo para usarlo como su grupo primario.

Antes de empezar a crear los usuarios, debes planear cómo vas a utilizar los grupos. Los usuarios pueden crearse con membresías en grupos que ya existen o los usuarios existentes pueden ser modificados para tener membresías en grupos existentes.

Si ya tienes previsto qué usuarios y grupos quieres crear, es más eficiente primero crear sus grupos y después crear los usuarios con sus pertenencias a grupos. Por otra parte, si creas primero tus usuarios, y luego sus grupos, necesitarás un paso extra para modificar los usuarios para que sean miembros de sus grupos.

*¿Quién es LPI? El Linux Professional Institute (o «Instituto Profesional de Linux» en español) es una organización comprometida en ayudar miembros de la comunidad de Linux y de código abierto, crecer en sus oportunidades profesionales, ofreciendo recursos profesionales y certificaciones de habilidades.*

## 14.2 Crear un Grupo

La razón más común para crear un grupo es que los usuarios puedan compartir archivos. Un ejemplo de esto podría ser cuando hay varias personas trabajando juntas en el mismo proyecto y necesitan colaborar en documentos almacenados en archivos para el proyecto. En este escenario, el administrador puede hacer a estas personas miembros de un grupo común, cambiar la propiedad del directorio al nuevo grupo y establecer permisos en el directorio que sólo permitirá a los miembros del grupo acceder a los archivos.

Después de crear o modificar un grupo, puedes verificar los cambios mediante la visualización de la información de configuración del grupo en el archivo `/etc/group` con el comando `grep`. Si trabajas con los servicios de autenticación de red, entonces el comando `getent` puede mostrar los grupos locales y en la red. Para uso local, estos comandos muestran el mismo resultado, en este caso para el grupo de root:

```
root@localhost:~# grep root /etc/group
root:x:0:
root@localhost:~# getent group root
root:x:0:
```

El comando `groupadd` puede ser ejecutado por el usuario root para crear un nuevo grupo. El comando requiere solamente el nombre del grupo que se creará. La opción `-g` puede utilizarse para especificar un `id` de grupo para el grupo nuevo:

```
root@localhost:~# groupadd -g 506 research
root@localhost:~# grep research /etc/group
```

```
research:x:506:
```

Si no se proporciona la opción `-g`, el comando `groupadd` proporcionará automáticamente un GID para el grupo nuevo. Para lograr esto, el comando `groupadd` ve el archivo `/etc/group` y utiliza un número que es de un valor mayor que el mayor número GID actual. La ejecución de los comandos siguientes ilustra esto:

```
root@localhost:~# grep research /etc/group
research:x:506:
root@localhost:~# groupadd development
root@localhost:~# grep development /etc/group
development:x:507:
```

#### 14.2.1 Consideraciones para el ID de Grupo

En algunas distribuciones de Linux, particularmente las basadas en Red Hat, cuando se crea un ID de usuario (UID), también se crea un grupo privado de usuario (UPG) con ese usuario como único miembro. En estas distribuciones, el UID del usuario y el id de grupo privado deben coincidir (tener el mismo número).

Por lo tanto, no debes crear los GIDs en los mismos rangos numéricos donde se espera crear identificadores de usuario (UIDs), con el fin de evitar un conflicto entre un GID creado frente a un número UPG que se crea para que coincida con un UID.

Recuerda que los GID bajo 500 están reservados para uso del sistema. Puede haber ocasiones en las que quieras asignar un GID menor de 500. Para lograr esto, utiliza `-r`. La opción `-r` asignará un GID al nuevo grupo que será menor que el estándar más bajo de UID:

```
root@localhost:~# groupadd -r sales
root@localhost:~# getent group sales
sales:x:491:
```

#### 14.2.2 Consideraciones para Nombrar un Grupo

Seguir estas directrices para los nombres de grupo te ayudará a seleccionar un nombre de grupo que sea portable (funcione correctamente con otros sistemas o servicios):

- El primer carácter del nombre debe ser un guión bajo `_` o un carácter alfabético en minúsculas `a-z`.
- En la mayoría de las distribuciones de Linux se permite hasta 32 caracteres, pero usar más de 16 puede ser problemático, ya que algunas distribuciones no pueden aceptar más de 16.
- Después del primer carácter, los caracteres restantes pueden ser alfanuméricos, un guión `-` o un guión bajo `_`.
- El último carácter no debe ser un guión `-`.

Lamentablemente estas pautas no se aplican siempre. El problema no es que el comando `groupadd` fracase, sino que otros comandos o servicios del sistema no funcionen correctamente.

### 14.3 Modificar un Grupo

El comando `groupmod` se puede utilizar para cambiar el nombre del grupo (con la opción `-n`) o cambiar el GID (con la opción `-g`) para el grupo.

**ADVERTENCIA:** Cambiar el nombre del grupo puede causar confusión para los usuarios que estaban familiarizados con el antiguo nombre y no hayan sido informados del nombre nuevo. Sin embargo, cambiar el nombre del grupo no causará problemas con el acceso a los archivos, ya que los archivos son propiedad del GID, no de los nombres de grupo. Por ejemplo:

```
root@localhost:~# ls -l index.html
-rw-r-----. 1 root sales 0 Aug  1 13:21 index.html

root@localhost:~# groupmod -n clerks sales

root@localhost:~# ls -l index.html
-rw-r-----. 1 root clerks 0 Aug  1 13:21 index.html
```

Después del comando `groupmod` anterior, el archivo `index.html` tiene un nombre de propietario de grupo diferente. Sin embargo, todos los usuarios que estaban en el grupo `sales` están ahora en el grupo `clerks`, así pues, todos los usuarios aún pueden acceder el archivo `index.html`. Una vez más, esto es porque el grupo se define por el GID, no por el nombre del grupo.

Por otro lado, si cambias el GID para un grupo, entonces todos los archivos que fueron asociados a ese grupo ya no estarán asociados a ese grupo. De hecho, todos los archivos que fueron asociados a ese grupo ya no estarán asociados con ningún nombre de grupo. Por el contrario, estos archivos serán propiedad de un GID solamente tal como se muestra a continuación:

```
root@localhost:~# groupmod -g 10003 clerks

root@localhost:~# ls -l index.html
-rw-r-----. 1 root 491 13370 Aug  1 13:21 index.html
```

Estos archivos sin nombre de grupo se denominan archivos «huérfanos». Como usuario `root`, probablemente quieras buscar todos los archivos que son propiedad de solamente un GID (no asociado con un nombre de grupo). Esto puede lograrse con la opción `-nogroup` para el comando `find`:

```
root@localhost:~# find / -nogroup
/root/index.html
```

## 14.4 Eliminando un Grupo

Si quieres eliminar un grupo con el comando `groupdel`, ten en cuenta que los archivos que pertenecen a ese grupo se convertirán en «huérfanos».

Sólo se puede eliminar a los grupos suplementarios, por lo que si un grupo es el grupo primario para cualquier usuario, no se puede eliminar. El administrador puede modificar qué grupo es el grupo primario del usuario, por lo que un grupo que estaba siendo utilizado como un grupo primario se puede transformar en un grupo suplementario y luego se puede eliminar.

Mientras que el grupo que se vaya a eliminar no sea el grupo principal del usuario, eliminar el grupo se logra mediante el comando `groupdel` junto con el nombre del grupo:

```
root@localhost:~# groupdel clerks
```

14.5 Archivo /etc/default/useradd

Antes de empezar a crear los usuarios para el sistema, debes verificar o establecer los valores prácticos que se utilizarán por defecto con el comando `useradd`. Esto se puede lograr modificando la configuración en los archivos de configuración utilizados por el comando `useradd`.

Asegurarse de que los valores en estos archivos de configuración sean razonables antes de agregar usuarios puede ayudarte a ahorrar tiempo y la molestia de tener que corregir la configuración de la cuenta de usuario después de agregar los usuarios.

La opción `-D` del comando `useradd` te permitirá visualizar o modificar algunos de los valores por defecto utilizados por el comando `useradd`. Los valores indicados por `useradd -D` también pueden visualizarse o actualizar mediante la manipulación del archivo `/etc/default/useradd`:

```
root@localhost:~# useradd -D
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
CREATE_MAIL_SPOOL=yes
```

La siguiente tabla describe cada uno de estos valores:

| Campo | Ejemplo | Descripción  |
|-------|---------|--|
| GROUP | 100     | <p>En las distribuciones que no usan UPG, este será el grupo principal de forma predeterminada para un usuario nuevo, si no se ha especificado uno con el comando <code>useradd</code>. Este normalmente es el grupo de «users» («usuarios» en español) con un GID de 100.</p> <p>Esta configuración afecta a la configuración por defecto del archivo <code>/etc/passwd</code> resaltado abajo:</p> <pre>bob:x:600:600:bob:/home/bob:/bin/bash</pre> <p>La opción <code>-g</code> para el comando <code>useradd</code> permite utilizar un grupo principal diferente al predeterminado, cuando se crea una nueva cuenta de usuario.</p> |

| Campo    | Ejemplo | Descripción   |
|----------|---------|---|
| HOME     | /home   | <p>El directorio <code>/home</code> es el directorio base predeterminado, en el cuál se creará un nuevo directorio home del usuario. Esto significa que un usuario con un nombre de cuenta de <code>bob</code> tendría un directorio de <code>/home/bob</code>.</p> <p>Esta configuración afecta a la configuración por defecto del archivo <code>/etc/passwd</code> resaltado abajo:</p> <pre>bob:x:600:600:bob:/home/bob:/bin/bash</pre> <p>La opción <code>-b</code> para el comando <code>useradd</code> permite utilizar un directorio base diferente al predeterminado, cuando se crea una nueva cuenta de usuario.</p>                                   |
| INACTIVE | -1      | <p>Este valor representa el número de días después de que caduca la contraseña hasta que la cuenta será deshabilitada. Un valor de <code>-1</code> significa que esta función no está habilitada por defecto y no se proporciona ningún valor «inactivo» para las nuevas cuentas por defecto.</p> <p>Esta configuración afecta a la configuración por defecto del archivo <code>/etc/shadow</code> resaltado abajo:</p> <pre>bob:pw:15020:5:30:7:60:15050:</pre> <p>La opción <code>-f</code> para el comando <code>useradd</code> permite utilizar un valor <code>INACTIVE</code> diferente al predeterminado, cuando se crea una nueva cuenta de usuario.</p> |
| EXPIRE   |         | <p>Por defecto, no hay ningún valor para la fecha de caducidad. Generalmente, una fecha de vencimiento se configura para una cuenta individual, no a todas las cuentas.</p>   |

| Campo | Ejemplo                | Descripción   |
|-------|------------------------|---|
|       |                        | <p>Por ejemplo, si tuvieras un contratista que fuese contratado para trabajar hasta el final del día 01 de noviembre de 2013, podrías asegurarte de que no pueda iniciar sesión después de esa fecha, utilizando el campo de <code>EXPIRE</code>.</p> <p>Esta configuración afecta a la configuración por defecto del archivo <code>/etc/shadow</code> resaltado abajo:</p> <pre>bob:pw:15020:5:30:7:60:15050:</pre> <p>La opción <code>-e</code> para el comando <code>useradd</code> permite utilizar un valor <code>EXPIRE</code> diferente al predeterminado, cuando se crea una nueva cuenta de usuario.</p> |
| SHELL | <code>/bin/bash</code> | <p>El valor de <code>SHELL</code> indica el shell por defecto para los usuarios cuando inician sesión en el sistema.</p> <p>Esta configuración afecta a la configuración por defecto del archivo <code>/etc/passwd</code> resaltado abajo:</p> <pre>bob:x:600:600:bob:/home/bob:/bin/bash</pre> <p>La opción <code>-s</code> para el comando <code>useradd</code> permite utilizar un shell de inicio de sesión diferente al predeterminado, cuando se crea una nueva cuenta de usuario.</p>  |
| SKEL  | <code>/etc/skel</code> | <p>El valor <code>SKEL</code> determina qué directorio «esqueleto» tendrá su contenido copiado en el directorio <code>home</code> de los usuarios nuevos. El contenido de este directorio se copia en el directorio <code>home</code> del usuario nuevo y el nuevo usuario recibe la propiedad de los nuevos archivos.</p> <p>Esto proporciona a los administradores una manera fácil de «rellenar» una nueva cuenta de usuario con los archivos de configuración clave.</p>  |

| Campo                          | Ejemplo          | Descripción  |
|--------------------------------|------------------|--|
|                                |                  | La opción <code>-k</code> para el comando <code>useradd</code> permite utilizar un directorio <code>SKEL</code> diferente al predeterminado, cuando se crea una nueva cuenta de usuario.   |
| <code>CREATE_MAIL_SPOOL</code> | <code>yes</code> | <p>El «mail spool» («carrete de correo» en español) es un archivo donde se coloca el correo entrante.</p> <p>Actualmente el valor para crear un mail spool es <code>yes</code>, lo que significa que los usuarios por defecto están configurados con la capacidad de recibir y guardar correo local. Si no piensas usar el correo local, este valor puede cambiarse a <code>no</code>.</p> |

Para modificar uno de los valores por defecto del `useradd`, el archivo `/etc/default/useradd` puede editarse con un editor de texto. Otra técnica (más segura) es usar el comando `useradd -D`.

Por ejemplo, si quieres permitir a los usuarios con una contraseña caducada seguir iniciando la sesión con un máximo de treinta días, puedes ejecutar lo siguiente:

```
root@localhost:~# useradd -D -f 30
root@localhost:~# useradd -D
GROUP=100
HOME=/home
INACTIVE=30
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
CREATE_MAIL_SPOOL=yes
```

## 14.6 Archivo `/etc/login.defs`

El archivo `/etc/login.defs` también contiene valores que se aplicarán por defecto a los nuevos usuarios que vayas a crear con el comando `useradd`. A diferencia del `/etc/default/useradd`, que puede ser actualizado con el comando `useradd -D`, el archivo `/etc/login.defs` generalmente lo edita directamente el administrador para modificar sus valores.

Este archivo contiene muchos comentarios y líneas en blanco, así que si quieres ver las líneas que no son comentarios o líneas en blanco (la configuración actual), puede utilizar el siguiente comando `grep`:

```
root@localhost:~# grep -Ev '^#|^$' /etc/login.defs
MAIL_DIR          /var/spool/mail
PASS_MAX_DAYS     99999
PASS_MIN_DAYS     0
PASS_MIN_LEN      5
PASS_WARN_AGE     7

UID_MIN           500
UID_MAX           60000
GID_MIN           500
GID_MAX           60000

CREATE_HOME       yes
UMASK             077
USERGROUPS_ENAB   yes
ENCRYPT_METHOD     SHA512
MD5_CRYPT_ENAB    no
```

El ejemplo anterior representa un típico archivo `/etc/login.defs` de la distribución CentOS 6 con sus valores. La siguiente tabla describe cada uno de estos valores:

| Campo         | Ejemplo         | Descripción   |
|---------------|-----------------|---|
| MAIL_DIR      | /var/mail/spool | El directorio en el que se crea el archivo mail pool del usuario.           |
| PASS_MAX_DAYS | 99999           | Esta configuración determina el número máximo de días en los que un usuario |



| Campo         | Ejemplo | Descripción   |
|---------------|---------|---|
|               |         | <p>podrá utilizar la misma contraseña. Puesto que por defecto viene configurado el valor de 99999 días o más de 200 años, significa que los usuarios nunca tienen que cambiar su contraseña.</p> <p>Las organizaciones con políticas eficaces para el mantenimiento de contraseñas seguras comúnmente cambian este valor a 60 o 30 días.</p> <p>Esta configuración afecta a la configuración por defecto del archivo <code>/etc/shadow</code> resaltado abajo:</p> <pre>bob:pw:15020:5:30:7:60:15050:</pre>   |
| PASS_MIN_DAYS | 0       | <p>Con esto configurado a un valor predeterminado de 0 (cero), el tiempo más corto que un usuario tiene que mantener una contraseña es de cero días, lo que significa que inmediatamente después de configurar la contraseña, se puede cambiar.</p> <p>Si el valor <code>PASS_MIN_DAYS</code> se estableció en 3 días, después de establecer una nueva contraseña, el usuario tendría que esperar tres días antes de que pueda cambiarla otra vez.</p> <p>Esta configuración afecta a la configuración por defecto del archivo <code>/etc/shadow</code> resaltado abajo:</p> <pre>bob:pw:15020:3:30:7:60:15050:</pre> |

| Campo         | Ejemplo | Descripción   |
|---------------|---------|---|
| PASS_MIN_LEN  | 5       | Esto indica el número mínimo de caracteres que debe contener una contraseña.  |
| PASS_WARN_AGE | 7       | <p>Este es el valor predeterminado para el campo de advertencia. Cuando un usuario se acerca al número máximo de días durante los que puede usar su contraseña, el sistema comprobará si es hora de empezar a avisar al usuario sobre cambiar su contraseña durante el inicio de sesión.</p> <p>Esta configuración afecta a la configuración por defecto del archivo <code>/etc/shadow</code> resaltado abajo:</p> <pre>bob:pw:15020:3:30:7:60:15050:</pre> |
| UID_MIN       | 500     | El <code>UID_MIN</code> determina el primer UID que se asignará a un usuario ordinario. Cualquier usuario con un UID menor que este valor, ya sea para una cuenta del sistema o bien para la cuenta de root.  |
| UID_MAX       | 60000   | Un UID técnicamente podría tener un valor de más de 4 billones. Para la máxima compatibilidad se recomienda dejarlo en su valor predeterminado de 60000.  |
| GID_MIN       | 500     | El <code>GID_MIN</code> determina el primer GID que se asignará a un grupo ordinario. Cualquier grupo con un GID menor que este valor, ya sea para un grupo del sistema o bien para el grupo de root.   |

| Campo           | Ejemplo | Descripción   |
|-----------------|---------|---|
| GID_MAX         | 60000   | Un GID igual que un UID técnicamente podría tener un valor de más de 4 billones. Cualquier valor que utilices para tu <code>UID_MAX</code> debes utilizar para <code>GID_MAX</code> para soportar UPG.  |
| CREATE_HOME     | yes     | Este valor determina si se crea o no un nuevo directorio para el usuario, al crear su cuenta.   |
| UMASK           | 077     | <p>Este <code>UMASK</code> funciona en el momento que se crea el directorio home del usuario; determinará cuáles serán los permisos predeterminados de este directorio. Utilizando el valor predeterminado de <code>077</code> para <code>UMASK</code> significa que sólo el usuario propietario tendrá algún tipo de permiso para acceder a su directorio.</p> <p><code>UMASK</code> se cubrirá en detalle en un capítulo posterior.</p> |
| USERGROUPS_ENAB | yes     | En las distribuciones que cuentan con un grupo privado para cada usuario, como se muestra en este ejemplo CentOS, el <code>USERGROUPS_ENAB</code> tendrá un valor de <code>yes</code> . Si no se utiliza la UPG en la distribución, entonces esto tendrá un valor <code>no</code> .   |
| ENCRYPT_METHOD  | SHA512  | El método de cifrado que se utiliza para cifrar las contraseñas de los usuarios en el archivo <code>/etc/shadow</code> . El valor de <code>ENCRYPT_METHOD</code> anula la configuración   |

| Campo          | Ejemplo | Descripción  |
|----------------|---------|--|
|                |         | de MD5_CRYPT_ENAB (véase el siguiente renglón).  |
| MD5_CRYPT_ENAB | no      | Este ajuste obsoleto originalmente permitía al administrador especificar el uso del cifrado de contraseñas MD5 en lugar del cifrado original DES. Ahora es reemplazado por el valor de ENCRYPT_METHOD. |

## 14.7 Crear un Usuario

Durante el proceso de instalación, la mayoría de los instaladores crean un usuario normal y ya sea da a este usuario el permiso para ejecutar comandos administrativos con `sudo` o bien requiere que se configure una contraseña de usuario root como parte del proceso de instalación. Esto significa que la mayoría de los sistemas Linux se configuran de tal manera que permitan que un usuario (no root) sin privilegios inicie sesión, y que tenga la capacidad de ejecutar los comandos como el usuario root, ya sea directa o indirectamente.

Si la computadora se va a utilizar por sólo una persona, puede ser suficiente tener sólo una cuenta de usuario normal. Sin embargo, si una computadora debe de ser compartida por varias personas, entonces es bueno tener una cuenta separada para cada persona que la utiliza. Hay varias ventajas para los individuos si tienen sus propias cuentas separadas:

- Las cuentas pueden utilizarse para conceder acceso selectivo a archivos o servicios. Por ejemplo, el usuario de cada cuenta tiene un directorio separado que generalmente no es accesible para otros usuarios.
- El comando `sudo` se puede configurar para conceder la capacidad de ejecutar comandos administrativos selectos. Si los usuarios están obligados a utilizar el comando `sudo` para llevar a cabo los comandos administrativos, el sistema registra cuando los usuarios realizaron tales comandos.
- Cada cuenta puede tener membresías de grupos y los derechos asociados con ellos, lo que permite una mayor flexibilidad de gestión.

### 14.7.1 Consideraciones de la Cuenta

Crear una cuenta de usuario para usarlo con un sistema Linux puede requerir que reúnas varias piezas de información. Mientras que todo lo que puede ser necesario es el nombre de la cuenta, probablemente quieras planificar el UID, el grupo principal, los grupos suplementarios, el directorio home, el esqueleto de directorio y el shell que vayas a utilizar. A la hora de planificar estos valores, considera lo siguiente:

**nombre de usuario:** El único argumento necesario para el comando `useradd` es el nombre que le quieras dar a la cuenta. Este nombre debe seguir las mismas pautas que vimos anteriormente en este capítulo para los nombres de grupo. Para resumir, debe tener 32 caracteres o menos, empezar con una letra en minúscula o un guión bajo y luego sólo debe contener letras minúsculas, números, guiones y guiones bajos. Si el usuario necesita tener acceso a múltiples sistemas, generalmente se recomienda que el nombre de la cuenta sea igual en esos sistemas. El nombre de la cuenta debe ser único para cada usuario.

**identificador de usuario (UID):** Una vez creado un usuario con un UID específico, el sistema generalmente incrementará el UID solo por uno para el siguiente usuario que vayas a crear. Si estás conectado a una red con otros sistemas, probablemente quieras asegurarte que este UID sea el mismo en todos los sistemas para ayudar a

proveer un acceso consistente. La opción `-u` para el comando `useradd` te permite especificar el número UID. Los UID por lo general pueden variar de cero a más de 4 mil millones, pero para mayor compatibilidad con los sistemas antiguos, el máximo valor recomendado para el UID es de 60,000.

Tal como habíamos visto previamente, el usuario root tiene un identificador de usuario (UID) de 0, lo que hace que la cuenta tenga privilegios especiales. Cualquier cuenta con un UID de cero actuaría como «administrador».

Las cuentas del sistema son las cuentas que generalmente se utilizan para ejecutar los servicios en segundo plano (llamados *daemons*). Al no tener los servicios ejecutados como usuario root, se limita la cantidad de daño que se puede hacer con una cuenta de servicio que ha sido comprometida. Las cuentas del sistema utilizadas por los servicios generalmente utilizarán los UID que están en el rango de «reservados». Una cuenta de sistema que es una excepción a esta regla es el usuario `nfsnobody`, que tiene un UID 65534.

El rango reservado utilizado para las cuentas de servicio se ha ampliado con el tiempo. Originalmente, era para los UID entre 1 y 99. La tendencia actual entre distribuciones es que las cuentas del sistema sean cualquier cuenta que tenga un UID entre 1 y 999, pero se sigue utilizando también el rango de 1-499.

Si estás configurando un nuevo sistema, es una buena práctica a empezar tus UID no inferiores a 1000. Esto también tiene la ventaja para asegurar que tendrás suficientes UID disponibles para muchos servicios del sistema y darte la habilidad de crear muchos GID en el rango «reservado».

**grupo primario:** En las distribuciones que utilizan UPG, este grupo se creará automáticamente con un GID y un nombre de grupo que coincida con el UID y el nombre de usuario de la cuenta de usuario recién creado. En las distribuciones que no usan UPG, el grupo primario normalmente por defecto es el grupo de «users» («usuarios» en inglés) con un GID de 100. Para especificar un grupo primario con el comando `useradd`, utiliza la opción `-g` con el nombre o el GID del grupo.

**grupo(s) suplementario(s):** Si quieres que el usuario fuera miembro de uno o más grupos suplementarios, la opción `-G` se puede utilizar para especificar una lista separada por comas de nombres de grupos o números.

**directorio home:** Por defecto, la mayoría de las distribuciones crearán el directorio home del usuario con el mismo nombre que la cuenta de usuario bajo `/home`. Por ejemplo, si creas una cuenta de usuario llamada `sam`, el nuevo directorio home del usuario sería `/home/sam`. Hay varias opciones para el comando `useradd` que pueden afectar la creación del directorio home del usuario:

- La opción `-b` te permite especificar un directorio diferente bajo el cual se creará el directorio home del usuario. Por ejemplo: `-b /test` resultaría en `/test/sam` siendo el directorio home para una cuenta de usuario llamada `sam`.
- La opción `-d` te permite especificar un directorio existente o un nuevo directorio para el usuario. Esto debe ser una ruta de acceso completa para el directorio home del usuario. Por ejemplo: `-d /home/sam`.
- La opción `-m` le dice a `useradd` que cree el directorio home; esto no es normalmente necesario ya que es el comportamiento predeterminado del comando `useradd`. Sin embargo, cuando utilices la opción `-k` (véase abajo) para especificar un esqueleto de directorio diferente, entonces necesitarás la opción `-m`.
- La opción `-M` se utiliza para especificarle al comando `useradd` que no debe crear el directorio home.

**esqueleto de directorio:** De forma predeterminada, el contenido del directorio `/etc/skel` se copia en el directorio home del usuario nuevo. Los archivos resultantes también son propiedad del nuevo usuario. Usando la opción `-k` con el comando `useradd`, el contenido de un directorio diferente se puede utilizar para rellenar el directorio home de un usuario nuevo. Cuando se especifica el directorio esqueleto con la opción `-k`, debe utilizarse la opción `-m`, de lo contrario el comando `useradd` fallará mostrando un error que dice: «-k flag is only allowed with the -m flag» (o «La opción `-k` solo se permite con la opción `-m`» en español).

**shell:** Mientras el shell por defecto se especifica en el archivo `/etc/default/useradd`, también puede ser cambiado con la opción de `useradd -s` en el momento de la creación de cuentas. Más tarde, el administrador puede usar la opción `usermod -s` para cambiar el shell o el usuario puede cambiar su shell con el comando `chsh`. Es común especificar el shell `/sbin/nologin` para las cuentas que se vayan a utilizar como cuentas del sistema.

**comentario:** El campo de comentario, originalmente llamado el campo General Electric Comprehensive Operating System (GECOS), normalmente se utiliza para contener el nombre completo del usuario. Muchos programas gráficos muestran el valor de este campo en lugar del nombre de la cuenta. La opción `-c` de los comandos `useradd` y `usermod` permite especificar el valor de este campo.

### 14.7.2 El Comando `useradd`

Una vez hayas comprobado qué valores se utilizarán por defecto y hayas reunido la información sobre el usuario, entonces estás listo para crear una cuenta de usuario. Un ejemplo de un comando `useradd` usando algunas opciones sería el siguiente:

```
root@localhost:~# useradd -u 1000 -g users -G wheel,research -c 'Jane Doe' jane
```

Este ejemplo del comando `useradd` crea un usuario con UID de 1000, un grupo primario de `users` («usuarios»), membresías suplementarias en los grupos `wheel` y `research`, un comentario de «Jane Doe» y un nombre de cuenta `jane`.

La información sobre la cuenta de usuario de `jane` se agregará automáticamente a los archivos `/etc/passwd` y `/etc/shadow`, mientras que la información sobre el acceso a grupos suplementarios de `jane` se añadirá automáticamente al archivo `/etc/group` y `/etc/gshadow`. Por ejemplo:

```
root@localhost:~# useradd -u 1000 -g users -G wheel,research -c "Jane Doe" jane
root@localhost:~# grep jane /etc/passwd
jane:x:1000:100:Jane Doe:/home/jane:/bin/bash
root@localhost:~# grep jane /etc/shadow
jane:!!:16003:0:99999:7:::
root@localhost:~# grep jane /etc/group
wheel:x:10:jane
research:x:2000:jane
root@localhost:~# grep jane /etc/gshadow
wheel:::jane
research:!:jane
root@localhost:~#
```

Ten en cuenta que la cuenta aún no tiene una contraseña válida!

Además, se crearía el archivo mail spool `/var/spool/mail/jane`, el directorio `/home/jane` se crearía con permisos de sólo permitir el acceso al usuario `jane` y el contenido del directorio `/etc/skel` se copiaría en el directorio:

```
root@localhost:~# ls /var/spool/mail
jane root rpc sysadmin
root@localhost:~# ls /home
jane sysadmin
root@localhost:~# ls -a /home/jane
.  ..  .bash_logout  .bashrc  .profile  .selected_editor
```

```

root@localhost:~# ls -a /etc/skel
.  ..  .bash_logout  .bashrc  .profile  .selected_editor
root@localhost:~#

```

## 14.8 Elegir una Contraseña

Elegir una buena contraseña no es una tarea fácil, pero es fundamental que se haga correctamente, ya que la seguridad de una cuenta (tal vez todo el sistema) puede verse comprometida. Escoger una buena contraseña es únicamente un comienzo; tienes que ser cuidadoso con tu contraseña para que otras personas no la puedan ver. Nunca debes decir a nadie tu contraseña y nunca dejes que alguien te vea escribir tu contraseña. Si quieres apuntar tu contraseña, entonces la debes guardar en un lugar seguro como una caja fuerte.

¡Es fácil crear una mala contraseña! Si utilizas cualquier información en tu contraseña que tenga que ver contigo, entonces otras personas pueden llegar a conocerla o descubrir tal información, por lo tanto tu contraseña podría verse fácilmente comprometida. Tu contraseña nunca debe contener información sobre ti o alguien que conoces, tales como:

- nombre
- segundo nombre
- apellido
- cumpleaños
- teléfono
- nombres de mascotas
- licencia de conducir
- seguro social

Hay numerosos factores a considerar al elegir una contraseña para una cuenta:

- **Longitud:** El archivo `/etc/login.defs` permite al administrador especificar la longitud mínima de la contraseña. Aunque hay personas que consideran que una contraseña larga es mejor, esto no es realmente correcto. El problema con las contraseñas que son demasiado largas es que no se recuerdan fácilmente y, consecuentemente, la gente las apunta a menudo en un lugar donde pueden ser fácilmente encontradas y comprometidas.
- **Composición:** Una buena contraseña debe estar compuesta por una combinación de caracteres alfabéticos, numéricos y simbólicos.
- **Vigencia:** La cantidad de tiempo que una contraseña se puede utilizar como máximo debe ser limitada por varias razones:
  - Si una cuenta está comprometida y se limita el tiempo que la contraseña es válida, el intruso perderá el acceso puesto que eventualmente la contraseña se volverá invalidada.
  - Si una cuenta no se usa, entonces se puede desactivar automáticamente cuando la contraseña no sea válida.
  - Si los atacantes intentan atacar con «fuerza bruta» con cada contraseña posible, entonces la contraseña puede cambiarse antes de que el ataque tenga éxito.

Sin embargo, el hecho de requerir a un usuario que cambie su contraseña a menudo podría plantear problemas de seguridad, incluyendo:

- La calidad de la contraseña que el usuario elija, podría ser inferior.

- El usuario puede empezar a apuntar su contraseña en papel, lo que aumenta la posibilidad de que la contraseña sea descubierta.
- Las cuentas de usuario rara vez utilizadas se pueden caducar y requerir atención administrativa para reiniciar.

Las opiniones varían sobre la frecuencia con la que los usuarios deben ser forzados a cambiar sus contraseñas. Para cuentas muy sensibles, se recomienda cambiar las contraseñas con mayor frecuencia, como cada 30 días. Por otro lado, para cuentas no críticas sin ningún tipo de acceso a información sensible, hay menos necesidad de cambio frecuente. Para cuentas con mínimo riesgo, una vigencia razonable sería 90 días.

#### 14.9 Establecer una Contraseña de Usuario

Existen varias formas de cambiar una contraseña de usuario: el usuario puede ejecutar el comando `passwd`, el administrador puede ejecutar el comando `passwd` proporcionando el nombre de usuario como argumento y las herramientas gráficas también están disponibles.

El administrador puede utilizar el comando `passwd` para cambiar la contraseña de la cuenta o establecer una contraseña inicial. Por ejemplo, si el administrador hubiera creado la cuenta `jane`, entonces ejecutando `passwd jane` proporcionaría al administrador un prompt para configurar la contraseña para `jane`. Si se completa con éxito, el archivo `/etc/shadow` se actualizará con la nueva contraseña del usuario.

Mientras que los usuarios regulares deben seguir muchas reglas de contraseña, el usuario `root` solo debe seguir una regla: la contraseña no se puede dejar en blanco. Todas otras reglas de contraseña que el usuario `root` no cumpla, simplemente resultarán en una advertencia que se imprime a la pantalla y la regla no se aplica:

```
root@localhost:~# passwd jane
Enter new UNIX password:
BAD PASSWORD: it is WAY to short
BAD PASSWORD: is too simple
Retype new UNIX password:
passwd: password updated successfully
root@localhost:~#
```

Suponiendo que el administrador estableció una contraseña para una cuenta de usuario, el usuario puede entonces iniciar sesión con el nombre de cuenta y la contraseña. Cuando el usuario abre una terminal, puede ejecutar el comando `passwd` sin argumentos para cambiar su propia contraseña. Se le pide su contraseña actual y luego se pedirá que introduzca la nueva contraseña dos veces.

Para un usuario común puede ser difícil establecer una contraseña válida, porque debe seguir todas las reglas para la contraseña. El usuario normalmente tiene tres intentos para proporcionar una contraseña válida antes de que el comando `passwd` salga con un error.

Con los privilegios del usuario `root`, las contraseñas encriptadas y otra información relacionada a la contraseña puede verse consultando el archivo `/etc/shadow`. Hay que recordar que los usuarios normales no pueden ver el contenido de este archivo.

#### 14.10 Usando el Comando `chage`

Aunque no aparezca como un comando que deber saber según los objetivos del curso, el comando `chage` ofrece muchas opciones para la gestión de la información de vencimiento de contraseña que se encuentra en el archivo `/etc/shadow`.

Aquí está un resumen de las opciones de `chage`:



| Opción corta                    | Opción larga                              | Descripción  |
|---------------------------------|---|--|
| <code>-l</code>                 | <code>--list</code>                       | Listar la información de vencimiento de la cuenta  |
| <code>-d LAST_DAY</code>        | <code>--lastday<br/>LAST_DAY</code>       | Fijar la fecha del último cambio de contraseña a <code>LAST_DAY</code>   |
| <code>-E<br/>EXPIRE_DATE</code> | <code>--expiredate<br/>EXPIRE_DATE</code> | Configurar cuenta para que expire el <code>EXPIRE_DATE</code>  |
| <code>-h</code>                 | <code>--help</code>                       | Mostrar la ayuda para <code>chage</code>   |
| <code>-I INACTIVE</code>        | <code>--inactive<br/>INACTIVE</code>      | Configurar la cuenta para permitir acceso <code>INACTIVE</code> días después de que la contraseña caduque.             |
| <code>-m MIN_DAYS</code>        | <code>--mindays<br/>MIN_DAYS</code>       | Definir el número mínimo de días antes de que se pueda cambiar la contraseña a <code>MIN_DAY</code>                    |
| <code>-M MAX_DAYS</code>        | <code>--maxdays<br/>MAX_DAYS</code>       | Definir el número máximo de días antes de que se pueda cambiar la contraseña a <code>MAX_DAY</code>                    |
| <code>-W<br/>WARN_DAYS</code>   | <code>--warndays<br/>WARN_DAYS</code>     | Establecer el número de días antes de que caduque una contraseña para mostrar una advertencia a <code>WARN_DAYS</code> |

Un buen ejemplo del comando `chage` sería cambiar el número máximo de días para la validez de la contraseña de una persona a 60 días:

```
root@localhost:~# chage -M 60 jane
```

14.11 Modificar un Usuario

Antes de hacer cambios a una cuenta de usuario, debes entender que algunos comandos no modificarán con éxito una cuenta de usuario si el usuario está actualmente conectado (como por ejemplo cambiando su nombre de usuario del inicio de la sesión).

Otros cambios que podrías hacer no serán efectivos si el usuario está conectado, sino será efectivo tan pronto como el usuario cierre sesión y luego inicie sesión de nuevo. Por ejemplo, si vas a modificar las membresías de los grupos, entonces las nuevas membresías estarán disponibles para el usuario hasta la próxima vez que el usuario inicie sesión.

En cualquier caso, es útil saber cómo utilizar los comandos `who`, `w` y `last`, para que puedas saber quién está conectado en el sistema, ya que esto puede afectar los cambios que quieres hacer a un usuario.

Los comandos `who` y `w` los vimos en el capítulo anterior. Ambos comandos te permiten ver quién está actualmente conectado en el sistema. El comando `w` es el más detallado de los dos, ya que muestra información de tiempo de actividad y carga del sistema, así como qué procesos está ejecutando cada usuario.

El comando `last` es ligeramente diferente de los comandos `who` y `w`. Por defecto, también muestra el nombre de usuario, terminal y ubicación del inicio de la sesión, no sólo de las sesiones actuales iniciadas en el sistema, sino las sesiones anteriores también. A diferencia de los comandos `who` y `w`, mostrará la fecha y hora en la que el usuario inició la sesión. Si el usuario cerró la sesión del sistema, entonces se mostrará el tiempo total de conexión o se mostrará "still logged in" (o «sigue conectado» en español).

El comando `last` lee la historia completa de la sesión desde el archivo `/var/log/wtmp` y muestra todos los inicios de sesión y los reinicios por defecto. Un detalle interesante de los registros de reinicio es que se muestra la versión del kernel Linux que fue arrancado en lugar de la ubicación del inicio de la sesión.

Proporcionando un nombre de usuario o un nombre de `tty` (terminal) como argumento, el comando sólo mostrará los registros que coincidan con ese nombre. Si necesitas averiguar quién se conectó a partir de una determinada fecha y hora, el comando `last` lo puede mostrar, si utilizas la opción `-t` para especificar tal fecha y hora.

14.11.1 El Comando `usermod`

El comando `usermod` ofrece muchas opciones para modificar una cuenta de usuario existente. Observa que la mayoría de estas opciones también están disponible con el comando `useradd` en el momento de crear la cuenta. La siguiente tabla proporciona un resumen de las opciones `usermod`:

| Opción corta             | Opción larga                 | Descripción   |
|--------------------------|------------------------------|---|
| <code>-c</code>          | <code>COMMENT</code>         | Establecer el valor del campo GECOS o comentario a <code>COMMENT</code> . |
| <code>-d HOME_DIR</code> | <code>--home HOME_DIR</code> | Establecer un nuevo directorio home para el usuario.                      |

| Opción corta                                | Opción larga  | Descripción  |
|---|---|--|
| <code>-e</code><br><code>EXPIRE_DATE</code> | <code>--expiredate</code><br><code>EXPIRE_DATE</code> | Configurar la fecha de caducidad de la cuenta a <code>EXPIRE_DATE</code> .                                 |
| <code>-f INACTIVE</code>                    | <code>--inactive</code><br><code>INACTIVE</code>      | Configurar la cuenta para permitir acceso <code>INACTIVE</code> días después de que la contraseña caduque. |
| <code>-g GROUP</code>                       | <code>--gid GROUP</code>                              | Establecer <code>GROUP</code> como grupo primario.   |
| <code>-G GROUPS</code>                      | <code>--groups</code><br><code>GROUPS</code>          | Establecer grupos adicionales a una lista especificada en <code>GROUPS</code> .                            |
| <code>-a</code>                             | <code>--append</code>                                 | Añadir grupos adicionales del usuario especificados por <code>-G</code> .                                  |
| <code>-h</code>                             | <code>--help</code>                                   | Mostrar la ayuda para <code>usermod</code> .   |
| <code>-l</code><br><code>NEW_LOGIN</code>   | <code>--login</code><br><code>NEW_LOGIN</code>        | Cambiar el nombre de inicio de sesión del usuario.   |
| <code>-L</code>                             | <code>--lock</code>                                   | Bloquear la cuenta de usuario.   |
| <code>-s SHELL</code>                       | <code>--shell SHELL</code>                            | Especificar el shell de inicio de sesión para la cuenta.   |

| Opción corta            | Opción larga               | Descripción   |
|-------------------------|----------------------------|---|
| <code>-u NEW_UID</code> | <code>--uid NEW_UID</code> | Especificar que el UID del usuario sea <code>NEW_UID</code> . |
| <code>-U</code>         | <code>--unlock</code>      | Desbloquear la cuenta de usuario.                             |

Varias de estas opciones son importantes debido a la forma en la que afectan la administración de usuarios. Puede ser muy complicado cambiar el UID del usuario con la opción `-u`, ya que los archivos pertenecientes al usuario quedarán huérfanos. Por otra parte, especificando un nuevo nombre de inicio de sesión para el usuario con la opción `-l` no resulta en archivos huérfanos.

Borrar un usuario con el comando `userdel` (véase la sección siguiente) puede resultar en archivos huérfanos o eliminados del usuario del sistema. En vez de eliminar la cuenta, otra opción es bloquear la cuenta con la opción `-l` del comando `usermod`. El bloqueo de una cuenta evita que la cuenta se utilice, pero sigue siendo propietario de los archivos.

Hay algunas cosas importantes que debes saber sobre el manejo de los grupos suplementarios. Si utilizas la opción `-G` sin la opción `-a`, debes listar todos los grupos a los que perteneciera el usuario. Usando solamente la opción `-G` puede accidentalmente quitar un usuario de todos los anteriores grupos suplementarios a los que pertenece.

Si utilizas la opción `-a` sin la opción `-G`, solamente tienes que listar los grupos nuevos a los que perteneciera el usuario. Por ejemplo, si el usuario `jane` pertenece actualmente a los grupos `wheel` y `research`, entonces para agregar su cuenta al grupo de desarrollo, ejecuta el siguiente comando:

```
root@localhost:~# usermod -aG development jane
```

## 14.12 Eliminar a un Usuario

Cuando eliminas una cuenta de usuario, también necesitas decidir si quieres eliminar el directorio home del usuario. Los archivos del usuario pueden ser importantes para tu organización, e incluso, puede que existan requisitos legales para mantener los datos durante un cierto periodo de tiempo, así que ten cuidado de no tomar esta decisión a la ligera. También, a menos que hayas hecho copias de seguridad de los datos, una vez que ejecutes el comando para borrar al usuario y sus archivos, tal acción no se puede revertir.

Para eliminar al usuario `jane` sin eliminar el directorio home del usuario (`/home/jane`) se puede ejecutar:

```
root@localhost:~# userdel jane
```

Ten en cuenta que borrar un usuario sin borrar su directorio home significa que los archivos del directorio home del usuario son ahora huérfanos y estos archivos serán propiedad únicamente de sus previos UID y GID.

Para eliminar al usuario `jane` y borrar el directorio `/home/jane`, utiliza la opción `-r`:

```
root@localhost:~# userdel -r jane
```

**ADVERTENCIA:** El comando anterior solo borra los archivos del usuario en su directorio home y mail spool. Si el usuario posee otros archivos fuera de su directorio, los archivos seguirán existiendo como archivos huérfanos.

## NDG Linux Essentials: Capítulo 15: Propiedad y permisos.

### 15.1 Introducción

La propiedad de archivo es crítica para la seguridad de los archivos. Cada archivo tiene un usuario propietario y un propietario de grupo.

Este capítulo se centrará en cómo especificar el usuario y grupo propietarios de un archivo. Además, se explorará el concepto de los permisos de archivo y directorio, incluyendo cómo cambiar los permisos de los archivos y directorios. Por defecto los permisos son los permisos establecidos para los archivos y directorios cuando se crean inicialmente.

#### Porqué obtener una certificación LPI?

LPI ofrece credenciales que son reconocidas a lo ancho de toda la industria como un medio de validar habilidades a través de todas las distribuciones de Linux; para proporcionarte conocimiento básico y el más amplio rango de oportunidades de trabajo. Linux Essentials es un primer gran paso en acelerar tu carrera tecnológica.

### 15.2 Propiedad de Archivo

De forma predeterminada, los usuarios poseen los archivos que crean. Mientras que esta propiedad puede cambiarse, esta función requiere privilegios administrativos. Aunque la mayoría de los comandos generalmente muestran al usuario propietario como un nombre, el sistema operativo en realidad asociará la propiedad del usuario con el UID para ese nombre de usuario.

Cada archivo también tiene un grupo propietario. En el capítulo anterior sobre la creación de los usuarios y grupos hablamos sobre el grupo primario del usuario. De forma predeterminada, el grupo primario del usuario que crea el archivo será el grupo propietario de los nuevos archivos. Los usuarios pueden cambiar a un propietario de grupo de un archivo a cualquier grupo al que pertenecen. De manera similar al usuario propietario, la asociación de un archivo con un grupo no está realmente hecha por nombre dentro del sistema operativo, sino por el GID del grupo.

Puesto que la propiedad se determina por el UID y GID asociado con un archivo, cambiar el UID de un usuario (o borrar el usuario) tiene el efecto de hacer que el archivo que originalmente fue propiedad del usuario no tenga ningún usuario propietario real. Cuando no hay ningún UID en el archivo `/etc/passwd` que coincide con el UID del propietario del archivo, el UID (el número) se mostrará como el usuario propietario del archivo en lugar del nombre de usuario (que ya no existe). Lo mismo ocurre con los grupos.

El comando `id` puede ser útil para verificar la cuenta del usuario que estás usando y qué grupos están disponibles para ti. Mediante la visualización de la salida de este comando, puedes ver la información de la identidad del usuario expresada como un número y un nombre.

La salida del comando `id` muestra el nombre de la cuenta de usuario y el UID del usuario actual seguido por el GID y el nombre del grupo primario y los GIDs y los nombres de grupo de todas las membresías de grupos:

```
sysadmin@localhost:~$ id
uid=500(sysadmin) gid=500(sysadmin) groups=500(sysadmin),10001(research),10002(development) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

El ejemplo anterior muestra que el usuario tiene un UID de 500 para la cuenta del usuario sysadmin. Muestra que el grupo primario para este usuario tiene un GID de 500 para el grupo sysadmin.

Ya que la cuenta de usuario y la cuenta del grupo primario tienen el mismo nombre y el mismo identificador numérico, esto indica que este usuario está en un grupo privado de usuario (UPG). Además, el usuario pertenece a dos grupos adicionales: el grupo `research` (o «investigación» en español) con un GID de 10001 y el grupo `development` (o «desarrollo» en español) con un GID de 10002.

Cuando se crea un archivo, éste pertenecerá al usuario actual y su grupo primario actual. Si el usuario del ejemplo anterior ejecutara un comando como `touch` para crear un archivo y luego ver los detalles de archivo, la salida sería como la siguiente:

```
sysadmin@localhost:~$ touch /tmp/filetest1
sysadmin@localhost:~$ ls -l /tmp/filetest1
-rw-rw-r--. 1 sysadmin sysadmin 0 Oct 21 10:18 /tmp/filetest1
```

El usuario propietario del archivo es `sysadmin` y el grupo propietario es `sysadmin`.

### 15.3 Los Comandos `newgrp` y `groups`

Si sabes que el archivo que vas crear debe pertenecer a un grupo diferente que tu grupo primario actual, entonces puedes utilizar el comando `newgrp` para cambiar tu grupo primario actual.

Como se mostró anteriormente, el comando `id` lista tu información de identidad, incluyendo tu pertenencia a grupos. Si sólo estás interesado en conocer a qué grupos perteneces, puedes ejecutar el comando `groups`:

```
sysadmin@localhost:~$ groups
sysadmin research development
```

La salida de `groups` puede no ser tan detallada como la salida del comando `id`, pero si todo lo que necesitas saber es a qué grupos puedes cambiar utilizando el comando `newgrp`, entonces el comando `groups` proporciona la información que necesitas. La salida del comando `id` muestra tu grupo primario actual, por lo que es útil verificar que el comando `newgrp` tuvo éxito.

Por ejemplo, si el usuario `sysadmin` quería tener un archivo que fuera propiedad del grupo `research`, pero no era el grupo primario del usuario, el usuario podría utilizar el comando `newgrp` y comprobar el grupo primario correcto con el comando `id` antes de crear el nuevo archivo:

```
sysadmin@localhost:~$ id
uid=502(sysadmin) gid=503(sysadmin) groups=503(sysadmin),10001(research),10002(development) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023

sysadmin@localhost:~$ newgrp research

sysadmin@localhost:~$ id
uid=502(sysadmin) gid=10001(research) groups=503(sysadmin),10001(research),10002(development) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

De la salida de los comandos anteriores, primero puedes ver que el GID del usuario es de 503 para el usuario `sysadmin`, y luego se ejecuta el comando `newgrp research`, y después el GID primario del usuario es de 10001, el grupo `research`. Después de estos comandos, si el usuario quiere crear otro archivo y ver sus detalles, el nuevo archivo pertenecería al grupo `research`:

```
sysadmin@localhost:~$ touch /tmp/filetest2
sysadmin@localhost:~$ ls -l /tmp/filetest2
-rw-r--r--. 1 sysadmin research 0 Oct 21 10:53 /tmp/filetest2
```

El comando `newgrp` abre un shell nuevo; mientras el usuario permanece en ese shell, el grupo primario no cambiará. Para cambiar el grupo principal hacia el original, el usuario podría abandonar el shell nuevo ejecutando el comando `exit`. Por ejemplo:

```
sysadmin@localhost:~$ id
uid=502(sysadmin) gid=10001(research) groups=503(sysadmin),10001(research),10002(development) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
sysadmin@localhost:~$ exit
exit
sysadmin@localhost:~$ id
uid=502(sysadmin) gid=503(sysadmin) groups=503(sysadmin),10001(research),10002(development) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Se requieren privilegios administrativos para cambiar permanentemente el grupo primario del usuario. El usuario root ejecutaría el comando `usermod -g groupname username`.

#### 15.4 Los Comandos `chgrp` y `stat`

Si quieres cambiar el grupo propietario de un archivo existente, puedes utilizar el comando `chgrp`. Como un usuario sin privilegios administrativos, el comando `chgrp` puede utilizarse solamente para cambiar el grupo propietario del archivo a un grupo del que el usuario ya sea miembro. Como usuario root, el comando `chgrp` puede utilizarse para cambiar el grupo propietario de cualquier archivo a cualquier grupo.

Mientras que puedes ver la propiedad de un archivo con la opción `-l` del comando `ls`, el sistema proporciona otro comando que es útil al visualizar los permisos y la propiedad de los archivos: el comando `stat`. El comando `stat` muestra información más detallada acerca de un archivo, incluyendo el grupo propietario tanto por nombre de grupo como por el número GID:

```
sysadmin@localhost:~$ stat /tmp/filetest1
File: `/tmp/filetest1'
Size: 0          Blocks: 0          IO Block: 4096   regular empty file
Device: fd00h/64768d  Inode: 31477       Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 502/sysadmin)   Gid: ( 503/sysadmin)
Access: 2013-10-21 10:18:02.809118163 -0700
Modify: 2013-10-21 10:18:02.809118163 -0700
Change: 2013-10-21 10:18:02.809118163 -0700
```

El comando `stat` también será útil más adelante en este capítulo cuando hablemos de los permisos, ya que proporciona más detalles que el comando `ls -l`.

El siguiente gráfico muestra al usuario `sysadmin` cambiando la propiedad de grupo de un archivo que posee el usuario:

```
sysadmin@localhost:~$ chgrp development /tmp/filetest1
sysadmin@localhost:~$ stat /tmp/filetest1                                File: `/tmp/filetest1'
  Size: 0                      Blocks: 0          IO Block: 4096   regular empty file
Device: fd00h/64768d   Inode: 528677          Links: 1
Access: (0664/-rw-rw-r--)  Uid: (  500/sysadmin)   Gid: (  502/development)
Access: 2013-10-21 10:18:02.809118163 -0500
Modify: 2013-10-21 10:18:02.809118163 -0500
Change: 2013-10-21 10:18:02.809118163 -0500
sysadmin@localhost:~$
```

Si un usuario intenta modificar la propiedad de grupo de un archivo que no posee, recibirá un mensaje de error:

```
sysadmin@localhost:~$ chgrp development /etc/passwd
chgrp: changing group of '/etc/passwd': Operation not permitted
```

A veces quieres no sólo cambiar los archivos en el directorio actual, pero también los archivos en los subdirectorios. Cuando se ejecuta con la opción `-R` (recursivo), el comando `chgrp` operará no sólo en el directorio actual, sino también en todos los directorios que pueden estar anidados bajo el directorio especificado. La operación también afectarán a todos los archivos en los subdirectorios, no sólo en los directorios.

El gráfico siguiente ilustra el uso de la opción `-R`:

```
sysadmin@localhost:~$ cp -r /etc/sound .
sysadmin@localhost:~$ ls -ld sound
drwxr-xr-x 1 sysadmin sysadmin 0 Dec 11 02:02 sound
sysadmin@localhost:~$ ls -lR sound
sound:
total 4
drwxr-xr-x. 2 sysadmin sysadmin 4096 Oct 28 13:06 events

sound/events:
total 48
-rw-r--r--. 1 sysadmin sysadmin 27223 Oct 28 13:06 gnome-2.soundlist
-rw-r--r--. 1 sysadmin sysadmin 18097 Oct 28 13:06 gtk-events-2.soundlist
```



```

sysadmin@localhost:~$ chgrp -R development sound
sysadmin@localhost:~$ ls -ld sound
drwxr-xr-x. 3 sysadmin development 4096 Oct 28 13:06 sound
ls -lR sound
sound:
total 4
drwxr-xr-x. 2 sysadmin development 4096 Oct 28 13:06 events

sound/events:
-rw-r--r--. 1 sysadmin development 27223 Oct 28 13:06 gnome-2.soundlist
-rw-r--r--. 1 sysadmin development 18097 Oct 28 13:06 gtk-events-2.soundlist
sysadmin@localhost:~$

```

### 15.5 Comando chown

El comando `chown` permite al usuario root cambiar el usuario propietario de archivos y directorios. Un usuario normal no puede utilizar este comando para cambiar el usuario propietario de un archivo, ni siquiera para pasar la propiedad de uno de sus propios archivos a otro usuario. Sin embargo, el comando `chown` también permite cambiar la propiedad de grupo, que se puede lograr a través del root o el propietario del archivo.

Existen tres maneras diferentes de ejecutar el comando `chown`. El primer método se utiliza para cambiar sólo al usuario propietario del archivo. Por ejemplo, si el usuario root quiere cambiar la propiedad de usuario del archivo `abc.txt` al usuario `ted`, entonces el siguiente comando puede ser ejecutado:

```
root@localhost:~# chown ted abc.txt
```

El segundo método es cambiar ambos el usuario y el grupo; esto también requiere privilegios de root. Para lograr esto, debes separar al usuario y el grupo por dos puntos o un punto. Por ejemplo:

```

root@localhost:~# chown user:group /path/to/file
root@localhost:~# chown user.group /path/to/file

```

Si un usuario no tiene privilegios de root, puede utilizar el tercer método para cambiar el grupo propietario de un archivo al igual que el comando `chgrp`. Para usar `chown` para cambiar sólo la propiedad de grupo del archivo, usar dos puntos o un punto como un prefijo para el nombre del grupo:

```
root@localhost:~# chown :group /path/to/file
```

```
root@localhost:~# chown .group /path/to/file
```

## 15.6 Permisos

Al ejecutar el comando `ls -l`, la salida resultante muestra diez caracteres al principio de cada línea, que indican el tipo de archivo y los permisos del archivo:

- El primer carácter indica el tipo de archivo.
- Los caracteres 2-4 indican los permisos para el usuario al que pertenece el archivo.
- Los caracteres 5-7 indican los permisos para el grupo al que pertenece el archivo.
- Los caracteres 8-10 indican los permisos para "otros" o lo que se conoce a veces como los permisos del mundo. Esto incluiría todos los usuarios que no sean el propietario del archivo o un miembro del grupo del archivo.

Por ejemplo, considera la salida del siguiente comando:

```
root@localhost:~# ls -l /etc/passwd
-rw-r--r--. 1 root root 4135 May 27 21:08 /etc/passwd
```

Basándose en la salida del comando anterior, los primeros diez caracteres podrían ser descritos por la siguiente tabla:

| File | User owner |       |         | Group owner |       |         | Other or world |       |         |
|------|------------|-------|---------|-------------|-------|---------|----------------|-------|---------|
| type | read       | write | execute | read        | write | execute | read           | write | execute |
| -    | r          | w     | -       | r           | -     | -       | r              | -     | -       |

### Caracter Tipo de Archivo

|   |   |
|---|---|
| - | Un archivo normal que puede estar vacío, contener texto o datos binarios.                   |
| d | Un archivo de directorio que contiene los nombres de otros archivos y enlaces a los mismos. |
| l | Un enlace simbólico es un nombre de archivo que hace referencia (apunta) a otro archivo.    |

| Caracter | Tipo de Archivo  |
|----------|--|
| b        | Un archivo de bloque es el que se refiere a un dispositivo de hardware de bloque donde los datos se leen en bloques de datos.  |
| c        | Un archivo de caracteres es aquel que se refiere a un dispositivo de hardware de caracteres, donde se leen los datos un byte a la vez.   |
| p        | Una archivo «pipe» («barra vertical» en español) funciona de forma similar al símbolo de barra vertical («pipe» en inglés), lo que permite a la salida de un proceso comunicarse con otro proceso por el archivo «pipe», donde se utiliza la salida de un proceso como entrada para el otro proceso. |
| s        | Un archivo de socket permite que se comuniquen dos procesos, donde se permite a ambos procesos enviar o recibir datos.   |

A pesar de que todos los tipos de archivos estan listados en la tabla anterior, lo más probable es que nunca te encontrarás nada más que archivos regulares, directorios y archivos de enlace, a menos que explores el directorio `/dev`.

Los caracteres de la parte de permisos de la salida tienen los siguientes significados:

- **r** significa el permiso de **leer**
- **w** significa el permiso de **escribir**
- **x** significa el permiso de **ejecutar**

Los permisos establecidos en estos archivos determinan el nivel de acceso que un usuario va a tener en el archivo. Cuando un usuario ejecuta un programa y el programa accede a un archivo, los permisos se comprueban para determinar si el usuario tiene los derechos de acceso correctos en el archivo.

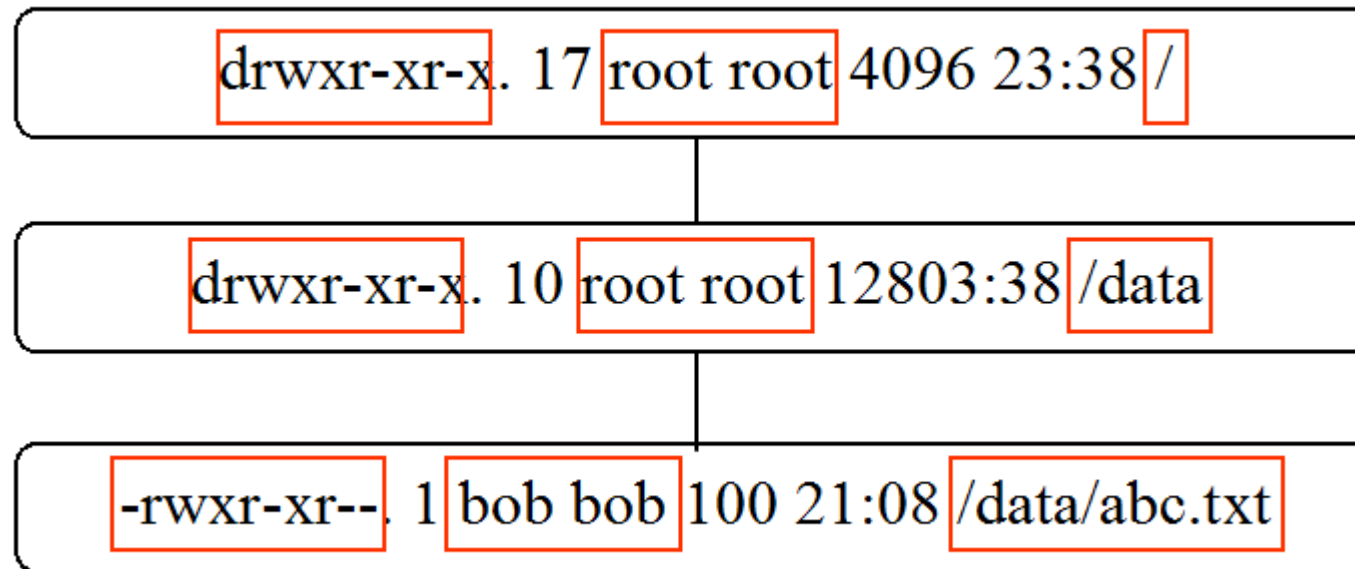
Los propios permisos son engañosamente simple y tienen un significado diferente dependiendo de si se aplican a un archivo o un directorio:

| Permiso        | Significado relacionado a un archivo   | Significado relacionado a un directorio   |
|----------------|--|---|
| <code>r</code> | El proceso puede leer el contenido del archivo, es decir, los contenidos se pueden ver y copiar.   | Los nombres de archivo en el directorio se pueden enumerar, pero otros detalles no están disponibles.   |
| <code>w</code> | El proceso puede escribir en este archivo, por lo que los cambios se pueden guardar. Ten en cuenta que el permiso <code>w</code> realmente requiere el permiso <code>r</code> en un archivo para que funcione correctamente. | Los archivos se pueden agregar a un directorio o quitar del mismo. Ten en cuenta que el permiso <code>w</code> requiere el permiso <code>x</code> en el directorio para que funcione correctamente.                     |
| <code>x</code> | El archivo se puede ejecutar o correr como un proceso.   | El usuario puede utilizar el comando <code>cd</code> para «entrar» al directorio y utilizar el directorio en una ruta de acceso para acceder a los archivos y, potencialmente, a los subdirectorios de este directorio. |

### 15.6.1 Comprendiendo los Permisos

Mientras que la tabla de la página anterior puede ser una referencia útil, por sí sola no ofrece una descripción clara de cómo funcionan los permisos. Para entender mejor cómo funcionan los permisos, considera los siguientes escenarios.

Para entender estos escenarios, primero debes entender el siguiente diagrama:



La información importante viene resaltada. El primer cuadro representa el directorio `/` con un usuario propietario `root`, un grupo propietario `root` y los permisos `drwxr-xr-x`. El segundo cuadro representa el directorio `/data`, un directorio que está bajo el directorio `/`. El tercer cuadro representa el archivo `abc.txt`, que se almacena en el directorio `/data`.

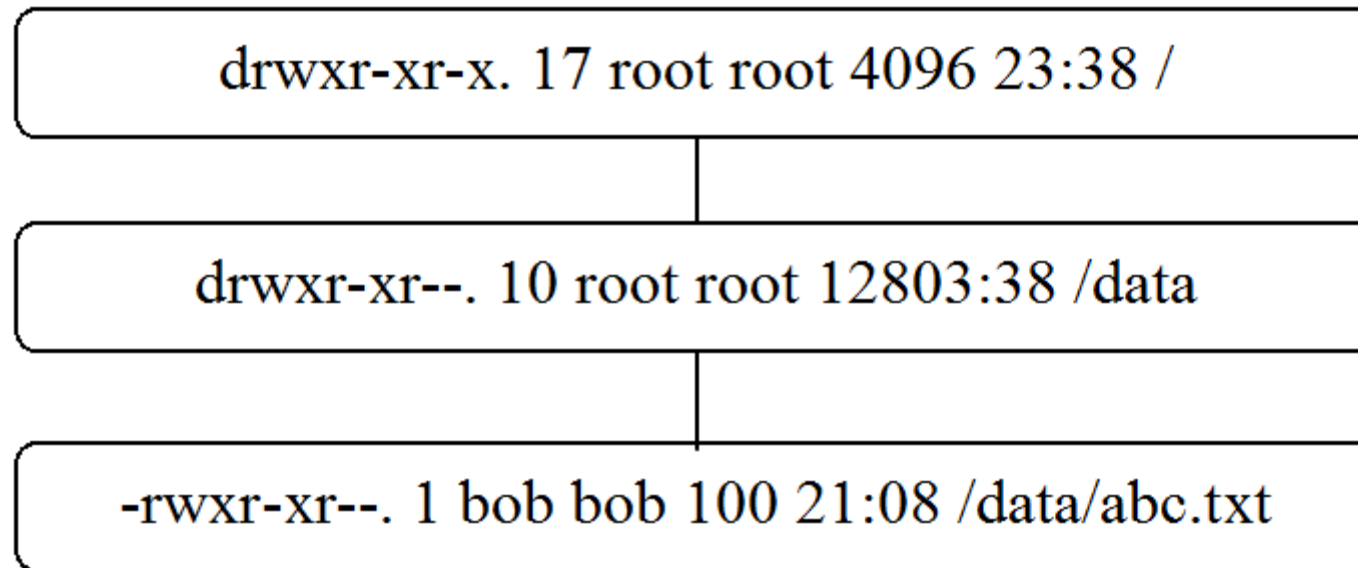
También debes entender que si eres el propietario del archivo/directorio, entonces sólo los permisos de propiedad de usuario se utilizarán para determinar el acceso a ese archivo/directorio.

Si no eres el propietario, pero eres un miembro del grupo que posee el archivo/directorio, entonces sólo los permisos de propiedad de grupo se utilizarán para determinar el acceso a ese archivo o directorio.

Si no eres el propietario y tampoco eres un miembro del grupo de archivos o directorios, entonces los permisos serían «otros».

#### 15.6.1.1 Escenario #1 - La importancia del Acceso al Directorio

**Pregunta:** Según el siguiente diagrama, ¿Qué acceso tendría el usuario `bob` en el archivo `abc.txt`?



**Respuesta:** Ninguno

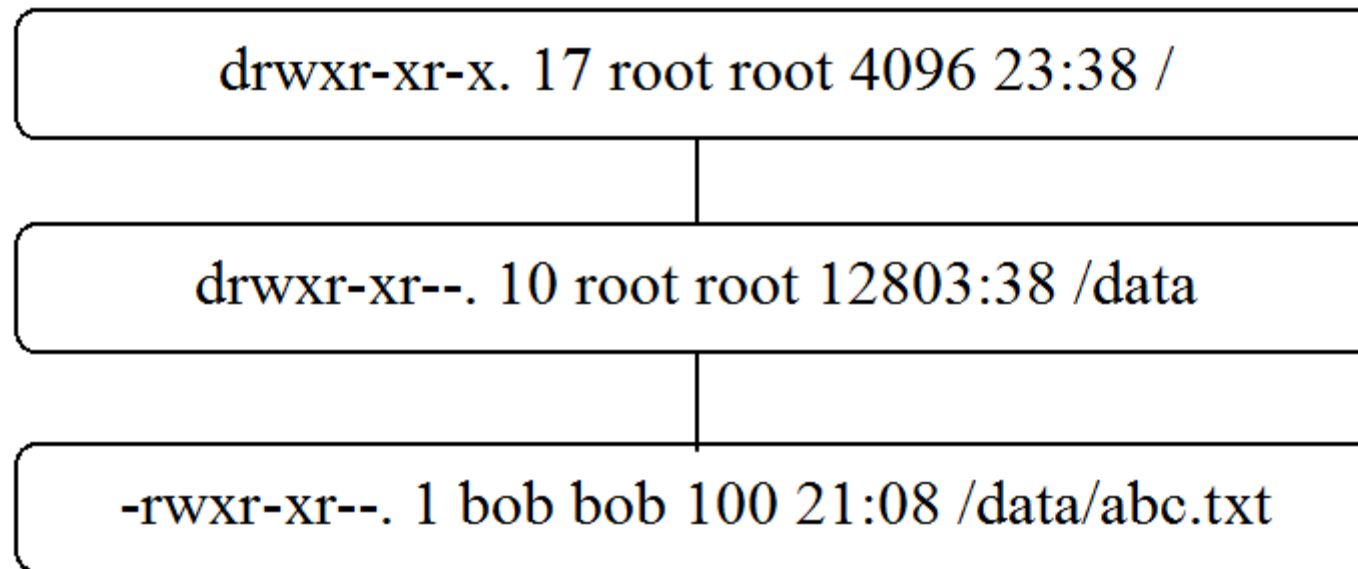
**Explicación:** Al principio podría parecer que el usuario bob puede ver el contenido del archivo abc.txt, igual que copiar el archivo, modificar su contenido y ejecutarlo como un programa. Esta conclusión errónea sería el resultado de tener en cuenta únicamente permisos del archivo (`rwx` para el usuario bob en este caso).

Sin embargo, para hacer algo con el archivo, el usuario debe primero «entrar» al directorio `/data`. Los permisos para bob en el directorio `/data` son los permisos para «otros» (`r--`), que significa bob no puede utilizar ni siquiera el comando `cd` para entrar al directorio. Si el permiso de ejecutar (`--x`) fuera configurado para el directorio, entonces el usuario bob sería capaz de «entrar» al directorio, lo que significa que se aplicarían los permisos del propio archivo.

**Lección aprendida:** Los permisos de todos los directorios padres deben considerarse antes de considerar los permisos en un archivo específico.

#### 15.6.1.2 Escenario #2 - Visualizar el Contenido del Directorio

Pregunta: Según el diagrama siguiente, ¿Quién puede utilizar el comando `ls` para mostrar el contenido del directorio `/data` (`ls /data`)?



**Respuesta:** Todos los usuarios

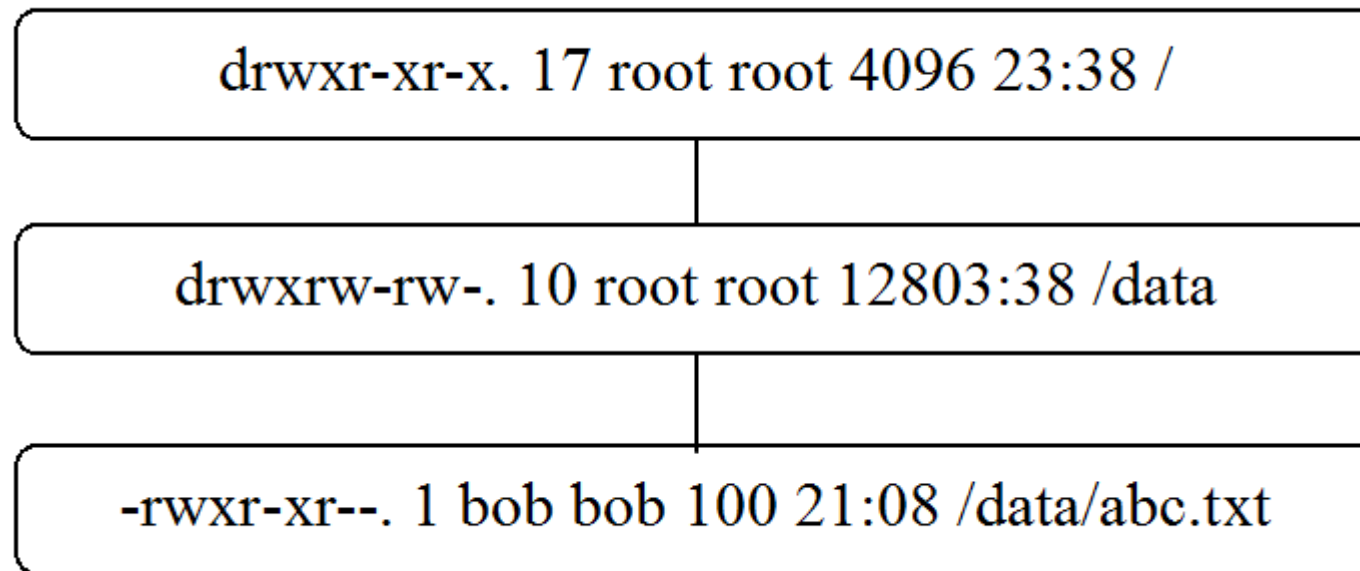
**Explicación:** Todo lo que es necesario para poder ver el contenido de un directorio es el permiso `r` en el directorio (y poder acceder a los directorios padres). El permiso `x` para todos los usuarios en el directorio `/` significa que todos los usuarios pueden usar `/` como parte de la ruta, así que todo el mundo puede pasar a través del directorio `/` al directorio `/data`. El permiso de `r` para todos los usuarios en el directorio `/data` significa que todos los usuarios pueden utilizar el comando `ls` para ver el contenido. Esto incluye los archivos ocultos, entonces el comando `ls -a` también funciona en este directorio.

Sin embargo, ten en cuenta que para ver los datos de los archivos (`ls -l`) requiere también el permiso `x` en el directorio. Así que mientras el usuario `root` y los miembros del grupo `root` tienen este acceso en el directorio `/data`, otros usuarios no pueden ejecutar `ls -l /data`.

**Lección aprendida:** El permiso `r` permite a un usuario ver un listado del directorio.

#### 15.6.1.3 Escenario #3 - Eliminar el Contenido del Directorio

**Pregunta:** Según el diagrama siguiente, ¿Quién puede eliminar el archivo `/data/abc.txt`?



**Respuesta:** Sólo el usuario root

**Explicación:** Un usuario no necesita ninguno permiso en el archivo para eliminarlo. El permiso **w** en el directorio en el cual se almacena el archivo se necesita para eliminar un archivo de un directorio. Con base en esto, parece que todos los usuarios pueden eliminar el archivo `/data/abc.txt`, ya que todo el mundo tiene el permiso **w** en el directorio.

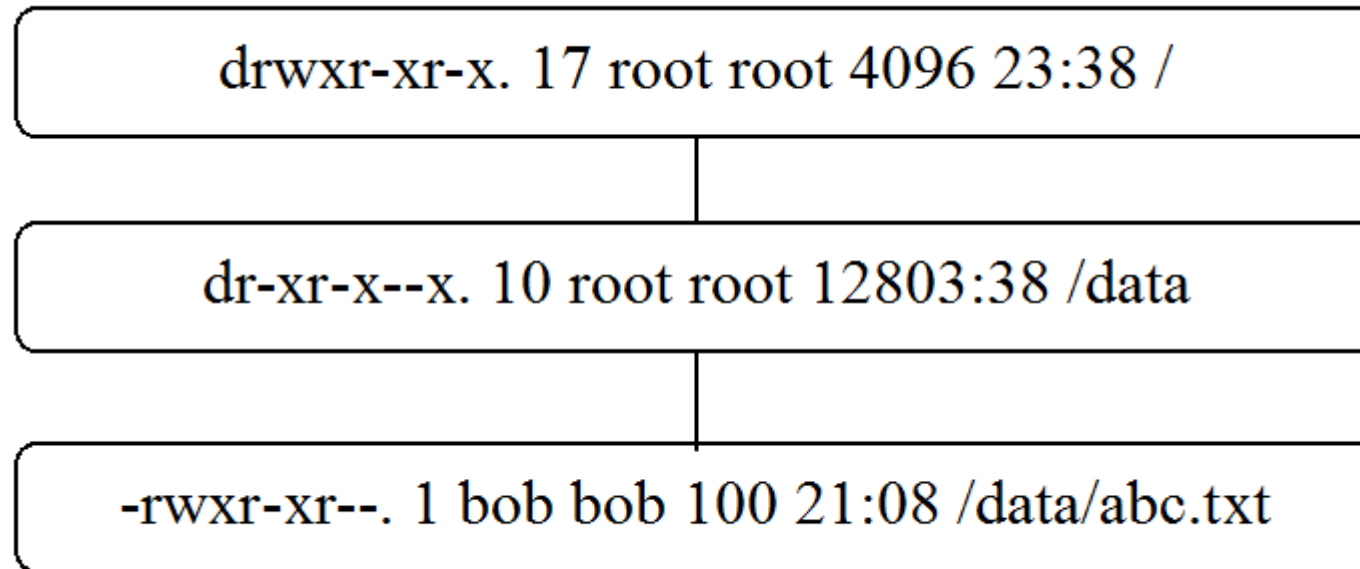
Sin embargo, para eliminar un archivo, también tienes que poder «entrar» al directorio. Puesto que sólo el usuario root tiene el permiso **x** en el directorio `/data`, sólo el root puede «entrar» a ese directorio con el fin de eliminar los archivos de este directorio.

**Lección aprendida:** El permiso **w** permite eliminar los archivos de un directorio, pero sólo si el usuario también tiene el permiso **x** en el directorio.

#### 15.6.1.4 Escenario #4 - Acceso a los Contenidos de un Directorio

**Pregunta:** Verdadero o falso: Según el siguiente diagrama, ¿Puede el usuario `bob` ejecutar con éxito el siguiente comando: `more /data/abc.txt`?





**Respuesta:** Verdadero

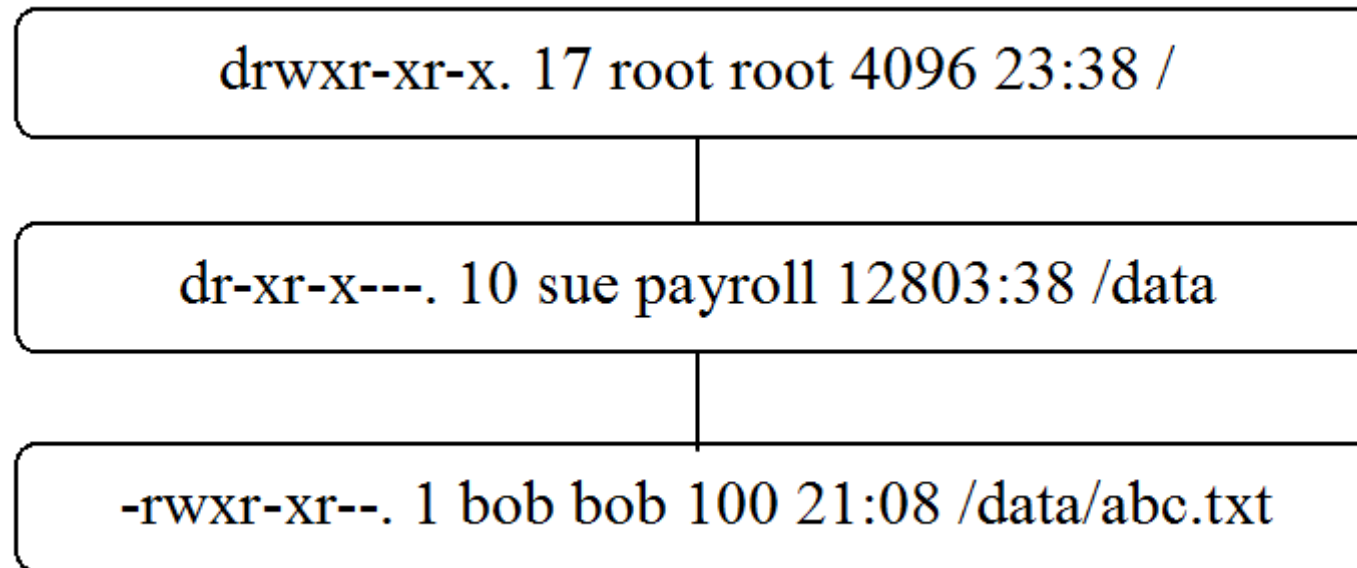
**Explicación:** Como se mencionó anteriormente, para acceder a un archivo, el usuario debe tener acceso al directorio. El acceso al directorio sólo requiere el permiso `x`; aunque el permiso de `r` sería útil para listar los archivos en un directorio, no se necesita para «entrar» en el directorio y acceder a los archivos dentro del directorio.

Cuando se ejecuta el comando `more /data/abc.txt`, se comprueban los siguientes permisos: el permiso `x` en el directorio `/`, el permiso `x` en el directorio `data` y el permiso `r` en el archivo `abc.txt`. Puesto que el usuario `bob` tiene todos estos permisos, el comando se ejecuta con éxito.

**Lección aprendida:** El permiso `x` se necesita para «entrar» a un directorio, pero no se necesita el permiso `r` en el directorio, a menos, que quieras listar el contenido del directorio.

#### 15.6.1.5 Escenario #5 - La Complejidad de los Usuarios y Grupos

**Pregunta:** Verdadero o falso: Según el siguiente diagrama, ¿Puede el usuario `bob` ejecutar con éxito el siguiente comando: `more /data/abc.txt` (observa que el directorio `/data` tiene un usuario y grupo propietarios diferentes a los ejemplos anteriores)?



**Respuesta:** No hay suficiente información para determinarlo.

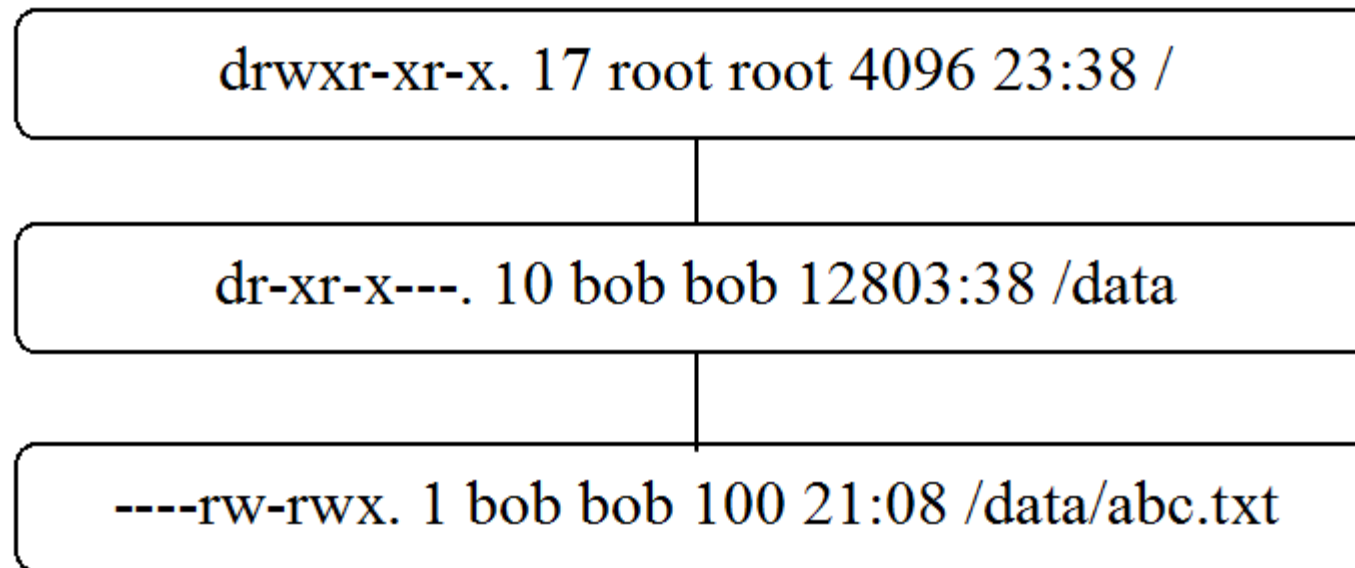
**Explicación:** Para acceder al archivo `/data/abc.txt`, el usuario `bob` necesita permiso para «entrar» en el directorio `/data`. Esto requiere el permiso `x`, que `bob` puede tener o no, dependiendo de si es un miembro del grupo `payroll`.

Si `bob` es un miembro del grupo `payroll`, entonces sus permisos en el directorio `/data` son `r-x` y el comando `more /data/abc.txt` se ejecutará con éxito (`bob` también necesita `x` en `/` y `r` en `abc.txt`, que ya tiene). Si no es un miembro del grupo `payroll`, sus permisos en el directorio `/data` son `---` y el comando fallaría.

**Lección aprendida:** Debes tener en cuenta los permisos de cada archivo y directorio por separado y ser consciente a qué grupos pertenece la cuenta de usuario.

#### 15.6.1.6 Escenario #6 - Prioridad de Permiso

**Pregunta:** Verdadero o falso: Según el siguiente diagrama, ¿Puede el usuario `bob` ejecutar con éxito el siguiente comando: `more /data/abc.txt` (observa que el directorio `/data` tiene un usuario y grupo propietarios diferentes a los ejemplos anteriores)?



**Respuesta:** Falso

**Explicación:** Recuerda que si eres el propietario de un archivo, entonces sólo los permisos que son validados son los del usuario propietario. En esta caso, sería -- para bob en el archivo `/data/abc.txt`.

En este caso, los miembros del grupo de bob y «otros» tienen más permisos sobre el archivo que los que tiene bob.

**Lección aprendida:** No proporcionar permisos para el grupo propietario y «otros» sin aplicar al menos el mismo nivel de acceso para el propietario del archivo.

#### 15.6.2 Usando el Comando `chmod` - Método Simbólico

El comando `chmod` (change mode) se utiliza para cambiar los permisos de un archivo o directorio. Hay dos técnicas a la hora de utilizar este comando: simbólico y numérico. Ambas técnicas utilizan la siguiente sintaxis básica:

```
chmod new_permission file_name
```

o

```
chmod nuevo_permiso nombre_de_archivo
```

**Importante:** Para cambiar los permisos de un archivo, necesitas ser el propietario del archivo o iniciar la sesión como el usuario root.

Si quieres modificar algunos de los permisos actuales, el método simbólico probablemente será más fácil de usar. Con este método especificas que permisos quieres cambiar en el archivo y los permisos de otros permanecen siendo como son.

Cuando se especifica el `nuevo_permiso`, comienzas por utilizar uno de los caracteres siguientes para indicar qué conjunto de permisos quieres cambiar:

- `u` = cambiar los permisos del usuario propietario

- **g** = cambiar los permisos del grupo propietario
- **o** = cambiar los permisos de «otros»

**a** = aplicar los cambios a todos los conjuntos de permisos (usuario propietario, grupo propietario y «otros»)

Debe especificar un **+** para agregar un permiso o un **-** para quitar un permiso. Por último, especifica **r** para la **lectura** **w** para **escritura** y **x** para **ejecución**.

Por ejemplo, para conceder permiso de lectura al usuario propietario en un archivo denominado `abc.txt`, podrías utilizar el siguiente comando:

```
root@localhost:~# chmod u+r abc.txt
```

Solamente se cambió el permiso del usuario propietario. Todos los otros permisos permanecieron como estaban antes de la ejecución del comando `chmod`.

Puedes combinar los valores para realizar múltiples cambios en los permisos del archivo. Por ejemplo, considera el siguiente comando que agregará permisos de lectura para el usuario propietario y grupo propietario mientras quita el permiso de escritura para «otros»:

```
root@localhost:~# chmod ug+r,o-w abc.txt
```

Por último, podrías utilizar el carácter **=** en lugar de **-** o **+** para especificar exactamente los permisos que quieres para un conjunto de permisos:

```
root@localhost:~# chmod u=r-x abc.txt
```

### 15.6.3 Usando el Comando `chmod` - Método Numérico

El método numérico (también llamado el método octal) es útil cuando quieres cambiar muchos permisos en un archivo. Se basa en el sistema octal de numeración en el que a cada tipo de permiso se le asigna un valor numérico:

---

|   |                    |
|---|--------------------|
| 4 | <b>read</b> (leer) |
|---|--------------------|

---

|   |                         |
|---|-------------------------|
| 2 | <b>write</b> (escribir) |
|---|-------------------------|

---

|   |                           |
|---|---------------------------|
| 1 | <b>execute</b> (ejecutar) |
|---|---------------------------|

---

Usando una combinación de números del 0 al 7, cualquier combinación de permisos posible para leer, escribir y ejecutar se pueden especificar por un conjunto de permisos individuales. Por ejemplo:

---

|   |                  |
|---|------------------|
| 7 | <code>rwx</code> |
|---|------------------|

---

|   |                  |
|---|------------------|
| 6 | <code>rw-</code> |
|---|------------------|

---

|   |     |
|---|-----|
| 5 | r-x |
| 4 | r-- |
| 3 | -wx |
| 2 | -w- |
| 1 | --x |
| 0 | --- |

Cuando se utiliza el método numérico para cambiar los permisos, se deben especificar los nueve permisos. Debido a esto, el método simbólico es generalmente más fácil para cambiar unos pocos permisos, mientras que el método numérico es mejor para los cambios más drásticos.

Por ejemplo, para establecer los permisos de un archivo llamado `abc.txt` a `rwxr-xr--` puedes utilizar el siguiente comando:

```
root@localhost:~]# chmod 754 abc.txt
```

## 15.7 Revisión del Comando stat

Recordemos el comando `stat` mencionado anteriormente en este capítulo. Este comando proporciona información más detallada que el comando `ls -l`.

Debido a esto, es posible considerar el uso del comando `stat` en lugar del comando `ls -l` durante la visualización de los permisos de un archivo. Una gran ventaja del comando `stat` es que muestra permisos tanto simbólicamente como por método numérico, como se demuestra a continuación:

```
sysadmin@localhost:~$ stat /tmp/filetest1

File: `/tmp/filetest1'
  Size: 0          Blocks: 0          IO Block: 4096   regular empty file
Device: fd00h/64768d  Inode: 31477       Links: 1
Access: (0664/-rw-rw-r--)  Uid: (  502/sysadmin)  Gid: (  503/sysadmin)
Access: 2013-10-21 10:18:02.809118163 -0700
Modify: 2013-10-21 10:18:02.809118163 -0700
Change: 2013-10-21 10:18:02.809118163 -0700
```

## 15.8 umask

El comando `umask` es una característica que se utiliza para determinar los permisos predeterminados establecidos al crear un archivo o directorio. Los permisos predeterminados se determinan cuando el *valor de umask* se resta de los permisos máximos predeterminados permisibles. Los permisos máximos por defecto son diferentes para los archivos y para los directorios:

|             |                        |
|-------------|------------------------|
| archivo     | <code>rw-rw-rw-</code> |
| directorios | <code>rw-rw-rw-</code> |

Los permisos que se establecen inicialmente en un archivo cuando se crea no pueden exceder `rw-rw-rw-`. Para tener el permiso de ejecución para un archivo, primero tienes que crear el archivo y luego cambiar los permisos.

El comando `umask` se puede utilizar para mostrar el valor umask actual:

```
sysadmin@localhost:~$ umask
0002
```

Desglose de la salida:

- El primer 0 indica que umask se da como un número octal.
- El segundo 0 indica qué permisos hay que restar de los permisos por defecto de usuario propietario.
- El tercer 0 indica qué permisos hay que restar de los permisos por defecto del grupo propietario.
- El último número 2 indica qué permisos hay que restar de los permisos por defecto de otros.

Ten en cuenta que los diferentes usuarios pueden tener diferentes umasks. Normalmente, el usuario root tendrá una umask más restrictiva que las cuentas de usuario:

```
root@localhost:~# umask
0022
```

### 15.8.1 Cómo Funciona umask

Para entender cómo funciona umask, supongamos que umask se establece en 027 y consideremos lo siguiente:

File Default

667

(valor predeterminado)

|           |      |
|-----------|------|
| Umask     | -027 |
| Resultado | 640  |

La umask 027 significa que, por defecto los archivos nuevos recibirían los permisos 640 o `rw-r-----` tal como se demuestra a continuación:

```
sysadmin@localhost:~$ umask 027
sysadmin@localhost:~$ touch sample
sysadmin@localhost:~$ ls -l sample
-rw-r-----. 1 sysadmin sysadmin 0 Oct 28 20:14 sample
```

Debido a que los permisos predeterminados para los directorios son diferentes que para los archivos, una umask 027 daría lugar a diferentes permisos iniciales sobre los nuevos directorios:

|                        |      |
|------------------------|------|
| Directory Default      | 777  |
| (valor predeterminado) |      |
| Umask                  | -027 |
| Resultado              | 750  |

La umask 027 significa que, por defecto los directorios nuevos recibirían los permisos 750 o `rwxr-x----` tal como se demuestra a continuación:

```
sysadmin@localhost:~$ umask 027
sysadmin@localhost:~$ mkdir test-dir
sysadmin@localhost:~$ ls -ld test-dir
drwxr-x---. 1 sysadmin sysadmin 4096 Oct 28 20:25 test-dir
```

La nueva umask sólo se aplicará a un archivo y los directorios creados durante esa sesión. Cuando arranque un nuevo shell, la umask por defecto será efectiva de nuevo.

Cambiar permanentemente la umask requiere la modificación del archivo `.bashrc` que se encuentra en el directorio home del usuario.

**NDG Linux Essentials: Capítulo 16: Permisos especiales, vínculos y ubicaciones de archivos.**

**16.1 Introducción**

En el capítulo anterior se detallaron los permisos básicos de Linux: leer, escribir y ejecutar. En la mayoría de las circunstancias, estos permisos básicos de Linux serán suficientes para dar cabida a las necesidades de seguridad de los usuarios u organizaciones individuales.

Sin embargo, cuando varios usuarios necesitan trabajar de forma rutinaria en los mismos directorios y archivos, estos permisos pueden no ser suficientes. Los permisos especiales `setuid`, `setgid` y `sticky bit` están diseñados para hacer frente a estas preocupaciones. En este capítulo vas a ver cómo estos permisos especiales funcionan y cómo configurarlos.

Conocer la estructura básica del sistema de archivos puede ser esencial para poder trabajar con éxito con Linux. El Estándar de Jerarquía del Sistema de Archivos (FHS) («Filesystem Hierarchy Standard» en inglés) proporciona una guía para la mayoría de las distribuciones de Linux para saber qué directorios hay y qué poner en ellos. Este capítulo trata de este estándar, así como del propósito de estos directorios comunes.

**Es bueno ser buscado.** 75% de los profesionales de Linux han recibido una llamada de un reclutador en los últimos 6 Meses. Casi la mitad recibieron más de 6 llamadas.

## 16.2 El Permiso `setuid`

Cuando el permiso `setuid` se encuentra en un archivo binario ejecutable (Alias, un programa), el archivo binario se «ejecuta como» el propietario del archivo, no como el usuario que lo ejecuta. Este permiso se establece en una cierta cantidad de utilidades del sistema para que puedan ser manejados por los usuarios normales, pero ejecutados con los permisos `root`, proporcionando acceso a los archivos del sistema a los que el usuario normal normalmente no tiene acceso.

Considera el siguiente escenario en el que el usuario `sysadmin` intenta ver el contenido del archivo `/etc/shadow`:

```
sysadmin@localhost:~$ more /etc/shadow
/etc/shadow: Permission denied
sysadmin@localhost:~$ ls -l /etc/shadow
-rw-r-----. 1 root root 5195 Oct 21 19:57 /etc/shadow
```

Como puedes ver, el archivo `/etc/shadow` no se puede ver (o modificar) por los usuarios normales debido a que los permisos del archivo `/etc/shadow` son: `-rw-r-----`. Dado que el archivo es propiedad del usuario `root`, el administrador del sistema podría modificar temporalmente los permisos en el caso de que quisieran ver o modificar este archivo.

Consideremos ahora el comando `passwd`. Cuando se ejecuta este comando, se modifica el archivo `/etc/shadow`. Esto parece imposible porque otros comandos que el usuario `sysadmin` ejecuta e intentan acceder a este archivo, fallan. Así que, ¿Por qué el usuario `sysadmin` puede modificar el archivo `/etc/shadow` mientras se ejecuta el comando `passwd` cuando normalmente este usuario no tiene acceso al archivo?

El comando `passwd` tiene el permiso especial **`setuid`**. Cuando se ejecuta el comando `passwd` y éste accede al archivo `/etc/shadow`, el sistema actúa como si el usuario que accede al archivo fuera el propietario del comando `passwd` (el usuario `root`), no el usuario que está ejecutando realmente el comando.

Puedes ver este conjunto de permisos mediante la ejecución del comando `ls -l`:

```
sysadmin@localhost:~$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 31768 Jan 28 2010 /usr/bin/passwd
```

Observa el resultado del comando `ls` anterior; el permiso `setuid` está representado por `s` en los permisos de propietario, donde normalmente se esperaría el permiso de ejecución.



Al igual que los permisos de lectura, escritura y ejecución, permisos especiales pueden ser ajustados con el comando `chmod`, utilizando cualquiera de los métodos, simbólicos y octales.

Para agregar el permiso setuid simbólicamente, ejecuta:

```
chmod u+s file
```

Para agregar el permiso setuid numéricamente, agrega 4000 a los permisos existentes del archivo (asume que el archivo tenía originalmente 775 para su permiso en el ejemplo siguiente):

```
chmod 4775 file
```

Para retirar el permiso setuid simbólicamente, ejecuta:

```
chmod u-s file
```

Para retirar el permiso setuid numéricamente, resta 4000 de los permisos existentes del archivo:

```
chmod 0775 file
```

En el capítulo anterior, establecimos permisos con el método octal utilizando códigos de tres dígitos. Cuando se proporciona un código de tres dígitos, el comando `chmod` supone que el primer dígito antes del código de tres dígitos es 0. Los permisos especiales serán establecidos sólo si se fijan los cuatro dígitos.

Si se especifican tres dígitos cuando se cambian los permisos en un archivo que ya tiene un conjunto de permisos especiales, el primer dígito se establecerá en 0 y el permiso especial se quita del archivo.

### 16.3 El Permiso setgid en un Archivo

El permiso setgid es similar a setuid, pero hace uso de los permisos del grupo propietario. En realidad, hay dos formas de permisos setgid: setgid en un archivo y setgid en un directorio. Cómo funciona el setgid depende de si se establece en un archivo o un directorio.

El permiso setgid en un archivo es muy similar a setuid; permite que un usuario ejecute un archivo binario ejecutable proporcionando un acceso adicional (temporal) de grupo. El sistema permitirá que el usuario que ejecuta el comando pertenezca al grupo al que pertenece el archivo, pero sólo «en» el programa setgid. A medida que el comando se ejecuta y accede a archivos, esta afiliación adicional al grupo puede proporcionar acceso a los archivos.

Un buen ejemplo del permiso setgid en un archivo ejecutable es el comando `/usr/bin/wall`. Observa los permisos de este archivo y el grupo propietario:

```
sysadmin@localhost:~$ ls -l /usr/bin/wall
-rwxr-sr-x. 1 root tty 10996 Jul 19 2011 /usr/bin/wall
```

Se puede ver que este archivo es setgid por la presencia de la `s` en la posición de ejecución del grupo. Debido a que este ejecutable pertenece al grupo `TTY`, cuando un usuario ejecuta este comando, el comando podrá acceder a los archivos que son propiedad del grupo `TTY`.

Este acceso es importante porque el comando `/usr/bin/wall` envía mensajes a «terminales». Esto se logra al escribir datos en archivos como los siguientes:

```
sysadmin@localhost:~$ ls -l /dev/tty?
```

```
crw-----. 1 root tty  4, 0 Mar 29  2013 /dev/tty0
crw--w----. 1 root tty  4, 1 Oct 21 19:57 /dev/tty1
```

Ten en cuenta que el grupo `tty` tiene permiso de escritura en los archivos anteriores, mientras que los usuarios que no están en el grupo `tty` («otros») no tienen permisos en estos archivos. Sin el permiso `setgid`, el comando `/usr/bin/wall` fallaría.

#### 16.4 El Permiso `setgid` en un Directorio

El permiso `setgid` también se puede establecer en un directorio. Cuando se establece en un directorio, `setgid` hace que los archivos creados en el directorio automáticamente sean propiedad del grupo que posee el directorio. Esto es contrario a cómo funcionaría normalmente la propiedad de grupo de un archivo nuevo, ya que por defecto los archivos nuevos son propiedad del grupo primario del usuario que creó el archivo.

Además, los directorios creados dentro de un directorio que tiene «establecido» el permiso `setgid` no sólo serán propiedad del grupo que posee el directorio `setgid`, sino además el nuevo directorio de forma automática tendrá el permiso `setgid` también. En otras palabras, si un directorio es `setgid`, entonces, los directorios creados dentro de ese directorio heredarán el permiso `setgid`.

A continuación se muestra la función del permiso `setgid` en un directorio. Observa que el usuario `sysadmin` crea un archivo en un directorio `setgid` `/tmp/data` y que el grupo al que pertenece el archivo no es el grupo primario `sysadmin`, sino más bien el grupo que posee el directorio `demo`:

```
sysadmin@localhost:~$ ls -ld /tmp/data
drwxrwsrwx. 2 root demo 4096 Oct 30 23:20 /tmp/data
sysadmin@localhost:~$ id
uid=500 (sysadmin) gid=500 (sysadmin)
groups=500 (sysadmin),10001 (research),10002 (development)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
sysadmin@localhost:~$ touch /tmp/data/file.txt
sysadmin@localhost:~$ ls -ld /tmp/data/file.txt
-rw-rw-r--. 1 bob demo 0 Oct 30 23:21 /tmp/data/file.txt
```

Entonces, ¿Por qué el administrador configuraría un directorio `setgid`? En primer lugar, ten en cuenta las siguientes cuentas de usuario:

- bob es miembro de payroll group
- sue es miembro de staff group
- tim es miembro de acct group

Estos tres usuarios necesitan trabajar en un proyecto conjunto. Se acercan al administrador para solicitar un directorio compartido en el que puedan trabajar juntos, pero que nadie más pueda acceder a sus archivos. El administrador hace lo siguiente:

- Crea un nuevo grupo llamado `team`.
- Añade bob, sue y tim al grupo `team`.
- Hace un nuevo directorio llamado `/home/team`.
- Hace que el grupo propietario del directorio `/home/team` sea el grupo `team`.
- Otorga al directorio `/home/team` los siguientes permisos: `rw-rwx---`

Como resultado, bob, sue y tim pueden acceder al directorio /home/team y agregar archivos. Sin embargo, hay un problema potencial: cuando bob crea un archivo en el directorio /home/team, los detalles del nuevo archivo se ven así:

```
-rw-r-----. 1 bob payroll 100 Oct 30 23:21 /home/team/file.txt
```

Desafortunadamente, mientras sue y tim pueden acceder al directorio /home/team, no pueden hacer nada con el archivo de bob. Sus permisos para ese archivo son los permisos de «otros» (---).

Si el administrador estableciera el permiso setgid al directorio /home/team, entonces, cuando bob crea el archivo, se vería de la siguiente manera:

```
-rw-r-----. 1 bob team 100 Oct 30 23:21 /home/team/file.txt
```

Y como resultado, sue y tim tendrían acceso de grupo al archivo (r--).

Ciertamente, bob podría cambiar la propiedad de grupo después de crear el archivo (o cambiar los permisos de «otros»), pero esto sería tedioso si se crearan muchos archivos nuevos. El permiso setgid hace que sea más fácil esta situación.

### 16.5 Configurar el Permiso setgid

Utiliza la siguiente sintaxis para agregar el permiso setgid simbólicamente:

```
chmod g+s <archivo|directorio>
```

Para agregar el permiso setgid numéricamente, añade 2000 a los permisos existentes del archivo (asume en el ejemplo siguiente de que el directorio tenía originalmente 775 para su permiso):

```
chmod 2775 <archivo|directorio>
```

Para retirar el permiso setgid simbólicamente, ejecuta:

```
chmod g-s <archivo|directorio>
```

Para retirar el permiso setgid numéricamente, resta 2000 de los permisos existentes del archivo:

```
chmod 0775 <archivo|directorio>
```

En un listado largo el permiso setgid está representado por una s en la posición de ejecución del grupo: drwxrwsr-x.

Una s minúscula significa tanto los permisos de ejecución de setgid como los de grupo están establecidos. Una S mayúscula significa que sólo el permiso setgid está establecido y no el permiso de ejecución del grupo: drwxrwSr-x.

Si ves una S mayúscula en la posición de ejecución del grupo, esto indica que, aunque se haya establecido el permiso setgid, en realidad no está realmente en efecto debido a que el grupo carece de los permisos de ejecución para utilizarlo.

### 16.6 El Permiso Sticky Bit

El permiso sticky bit se utiliza para evitar que otros usuarios eliminen los archivos de los que no son dueños en un directorio compartido. ¡Recordemos que cualquier usuario con permiso de escritura en un directorio puede crear archivos en ese directorio, así como eliminar cualquier archivo del directorio, incluso si no es el propietario del archivo!

El permiso sticky bit permite que los archivos se puedan compartir con otros usuarios, cambiando el permiso de escritura en el directorio para que los usuarios aún puedan añadir y eliminar los archivos del directorio, pero los archivos sólo pueden ser borrados por el propietario del archivo o el usuario root.

Para establecer el permiso sticky bit simbólicamente, ejecuta un comando como el siguiente:

```
chmod o+t <directorio>
```

Para establecer el permiso sticky bit numéricamente, añade 1000 a los permisos existentes del directorio (asume que el directorio en el ejemplo siguiente tenía originalmente 775 para su permiso):

```
chmod 1775 <archivo|directorio>
```

Para retirar el permiso sticky bit simbólicamente, ejecuta:

```
chmod o-t <directorio>
```

Para retirar el permiso sticky bit numéricamente, resta 1000 de los permisos existentes del directorio:

```
chmod 0775 <directorio>
```

La salida del comando `ls -l` visualiza el permiso sticky bit con una `t` en la posición de ejecución de «otros»: `drwxrwxrwt`.

Una `t` minúscula significa que tanto el permiso sticky bit como los permisos de ejecución están establecidos para «otros». Una `T` mayúscula significa que sólo el permiso sticky bit está establecido: `drwxrwxrwT`.

Mientras la `S` mayúscula indica un problema con los permisos `setuid` o `setgid`, una `T` mayúscula no indica necesariamente un problema, siempre y cuando el grupo propietario aún tenga permiso de ejecución.

Un buen ejemplo del uso de directorios con el permiso sticky bit serían los directorios `/tmp` y `/var/tmp`. Estos directorios están diseñados como ubicaciones donde cualquier usuario puede crear un archivo temporal.

Debido a que estos directorios están destinados a ser modificables por todos los usuarios, están configurados para utilizar el permiso sticky bit. Sin este permiso especial, un usuario podría eliminar los archivos temporales de otro usuario.

## 16.7 Los Enlaces Físicos y los Enlaces Simbólicos

Considera el siguiente escenario: hay un archivo profundamente enterrado en el sistema de archivos llamado `/usr/share/doc/superbigsoftwarepackage/data/2013/october/tenth/valuable-information.txt`. Es un archivo que se actualiza de forma rutinaria por otro usuario y al que habitualmente necesitas tener acceso. Introducir este largo nombre de archivo no es una opción ideal, pero el archivo debe residir en este lugar y, ya que se actualiza de forma regular, no se puede simplemente hacer una copia del archivo.

Ésta es una situación en la que los enlaces físicos y los simbólicos (blandos) pueden ser útiles. Se puede crear un archivo que se enlazara al que está «profundamente enterrado». Este nuevo archivo podría ser colocado en tu directorio `home` o cualquier otro lugar conveniente. Cuando accedes al archivo nuevo de «enlace», éste accederá a los contenidos del archivo `valuable-information.txt`.

Cada método de enlace, físico o simbólico, resulta en el mismo acceso global, pero utilizan diferentes técnicas. Existen ventajas y desventajas de cada método, por lo que conocer ambas técnicas y cuando se usan es importante.

16.7.1 Creación de los Vínculos Físicos

Con el fin de entender los enlaces físicos, hay que entender un poco cómo el sistema de archivos realiza un seguimiento de los archivos. Por cada archivo creado, hay un bloque de datos en el sistema de archivos que almacena la información meta del archivo. La *información meta* incluye información sobre el archivo, tal como los permisos, propiedades y marcas de tiempo. Información meta no incluye el nombre del archivo o el contenido del archivo, pero sí incluye casi toda la demás información sobre el archivo.

A esta información meta se le llama la *tabla de inodos* del archivo. La tabla de inodos también incluye punteros a otros bloques en el sistema de archivos llamados *bloques de datos* donde se almacenan los datos.

Cada archivo en una partición tiene un número de identificación único llamado *número de inodo*. El comando `ls -li` mostrará el número de inodo de un archivo. Al igual que los usuarios y grupos, lo que realmente define un archivo no es su nombre, sino más bien el número que se le haya asignado.

La tabla de inodos no incluye el nombre del archivo. Para cada archivo, también hay una entrada que se almacena en el área de datos de un directorio (bloque de datos) que incluye una asociación entre un número de inodo y un nombre de archivo.

En el bloque de datos para el directorio `/etc` habría una lista de todos los archivos de este directorio y su número de inodo correspondiente. Por ejemplo:

|         |     |
|---------|-----|
| passwd  | 123 |
| shadow  | 175 |
| group   | 144 |
| gshadow | 897 |

Cuando se intenta acceder al archivo `/etc/passwd`, el sistema utiliza esta tabla para traducir el nombre de archivo en un número de inodo. A continuación, recupera los datos del archivo al ver la información en la tabla de inodos para el archivo.

Los enlaces físicos son dos nombres de archivo que apuntan al mismo inodo. Por ejemplo, considera las siguientes entradas de directorio:

|          |     |
|----------|-----|
| passwd   | 123 |
| mypasswd | 123 |
| shadow   | 175 |

|         |     |
|---------|-----|
| group   | 144 |
| gshadow | 897 |

Debido a que tanto el archivo `passwd` como el `mypasswd` tienen el mismo número de inodo, en realidad, son el mismo archivo. Puedes acceder a los datos del archivo utilizando cualquiera de los dos nombres de archivo.

Al ejecutar el comando `ls -li`, el número que aparece para cada archivo entre los permisos de usuario y el propietario es la cuenta de enlaces:

```
sysadmin@localhost:~$ echo data > file.original
sysadmin@localhost:~$ ls -li file.*
278772 -rw-rw-r--. 1 sysadmin sysadmin 5 Oct 25 15:42 file.original
```

El número contador de enlaces indica cuántos enlaces físicos fueron creados. Cuando el número es un valor de uno, entonces el archivo sólo tiene un nombre relacionado con el inodo.

Para crear los enlaces físicos, se utiliza el comando `ln` con el primer argumento siendo el nombre de archivo existente y el segundo argumento es el nuevo archivo. Cuando utilizas el comando `ln` para crear un enlace físico, el número del enlace aumentará en una unidad por cada nombre de archivo adicional que enlaces:

```
sysadmin@localhost:~$ ln file.original file.hard.1
sysadmin@localhost:~$ ls -li file.*
278772 -rw-rw-r--. 2 sysadmin sysadmin 5 Oct 25 15:53 file.hard.1
278772 -rw-rw-r--. 2 sysadmin sysadmin 5 Oct 25 15:53 file.original
```

### 16.7.2 Creación de Enlaces Simbólicos

Un enlace simbólico es simplemente un archivo que apunta a otro archivo. Hay varios enlaces simbólicos que ya están en el sistema, incluyendo varios en el directorio `/etc`:

```
sysadmin@localhost:~$ ls -l /etc/grub.conf
lrwxrwxrwx. 1 root root 22 Feb 15 2011 /etc/grub.conf -> ../boot/grub/grub.conf
```

En el ejemplo anterior, se puede ver que el archivo `/etc/grub.conf` «apunta» al archivo `../boot/grub/grub.conf`. Por lo tanto, si intentaras ver el contenido del archivo `/etc/grub.conf`, éste seguiría el puntero y mostraría el contenido del archivo `../boot/grub/grub.conf`.

Para crear un enlace simbólico, utiliza la opción `-s` con el comando `ln`:

```
sysadmin@localhost:~$ ln -s /etc/passwd mypasswd
sysadmin@localhost:~$ ls -l mypasswd
lrwxrwxrwx. 1 sysadmin sysadmin 11 Oct 31 13:17 mypasswd -> /etc/passwd
```

### 16.7.3 Comparación de los Enlaces Físicos y Simbólicos

Mientras que los enlaces físicos y simbólicos tienen el mismo resultado final, las diferentes técnicas producen diferentes ventajas y desventajas. De hecho, la ventaja de una técnica compensa la desventaja de la otra técnica.

### **Ventaja: Enlaces físicos - ningún punto único de fallo**

Una de las ventajas de utilizar los enlaces físicos es que cada nombre de archivo para el contenido del archivo es equivalente. Si tienes cinco archivos enlazados físicamente entre sí, entonces eliminando cualquiera de estos cuatro archivos no daría lugar a la eliminación del contenido del archivo real.

Recordemos que realmente un archivo está asociado con un número de inodo único. Mientras permanezca uno de los archivos enlazados físicamente, entonces ese número de inodo sigue existiendo. Esto significa que aún existen los datos del archivo.

Los enlaces simbólicos, sin embargo, tienen un único punto de fallo: el archivo original. Consideremos el siguiente ejemplo en el que el acceso a los datos fallará si se elimina el archivo original:

```
sysadmin@localhost:~$ echo "hi there" > test.txt
sysadmin@localhost:~$ ln -s test.txt mytest.txt
sysadmin@localhost:~$ more test.txt
hi there
sysadmin@localhost:~$ more mytest.txt
hi there
sysadmin@localhost:~$ rm test.txt
sysadmin@localhost:~$ more mytest.txt
mytest.txt: No such file or directory
sysadmin@localhost:~$ ls -l mytest.txt
lrwxrwxrwx. 1 sysadmin sysadmin 8 Oct 31 13:29 mytest.txt -> test.txt
```

### **Ventaja: Los enlaces blandos - fáciles de ver**

A veces puede ser difícil saber dónde están los enlaces físicos a un archivo. Si ves un archivo normal con un número de enlace mayor que uno, puedes utilizar el comando `find` con el criterio de búsqueda `-inum` para localizar los otros archivos que tienen el mismo número de inodo. Para encontrar el número de inodo, primero utilizarías el comando `ls -li`:

```
sysadmin@localhost:~$ ls -li file.original
278772 file.original
sysadmin@localhost:~$ find / -inum 278772 2> /dev/null
/home/sysadmin/file.hard.1
/home/sysadmin/file.original
```

Los enlaces blandos son mucho más visuales y no requieren ningún comando adicional más allá del comando `ls` para determinar el vínculo:

```
sysadmin@localhost:~$ ls -l mypasswd
```

```
lrwxrwxrwx. 1 sysadmin sysadmin 11 Oct 31 13:17 mypasswd -> /etc/passwd
```

### Ventaja: Los enlaces blandos - pueden vincular cualquier archivo

Debido a que cada sistema de archivos (partición) tiene un conjunto independiente de inodos, no se pueden crear enlaces físicos que intenten cruzar los sistemas de archivos:

```
sysadmin@localhost:~$ ln /boot/vmlinuz-2.6.32-358.6.1.el6.i686 Linux.Kernel
ln: creating hard link `Linux.Kernel' => `/boot/vmlinuz-2.6.32-358.6.1.el6.i686': Invalid cross-device link
```

En el ejemplo anterior, se intentó crear un enlace físico entre un archivo en el sistema de archivos `/boot` y el sistema de archivos `/`; éste falló porque cada uno de estos sistemas de archivos tiene un conjunto único de números de inodo que no se pueden utilizar fuera del sistema de archivos.

Sin embargo, debido a que un enlace simbólico apunta a otro archivo con un nombre de ruta, se puede crear un vínculo simbólico a un archivo en otro sistema de archivos:

```
sysadmin@localhost:~$ ln -s /boot/vmlinuz-2.6.32-358.6.1.el6.i686 Linux.Kernel
sysadmin@localhost:~$ ls -l Linux.Kernel
lrwxrwxrwx. 1 sysadmin sysadmin 11 Oct 31 13:17 Linux.Kernel -> /boot/vmlinuz-2.6.32-358.6.1.el6.i686
```

### Ventaja: Enlaces blandos - pueden enlazar un directorio

Otra limitación de los enlaces físicos es que no se pueden crear en directorios. La razón de esta limitación es que el propio sistema operativo utiliza enlaces físicos para definir la jerarquía de la estructura de directorios. El siguiente ejemplo muestra el mensaje de error que se muestra si se intenta enlazar físicamente un directorio:

```
sysadmin@localhost:~$ ln /bin binary
ln: `/bin': hard link not allowed for directory
```

Enlazando los directorios utilizando un enlace simbólico está permitido:

```
sysadmin@localhost:~$ ln -s /bin binary
sysadmin@localhost:~$ ls -l binary
lrwxrwxrwx. 1 sysadmin sysadmin 11 Oct 31 13:17 binary -> /bin
```

## 16.8 Estándar de Jerarquía del Sistema de Archivos

Entre los estándares soportados por la Fundación Linux está el **Estándar de Jerarquía del Sistema de Archivos (FHS)**, que está alojado en la URL <http://www.pathname.com/fhs>.

Un estándar es un conjunto de reglas o directrices que se recomiendan a seguir. Sin embargo, estas directrices sin duda pueden romperse, ya sea por las distribuciones enteras o por los administradores en las máquinas individuales.

El estándar FHS categoriza cada directorio del sistema de dos maneras:



- Un directorio puede ser categorizado como compatible o no, es decir, si el directorio puede ser compartido en una red y utilizado por varios equipos.
- El directorio se pone en una categoría de tener archivos estáticos (el contenido del archivo no cambiará) o archivos variables (el contenido del archivo puede cambiar).

Con el fin de poder hacer estas clasificaciones, a menudo es necesario hacer referencia a los subdirectorios debajo del nivel superior de los directorios. Por ejemplo, el directorio `/var` en sí no puede ser clasificado como compatible o no compatible, pero uno de sus subdirectorios, el directorio `/var/mail`, es compatible. Por el contrario, el directorio `/var/lock` no debe ser compatible.

|          | No compatible          | Compatible             |
|----------|------------------------|------------------------|
| Variable | <code>/var/lock</code> | <code>/var/mail</code> |
| Estático | <code>/etc</code>      | <code>/opt</code>      |

El estándar FHS define cuatro jerarquías de directorios utilizados en la organización de los archivos del sistema de archivos. El nivel superior o jerarquía root viene seguido por:

| Directorio         | Propósito del Directorio  |
|--------------------|---|
| <code>/</code>     | La base de la estructura, o el root del sistema de archivos, este directorio unifica todos los directorios independientemente si son particiones locales, dispositivos extraíbles o recursos compartidos de red |
| <code>/bin</code>  | Para mantener binarios esenciales como los comandos <code>ls</code> , <code>cp</code> , y <code>rm</code> , y ser parte del sistema de archivos root.   |
| <code>/boot</code> | Contiene los archivos necesarios para arrancar el sistema, como el kernel de Linux y los archivos de configuración asociados  |

| Directorio                    | Propósito del Directorio  |
|-------------------------------|---|
| <code>/dev</code>             | Viene relleno de archivos que representan los dispositivos de hardware y otros archivos especiales, tales como los archivos <code>/dev/null</code> y <code>/dev/zero</code> |
| <code>/etc</code>             | Contiene los archivos de configuración de host esenciales, como los archivos <code>/etc/hosts</code> o <code>/etc/passwd</code>   |
| <code>/home</code>            | La ubicación de los directorios home de los usuarios  |
| <code>/lib</code>             | Las librerías esenciales de soporte para los archivos ejecutables en los directorios <code>/bin</code> y <code>/sbin</code>   |
| <code>/lib&lt;qual&gt;</code> | Las librerías esenciales creadas para una arquitectura específica. Por ejemplo, el directorio <code>/lib64</code> para los procesadores de 64 bit AMD/Intel x86 compatibles |
| <code>/media</code>           | El punto de montaje para los medios extraíbles que se montan automáticamente  |
| <code>/mnt</code>             | Un punto de montaje para montar temporalmente sistemas de archivos de manera manual   |
| <code>/opt</code>             | Ubicación opcional de la instalación de software de terceros  |

| Directorio | Propósito del Directorio  |
|------------|---|
| /proc      | Un sistema de archivos virtual para que el kernel reporte procesos y otra información   |
| /root      | El directorio inicial del usuario root  |
| /sbin      | Los binarios esenciales del sistema utilizados principalmente por el usuario root   |
| /sys       | Un sistema de archivos virtual que contiene información acerca de los dispositivos de hardware conectados al sistema  |
| /srv       | Ubicación donde los servicios específicos del sitio pueden estar alojados   |
| /tmp       | Directorio en el que todos los usuarios tienen permiso para crear archivos temporales que deberían ser borrados durante el arranque (pero a menudo no es así) |
| /usr       | La segunda jerarquía de archivos que no son esenciales para el uso de múltiples usuarios  |
| /usr/local | La tercera jerarquía de archivos para software que no sea originario de la distribución   |

| Directorio              | Propósito del Directorio   |
|-------------------------|--|
| <code>/var</code>       | La jerarquía <code>/var</code> contiene archivos que cambian durante el tiempo |
| <code>/var/cache</code> | Archivos utilizados para almacenar en caché, los datos de la aplicación        |
| <code>/var/log</code>   | El directorio donde se ubica la mayoría de los archivos de registro            |
| <code>/var/lock</code>  | Ubicación para guardar los archivos de bloqueo de los recursos compartidos     |
| <code>/var/spool</code> | Ubicación para almacenar los archivos spool de impresión y correo              |
| <code>/var/tmp</code>   | Los archivos temporales que se deben conservar entre los reinicios             |

La segunda y la tercera jerarquía, que se encuentra bajo los directorios `/usr` y `/usr/local`, repiten el patrón de muchos de los directorios clave que se encuentran debajo de la primera jerarquía o el sistema de archivos root. La cuarta jerarquía, el directorio `/var`, también repite algunos de los directorios de primer nivel, como `lib`, `opt` y `tmp`.

Cuando el sistema de archivos root y su contenido se consideran esenciales o necesarios para arrancar el sistema, los directorios `/var`, `/usr` y `/usr/local` no se consideran esenciales para el proceso de arranque. Como resultado, el sistema de archivos root y sus directorios pueden ser los únicos disponibles en ciertas situaciones, tales como arrancar en modo de usuario único, un entorno diseñado para la solución de problemas del sistema.

El directorio `/usr` sirve para contener software para su uso por varios usuarios. El directorio `/usr` a veces se comparte a través de la red y se monta como de sólo lectura. Los directorios comunes de segundo nivel se describen en la siguiente tabla:

| Directorio                        | Propósito del Directorio   |
|-----------------------------------|--|
| <code>/usr/bin</code>             | Los binarios para el usuario común, usados cuando el sistema está en modo multiusuario                                   |
| <code>/usr/include</code>         | Los archivos que se incluyen para compilar el software de distribución   |
| <code>/usr/lib</code>             | Las librerías de soporte para los archivos ejecutables en los directorios <code>/usr/bin</code> y <code>/usr/sbin</code> |
| <code>/usr/lib&lt;qual&gt;</code> | Las librerías no esenciales creadas para una arquitectura específica   |
| <code>/usr/libexec</code>         | Los programas ejecutables utilizados por otros programas y no directamente por los usuarios                              |
| <code>/usr/sbin</code>            | Los binarios del sistema para su uso por el administrador en modo multiusuario   |
| <code>/usr/share</code>           | Ubicación para almacenar documentación de software y otros datos de aplicación   |
| <code>/usr/src</code>             | El código fuente para compilar el kernel   |

La jerarquía `/usr/local` sirve para la instalación del software que no se origina con la distribución. A menudo, este directorio se utiliza para software que se compila a partir del código fuente. Los directorios de tercer nivel comunes que se encuentran bajo el directorio `/usr/local` se describen en la siguiente tabla:

| Directorio                      | Propósito del Directorio   |
|---------------------------------|--|
| <code>/usr/local/bin</code>     | Los binarios de software locales para el uso de un usuario ordinario   |
| <code>/usr/local/etc</code>     | Los archivos de configuración de software locales  |
| <code>/usr/local/include</code> | Los archivos que necesitan ser incluidos con el fin de compilar el código fuente local   |
| <code>/usr/local/lib</code>     | Los archivos de la librería de soporte para los archivos ejecutables en los directorios <code>/usr/local/bin</code> y <code>/usr/local/sbin</code> |
| <code>/usr/local/libexec</code> | Los programas ejecutables utilizados por otros programas y no directamente por los usuarios  |
| <code>/usr/local/sbin</code>    | Los binarios locales para uso del administrador del sistema  |
| <code>/usr/local/share</code>   | Ubicación para almacenar las páginas man, páginas de información, y otra información de aplicaciones locales                                       |
| <code>/usr/local/src</code>     | La ubicación en la que menudo se coloca el código fuente de software para ser compilado localmente   |

## 16.9 La Organización Dentro de la Jerarquía del Sistema de Archivos

Aunque el estándar FHS es útil para una comprensión detallada de la disposición de los directorios utilizados por la mayoría de las distribuciones de Linux, a continuación se ofrece una descripción más generalizada de la estructura de los directorios como realmente existen en una distribución típica de Linux.

### Los Directorios Home

El directorio `/home` tendrá típicamente un directorio inferior para cada cuenta de usuario. Por ejemplo, el usuario `bob` normalmente tendrá su directorio home de `/home/bob`. Normalmente, sólo el usuario `bob` tendrá acceso a este directorio. Sin ser asignados permisos especiales en otros directorios, un usuario normalmente sólo puede crear archivos en su directorio home, en el directorio `/tmp` y el directorio `/var/tmp`.

### Los Directorios Binarios

Los directorios binarios contienen programas que los usuarios y los administradores ejecutarán para iniciar los procesos o las aplicaciones del sistema. Los directorios binarios, que están destinados a ser utilizados por los usuarios sin privilegios, incluyen los directorios `/bin`, `/usr/bin` y `/usr/local/bin`. A veces el software de terceros también almacenará sus archivos ejecutables en los directorios, tales como `/usr/local/application/bin` y `/opt/application/bin`. Además, no es inusual que cada usuario tenga su propio directorio `bin` ubicado en su directorio home, tal como `/home/bob/bin`.

Por otra parte, los directorios `sbin` están destinados principalmente a ser utilizados por el administrador del sistema (usuario `root`). Estos por lo general incluyen los directorios `/sbin`, `/usr/sbin` y `/usr/local/sbin`, aunque las aplicaciones administrativas de terceros también podrían utilizar directorios como `/usr/local/application/sbin` o `/opt/application/sbin`.

Dependiendo de la distribución, la variable `PATH` puede no contener todos los directorios `bin` y `sbin`. Con el fin de poder ejecutar un comando en uno de estos directorios, el directorio debe ser incluido en la lista de las variables `PATH` o el usuario tiene que especificar la ruta al comando, por ejemplo: `/sbin/ifconfig`.

### Los Directorios de las Aplicaciones de Software

A diferencia del sistema operativo Windows, donde las aplicaciones pueden tener todos sus archivos instalados en un único subdirectorio bajo el directorio `C:\Program Files`, las aplicaciones de Linux pueden tener sus archivos en varios directorios repartidos a lo largo del sistema de archivos de Linux. Para las distribuciones Debian, puedes ejecutar la aplicación `dpkg -I` para obtener la lista de ubicación de los archivos. En las distribuciones de Red Hat, puedes ejecutar la aplicación `rpm -ql` para listar la de ubicación de los archivos que pertenecen a esa aplicación.

Los archivos binarios de los programas ejecutables pueden ir en el directorio `/usr/bin`, si vienen incluidos en el sistema operativo, o de lo contrario pueden ir a los directorios `/usr/local/bin` o `/opt/application/bin` en caso de que procedan de un tercero.

Los datos para la aplicación pueden ser almacenados en uno de los siguientes subdirectorios: `/usr/share`, `/usr/lib`, `/opt/application` o `/var/lib`.

El archivo relacionado con la documentación se puede almacenar en uno de los siguientes subdirectorios: `/usr/share/doc`, `/usr/share/man` o `/usr/share/info`.

El archivo(s) de configuración global para una aplicación muy probablemente se almacene en un subdirectorio bajo el directorio `/etc`, mientras que los archivos de configuración personalizados (específicos para un usuario) para la aplicación están, probablemente, en un subdirectorio oculto del directorio home del usuario.

### Los Directorios de Librerías

Las librerías son archivos que contienen código que se comparte entre varios programas. La mayoría de los nombres de archivo de la librería terminan con una extensión de archivo `.so`, lo que significa *objeto compartido (shared object)*.

Puede haber varias versiones de una librería debido a que el código puede ser diferente dentro de cada archivo a pesar de que puede llevar a cabo funciones similares a las otras versiones de la librerías. Una de las razones por las que el código puede ser diferente, a pesar de que puede hacer lo mismo que otro archivo de la biblioteca, es que está compilado para ejecutarse en un tipo diferente de procesador. Por ejemplo, es habitual que los sistemas que utilizan un código diseñado para los procesadores de tipo Intel/AMD de 64 bits, tengan las dos bibliotecas, la de 32 bits y de 64 bits.

Las librerías que dan soporte a los programas binarios esenciales que se encuentran en los directorios `/bin` y `/sbin` típicamente se encuentran en `/lib` o `/lib64`.

Para dar soporte a los ejecutables `/usr/bin` y `/usr/sbin`, normalmente se usan los directorios librerías `/usr/lib` y `/usr/lib64`.

Para dar soporte a las aplicaciones que no se distribuyen con el sistema operativo, a menudo se utilizan los directorios librería `/usr/local/lib` y `/opt/application/lib`.

### Los Directorios de Datos Variables

Los directorios `/var` y muchos de sus subdirectorios pueden contener datos que vayan a cambiar con frecuencia. Si el sistema se utiliza para correo electrónico, normalmente se utilizará `/var/mail` o `/var/spool/mail` para almacenar los datos de correo electrónico del usuario. Si vas a imprimir desde tu sistema, entonces el directorio `/var/spool/cups` se utilizará para almacenar temporalmente los trabajos de impresión.

Dependiendo de los eventos que el sistema está registrando y la cantidad de actividad que hay en el sistema, se determinará el tamaño de tu archivo de registro. En un sistema ocupado, puede haber una considerable cantidad de datos en los archivos de registro. Estos archivos se almacenan en el directorio `/var/log`.

Mientras que los archivos de registro pueden ser extremadamente útiles para solucionar los problemas, también pueden causar problemas. Una de las principales preocupaciones de todos estos directorios es que pueden llenar rápidamente el espacio del disco en un sistema activo. Si el directorio `/var` no es una partición separada, entonces el sistema de archivos root se podría llenar por completo y bloquear el sistema.

- [Previous](#)
- [Next](#)