Linux: Orígenes y Características

¿Qué es UNIX?

Es una familia de sistemas operativos, la primera de sus versiones fue creada entre 1969 y 1970 en el centro de investigación de Bell Labs. Algunas de las personas que trabajaron desde sus inicios fueron Ken Thompson, Dennis Ritchie, Joe Ossana y Douglas McIlroy.

¿Por qué fue importante UNIX?

Lo fue porque adopto funcionalidades novedosas para la época, tales como la multitarea y la posibilidad de ser utilizado por distintos usuarios al mismo tiempo.
Además, su evolución fue paralela a la del lenguaje C de programación, que contribuyo para que se pueda utilizar en múltiples arquitecturas de computadoras y por una gran cantidad de programadores.
El sistema operativo BSD desarrollado en la Universidad de Berkeley es un ejemplo de la influencia de UNIX. Luego comenzaron a surgir las primeras versiones comerciales de UNIX: System V, HP-UX, Solaris, AIX y Xenix. Algunos conceptos y funcionalidades por UNIX fueron:

- Sistema de archivos jerárquico
- Una Shell que permitiera tanto ingresar comandos como programar scripts
- El uso de archivos de texto de configuración
- La sintaxis de expresiones regulares
- Herramientas que realizan una tarea especifica y de manera eficiente
- Software desarrollado de manera modular
- La suite de protocolos de la red TCP/IP
- Software documentado y con código fuente accesible

A comienzos de la década de 1980, sin embargo, comenzó a ser una práctica comercial la distribución de software solamente en formato binario.

Tipos de sistemas UNIX:

Podemos usar la clasificación por Eric Raymond, que sostiene la existencia de 3 tipos de UNIX

Tipo	Característica	Ejemplos
Genéticos	Son derivados del UNIX original de Bell Labs	FreeBSD, NetBSD, OpenBSD, AIX, Oracle Solaris, macOS, HP-UX
Registrados	Son UNIX marca registrada certificada por The Open Group	Oracle Solaris, macOS, HP-UX
Funcionales	Toman muchas de las funcionalidades y conceptos del UNIX original, pero no hereda su código	**Linux**, Minix

El nacimiento de Linux

Linus Torvalds creo el núcleo Linux en 1991 en la Universidad de Helsinki, Finlandia. Linus comenzó a desarrollar Linux sobre el sistema operativo Minix. Poco tiempo después el reemplaza las herramientas Minix existentes por las desarrolladas en el proyecto GNU. El nombre del kernel fue puesto por un compañero de la Universidad de Helsinki, que es la denominación que finalmente se impuso. El 5 de octubre de 1991, lanza la primera versión considerada publica: la 0.02. Ya podía ejecutar bash (el Shell de GNU) y gcc (el compilador de C de GNU), pero no hacía mucho más.

A finales de 1991 se hizo público Linux con la versión 0.10. Un mes después, en diciembre, apareció la versión 0.11. Linus animo a otras personas a colaborar en su desarrollo. Un paso significativo lo dio en enero de 1991, ya que fue la primera versión (0.12) con la licencia de software libre GNU GLP.
Muchos programadores se unieron al proyecto a través de la naciente Internet. Linux continua su desarrollo hoy en día gracias a un equipo mundial dirigido por Linus, que trabaja a través de internet.
Gran parte del software desarrollado para Linux es creado por el proyecto GNU creado por el estadounidense Richard Stallman.

¿Qué es Linux?
Es el nombre del núcleo de un sistema operativo. En general, también, se extiende este nombre para todo el sistema operativo. Muchos se refieren al sistema operativo como GNU/Linux, ya que consideran que el software del proyecto GNU fue determinante para el desarrollo de Linux y porque el sistema que Stallman iba a crear se llamaría GNU. No obstante, en la actualidad, en cualquier distribución de Linux existe una gran cantidad de software que no proviene del proyecto GNU.
Por otro lado, existen sistemas operativos Linux que no poseen herramientas GNU, por ejemplo: Android y Replicant.
Es decir: existen argumentos a favor y en contra de la denominación "GNU/Linux".
Linux creció a la par del movimiento del software libre y de la iniciativa para el código abierto.
El sistema operativo del pingüino (La mascota de Linux que se llama Tux) se convirtió en poco tiempo en el producto más famoso del software libre, y, también, en la apuesta de mas de una multinacional en el sector informático.

Distribuciones de Linux

La posibilidad de que exista un gran numero de programas de software libre permite la creación de diferentes sistemas Linux.
Es decir, existen usuarios, comunidades, y/o empresas que se encargan de compilar software y convertirlo en un sistema operativo instalable y usable.
Lo importante es saber que existen dos ramas principales de distribuciones de Linux, aquellas basadas en **Debian** y las que están relacionadas de alguna manera en su desarrollo con **Red Hat**.
Debian es una de las primeras distribuciones y data del año 1993, creada por Ian Murdock. Es uno de los sistemas operativos mas respetados y se usa preferiblemente en servidores.
Red Hat Enterprise Linux fue creada por la empresa Red Hat en el año 2003 y apunta más bien al ámbito corporativo. Su antecesora Red Hat Linux fue la primera en proporcionar un sistema sencillo de instalación. La empresa Red Hat cobra suscripciones mediante la cual ofrece soporte comercial.

¿Qué significa FREE?
Es común hablar acerca de la venta de software. Pero en realidad, el software no se vende, sino que se licencia. Es decir, el autor establece una serie de condiciones mediante el cual un determinado programa puede ser utilizado.
Lo que hace tan especial de una distribución de Linux es que la mayoría o todo el software que lo compone posee una licencia de free software (software libre).
En ingles la palabra free puede significar tanto gratis como libre. En el contexto del que estamos hablando quiere decir libre. De acuerdo a la Free Software Foundation un software es libre cuando su licencia permite:

- Utilizar el programa, para cualquier propósito
- Estudiar el funcionamiento del programa y modificarlo, para adaptarlo a sus necesidades
- Redistribuir copias del programa
- Mejorar el programa y poner sus mejoras a disposición del público, para beneficio de toda la comunidad

Por lo tanto, cuando hablamos de software libre no tiene relación necesaria con el precio del mismo. Si bien una buena cantidad de software libre lo podemos obtener gratuitamente de Internet y es perfectamente legal hacer copias, muchas empresas viven del software libre.
La rentabilidad de estas empresas esta basada en ofrecer servicios alrededor del software que proporcionan, algunas de ellas son: Red Hat, Novell, AdaCore, Oracle, etc.
Es importante remarcar que el software libre tiene licencias. Una licencia de software libre debe ofrecer las cuatro libertades arriba mencionadas. Existen muchas licencias de software libre, pero podemos dividirlas en dos categorías:

- Copyleft: Obliga a que las obras derivadas se distribuyan bajo los mismos términos, por ejemplo: la Licencia de Publico General GNU (GLP)
- No-Copyleft: Permiten que se creen obras derivadas con otra licencia, por ejemplo, las licencias BSD.

Linux: Características

Multitarea total
Se pueden ejecutar varias tareas y se puede acceder a varios dispositivos al mismo tiempo.

Memoria virtual
Linux puede simular una mayor cantidad de memoria, gracias a un método llamado virtualización de memoria.

Soporte multiusuario
Linux permite que varios usuarios accedan a su sistema simultáneamente sin que haya conflicto entre ellos y cada uno con su espacio de trabajo.

Código fuente libre
Linux cuenta con miles de programas, librerías y software de diferente tipo con licencias de software libre.

Adaptabilidad
Puede ejecutar una amplia variedad de software, disponible gracias a una gran cantidad de proyectos existentes. Este software incluye, desde compiladores (GNU C y GNU C++) a la administración del sistema y redes (GNU coreutils, gawk, CUPS, SAMBA, etc), juegos (pychess, Kigo y NetHack) y herramientas de productividad como LibreOffice.

Estabilidad
Presenta una gran estabilidad en la gestión de sus procesos internos del sistema. Es muy difícil conseguir que Linux se “cuelgue” y, por supuesto, jamás se verá una “pantalla azul”.

Gran oferta de software
Una misma distribución puede utilizarse para una PC de escritorio o notebook con entorno gráfico y herramientas de productividad. Y sin dudas para servidores poseen software para hasta un servicio de correo, sitio web, almacenamiento, firewall, etc.

Defensa contra los virus

Aunque la mayor parte de los virus que rondan por internet son desarrollados para Windows, es cierto que, existen algunos para Linux, pero son más difíciles de crear debido a que Linux emplea un sistema de permisos de archivos previendo los posibles desastres que se ven todos los días en los entornos de Windows. Además, los virus para Windows no se pueden ejecutar en Linux. Y aunque existiera un virus para Linux, este solo podría dañar los archivos del usuario que fue infectado, pero nunca el sistema en sí mismo.

Relación con Internet

Debido a que Linux creció junto a Internet, es un sistema naturalmente orientado al mismo y a las redes informáticas en general.

Entornos gráficos

Si bien en servidores es recomendable usar solamente interfaces de texto, cuando se lo usa desde estaciones de trabajo tenemos a disposición entornos gráficos muy amigables. Hay distribuciones de Linux con entorno grafico que datan desde 1992. Pero a diferencia de otros sistemas operativos, Linux cuenta con una variedad de entornos de escritorio y manejadores de ventanas. Los principales entornos de escritorio son KDE Plasma, GNOME y XFCE.

Servidores en equipos pequeños

En algunos escenarios, tales como en pequeñas empresas, o incluso en hogares podemos tener un servidor web o servidor de archivos, o firewall con recursos modestos con Linux.

La comunidad Linux

Linux tiene una comunidad muy activa de desarrolladores, los cuales muchos de manera voluntaria están dispuestos a responder consultas por medio de listas de correo, fotos o canales de chat.

Soporte de hardware

En la actualidad la mayor parte del hardware esta soportado por Linux. Sin embargo, antes de comprar, es conveniente verificarlo en sitios web que poseen listas de compatibilidad con Linux.

Recursos de Software para Linux

Hay al menos tres modos de reemplazar una herramienta con licencia privativa que corre solamente en Windows. Supongamos que deseamos reemplazar Microsoft Word:

- El primero es el de dar las mismas o similares funcionalidades que las que tienen esas aplicaciones que existen en Windows, aunque su aspecto externo sea diferente al de las aplicaciones conocidas en Windows. Ejemplos: Calligra Words y AbiWord.
- La segunda manera añade además la similitud visual y funcional con la que funciona en Windows, con el fin de que el usuario se encuentre en un entorno familiar. Ejemplos: LibreOffice y OnlyOffice.
- El tercero es el más drástico ya que se crean herramientas con aspectos, funcionalidades y modos de trabajo totalmente diferentes a estas aplicaciones. Ejemplo: usar Markdown (también conocido como CommonMark), un lenguaje de marcas ligero (mucho más sencillo que el HTML) que permite de acuerdo al editor, exportar el texto a HTML o a PDF. Existen varias herramientas que usan Mardown, una de ellas es Mark Text.

Maneras de instalar Linux

Existen muchas maneras de instalar Linux. Una de ellas es mediante un CD o DVD que lanza meramente un instalador. Otra es mediante un LiveCD. Un LiveCD es un sistema Linux que se ejecuta – en principio – sin tocar en absoluto los discos rígidos y permite probar el sistema operativo utilizando la memoria RAM. En general, los LivesCDs poseen un programa instalador que copia el sistema operativo al disco para usarlo de manera permanente.
Los Linux de tipo Live se pueden arrancar también desde una unidad USB removible.

Maneras alternativas de instalar Linux
Además de instalarlo directamente en equipos Intel compatibles, en la actualidad es bastante común instalar o implementar Linux como máquina virtual, algunos de estos ejemplos son:

- VirtualBox
- VMWare
- KVM
- Lxc
- Docker

En estos casos generalmente se usa un archivo ISO en lugar de un disco óptico físico.
Además, el sistema se puede instalar en el propio hardware en una nube privada, usando por ejemplo OpenStack, o bien instalarse en un proveedor externo de infraestructura, algunos ejemplos son:

- Amazon EC2
- Rackspace
- Digital Ocean

Instalación y primer uso de Linux

¿Qué es particionar?
Cuando vamos a realizar una instalación de nuestro sistema operativo Linux, necesitamos particionar nuestro disco rígido.
Particionar significa dividir nuestro disco en varias partes, donde cada una de ellas se utilizará para instalar determinadas secciones de nuestro sistema.

Tipos de particiones MBR
Hay diferentes tipos de particiones

- Primarias
- Extendidas
- Lógicas

Las particiones que se crean deben tener algún tipo de filesystem (sistema de archivos): ext2, ext3, ext4, btrfs, xfs, etc.

Particiones Primarias
Todos los discos duros que tengan un sistema de archivos, usan una partición primaria. Es la primera partición creada en el disco. Si todo el espacio es utilizado por la partición primaria, esta será la única partición del disco. Es posible tener varias particiones primarias en un único disco físico. Estas particiones se utilizan para arrancar el sistema y están limitadas a un máximo de cuatro en un mismo disco rígido.

Particiones Extendidas
Si se necesitan más de cuatro particiones en el disco, es necesario crear una partición

extendida. Cuando existe una partición extendida en el disco, no puede haber mas de 3 particiones primarias en el mismo. Una partición extendida por si misma carece de utilidad. En realidad, actúa como un contenedor de particiones lógicas, y puede contener varias particiones lógicas. Estas particiones no son arrancables (boot), pero permiten tener un gran número de particiones en el sistema. Las particiones lógicas solo pueden existir dentro de una partición extendida.

Tablas de Particiones GPT

Es un estándar para la colocación de la tabla de particiones en un disco duro físico. Es parte del estándar Extensible Firmware Interface (EFI) propuesto por Intel para reemplazar el viejo BIOS del PC, heredada del IBM PC original.
La GPT sustituye al Master Boot Récord (MBR) usado con el BIOS.
Una de las principales ventajas de GPT es la posible capacidad del disco duro. Las unidades MBR solo pueden manejar 2 TB de datos o menos. GPT puede ir más allá de esta capacidad.
Por otra parte, las particiones MBR solo permiten a los usuarios definir cuatro particiones primarias. El usuario puede utilizar una partición extendida para subdividir el disco duro.

Ventajas de GPT

- Utiliza GUID (UUID) para identificar los tipos de particiones
- Proporciona un GUID único de disco y un GUID único de partición para cada partición
- Numero arbitrarios de particiones (depende del espacio asignado por la tabla de particiones). No hay necesidad de particiones extendidas y lógicas. Se pueden crear un número ilimitado de particiones, aunque en general las herramientas de particionado permiten de manera predeterminada hasta 128 particiones
- Utiliza 64-bit LBA para almacenar números del Sector – tamaño máximo del disco manejable es de 2 ZiB.
- Almacena una copia de seguridad del encabezado y de la tabla de particiones al final del disco que ayuda en la recuperación en el caso de que los primeros estén dañados
- Checksum CRC32 para detectar errores y daños de la cabecera y en la tabla de particiones

¿Qué y cuantas particiones?

Primero tenemos que saber que son los **puntos de montaje**: Son aquellos directorios que sirven para acceder a una determinada partición.
A la hora de la instalación, cuando hacemos la partición de nuestro disco, hay que seleccionar en que particiones se van a “montar” determinados directorios.
El particionado más básico es de una partición para el directorio raíz y otra partición para swap. Se podría prescindir incluso de crear la partición para swap y utilizar un archivo, pero eso generalmente no es recomendable.
La partición swap se utiliza como memoria RAM de reserva adicional. Su tamaño debería oscilar entre 1x a 2x la memoria RAM. Cabe aclarar que la memoria swap no usa un directorio como punto de montaje y que es invisible en el sistema de archivos.
Dependiendo del uso que se le quiera dar, Linux puede instalarse con más particiones del mínimo requerido. Los directorios que son susceptibles de ponerse en una partición especifica son:

- /boot: Útil para usarlo como solo lectura para evitar borrar por accidente el contenido de este directorio (este directorio contiene el kernel). Generalmente alcanza con unos 500 MB

- /boot/efi: Esta partición es obligatoria si se posee UEFI en lugar de BIOS (debe tener unos 256 MB y estar formateada en FAT32 o su equivalente en Linux: vfat)
- /home: Este directorio contiene los documentos y configuraciones de los usuarios, de manera que es indicado para estaciones de trabajo y computadoras de usuario final
- /usr
- /var

Si no estamos seguro aun de la cantidad de particiones que vamos a necesitar podemos optar por el método de particionado automático que nos ofrece la distribución.

Particiones que debemos conocer

Partición /boot y partición ESP

Antiguamente existía una limitación técnica que impedía acceder más allá del cilindro 1024. En la actualidad esa restricción prácticamente no existe. Por otro lado, cuando se usa UEFI, se necesita una partición llamada ESP (EFI System Partition) en el directorio /boot/efi formateada como FAT32 o vfat. Esa partición la realizan automáticamente la mayoría de las distribuciones al detectar UEFI.

Particiones de Intercambio (swap)

Este tipo de partición hace referencia al área de intercambio, que se conoce como memoria virtual.
Este tipo de partición la utiliza el sistema para cuando nuestra memoria RAM se encuentra limitada, utilizando este espacio para poder guardar lo que necesita y luego cargarla en RAM. Esta no es la mejor solución, dado que la lectura/escritura en disco, es muy lenta. En la utilización de distribuciones de escritorio es muy posible que veamos mayor utilización de SWAP. En caso de tener poca RAM los objetos menos utilizados serán almacenados en SWAP. En caso de servidores, al detectar uso de swap de manera reiterada/continua, lo recomendables es adquirir más memoria RAM.
Como valor genérico se puede configurar una partición de swap de 4 GB hasta 8 GB. Algunos casos puntuales requieren un tamaño especifico de SWAP, por ejemplo, al instalar una base de datos de Oracle.

Nomenclatura de discos y particiones

En Linux las particiones y los discos no se identifican por letras. En Linux en cambio cada disco y cada partición es un archivo. El siguiente cuadro muestra algunos ejemplos:

Primer disco	/dev/sda
Segundo disco	/dev/sdb
Primera partición primaria (o extendida) del primer disco	/dev/sda1
Primera partición lógica del primer disco	/dev/sda5

Una partición lógica siempre tiene un número mayor o igual a 5.

¿Qué es LVM?

LVM (Logical Volume Manager) es un gestor de volúmenes lógicos para el sistema operativo Linux y es una manera de asignar espacio de un medio en volúmenes Lógicos que pueden ser fácilmente redimensionados.

Ventajas de LVM

- *Flexibilidad*: Redimensionamiento de manera fácil
- *Disponibilidad*: Pueden agregarse discos y migrar datos fácilmente
- Instantáneas para backup (snapshots)

Formatos de las particiones

Sea que usemos particiones primarias, lógicas o volúmenes lógicos los archivos de un sistema Linux debe estar formateado con algunos de los siguientes sistemas de archivos:

- Xfs
- Ext2, ext3 o ext4 (usar este último preferiblemente)
- Btrfs
- Jfs

Estos sistemas de archivos decimos que son nativos, es decir, pueden ser usados por archivos del sistema operativo.

Pasos comunes en una instalación de Linux
Mas allá de la distribución seleccionada existen una seria de pasos comunes a la hora de instalar Linux, ellos son:

- Seleccionar modo de instalación
- Seleccionar el idioma
- Configurar idioma y distribución del teclado
- Elegir esquema de particionado
- Instalar el cargador de arranque
- Establecer contraseña de root
- Crear usuario distinto de root
- Selección de software a instalar

Usar el sistema instalado

¿Cómo se empieza a trabajar en Linux?
Una vez que el sistema operativo está instalado deberemos loguearnos, es decir iniciar una sesión de usuario. A grandes rasgos hay dos maneras de loguearse: Usando una terminal virtual de texto o una terminal gráfica.
De manera simplificada, una terminal virtual es una pantalla virtual. En Linux en la mayoría de las distribuciones existen 6 terminales de texto y una gráfica (esto último solo si el entorno grafico está instalado). En algunas distribuciones la terminal grafica está en la terminal 1, mientras que en otras está en la 7.
Para pasar de una terminal de texto a otra se usa el atajo de teclado **Alt+Fn** donde Fn es una tecla del teclado y "n" es el número de terminal, por ejemplo, si estamos en la terminal de texto 2 y queremos ir a la terminal 3, usaremos el atajo **Atl+F3**.
Para pasar de una terminal grafica a una de texto puro (la 4 por ejemplo), es necesario ejecutar el atajo de la siguiente forma: **Ctrl+Alt+F4**.
En cada una de esas terminales tenemos la posibilidad de loguearnos y comenzar a trabajar. Para hacerlo, es necesario ingresar o seleccionar el nombre de usuario y contraseña.
Si se usa el entorno grafico existen programas que proporcionan terminales virtuales, por ejemplo, en GNOME se puede acceder mediante el atajo de teclado **Alt+F2** y luego ingresando **gnome-terminal** y presionando la tecla Enter.

¿Cómo sé con qué usuario estoy trabajando?
Una vez logueado aparecerá inmediatamente antes del cursor un carácter que puede ser # (si el usuario es root) o un $ (en el caso de que el usuario sea uno común). De todas maneras, ya que el prompt es personalizable, lo mejor es usar el siguiente comando:

```
# whoami
```
Root

¿Como veo la hora actual?
Se puede ver con el comando date:

```
# date
```
Mar may 16 16:50:14 ART 2020

¿Como apago y reinicio el sistema?
Para apagar se usa el comando **poweroff**:

```
# poweroff
```

Si estas como un usuario distinto de root se puede apagar el sistema de la siguiente manera, esto varía según la distribución:

```
$ su -c "poweroff"
```

En este caso se pedirá la contraseña de root.
El comando **su** sirve para ejecutar comandos como otro usuario y lo usaremos para iniciar sesión como root así:

```
$ su –
```

Si un comando nos da error de permisos, probablemente sea que necesitamos hacerlo como root.

Otros comandos básicos
dir: Muestra el contenido de directorios
cal: Muestra un calendario
exit: Cierra una terminal
factor: Factoriza números
less: Visor de texto (se sale con la tecla q)
logout: Cierra terminal con sesión (es decir una en la que el usuario uso **su –** o en la que se logeo)
more: Visor de texto más antiguo que les (se sale con la tecla q)
pwd: Muestra la ruta del directorio actual
tyy: Muestra el archivo especial correspondiente con la terminal virtual utilizada
users: Muestra los usuarios actualmente logueados

Administración básica de archivos

Objetos del sistema de archivos
Los sistemas de archivos que vamos a explicar poseen objetos que se agrupan en un árbol jerárquico. Esta organización nos brinda la posibilidad de contener diversos archivos, donde algunos tendrán el mismo nombre, pero al estar organizados en una forma de árbol jerárquico podremos realizar diversas tareas; entre ellas, "ver" nombres de archivos iguales. La manera

en que se administran estos objetos es mediante tablas propias del sistema de archivos, las cuales tienen diferentes propiedades.

Directorios y archivos
Los directorios son contenedores que van a incluir información de los objetos, tanto de directorios como de otros archivos. En Linux, el directorio de mayor jerarquía es la raíz "/", a partir de ahí surgen todos los demás directorios y archivos. Un archivo, por otro lado, es el que va a contener la información; en cambio, un directorio, nos da un orden de como vamos a poder ver los archivos y/o directorios. Para poder especificar una ruta se antepone "/" porque todo nace de ahí; en cambio, si quisiéramos nombrar un archivo de nuestro home, la ruta completa del archivo seria la siguiente: /home/nombre_usuario/test.sh
No debemos confundir root filesystem con root home directory.

Inodos
La forma en que el sistema de archivos identifica a los objetos es mediante los inodos. Estos traen información acerca de los objetos; cada uno tiene distintas propiedades, que, dependiendo de su uso, va a cambiar el valor que tenga.
El comando **stat** es de gran utilidad para mostrar esta información, por ejemplo:

```
# stat /etc/resolv.conf
```

Este comando nos informa sobre las propiedades de un archivo:

- Nombre
- Tamaño
- Tipo de archivo
- El número de inodo
- La cantidad de enlaces, que en archivos regulares refiere a cuantos archivos tienen el mismo número de inodos
- Permisos y propietarios
- Marcas de tiempo
    - Acceso (atime): La ultima vez que se accedió al archivo
    - Modificación (mtime): La ultima vez que se modifico el contenido de un archivo
    - Cambio (ctime): La ultima vez que se cambiaron metadatos del archivo
    - Creación: La fecha de creación tradicionalmente no se usa en los sistemas de archivos de Unix

Comandos básicos

Comando ls
Este se utiliza para mostrar el contenido de directorios, como así también ver propiedades de los archivos.
Sintaxis:

ls [opciones] [archivo1] [archivo2] [archivo3]
ls [opciones] [directorio1] [directorio2]

Las opciones mas utilizadas son:

- **-a**: Muestra todos los archivos, incluso los ocultos
- **-l**: Muestra el tipo de archivo, permiso, propietarios, tamaño, fecha de modificación

- **-h**: Muestra el tamaño de los archivos en una unidad fácil de leer
- **-r**: Ordena los archivos de manera inversa
- **-t**: Ordena los archivos por fecha de modificación

Ejemplo:

```
# ls -ltr /etc/X11/
```

Comando cp
Se utiliza para copiar archivos/directorios
Sintaxis:

cp [opciones] archivo1 archivo2
cp [opciones] archivos directorio

Opciones más frecuentes:

- **-f**: Fuerza la reescritura si ya existe el destino
- **-i**: Me habilita el "promp" interactivo para evitar la sobreescritura
- **-p**: Preserva todos los permisos de la estructura del directorio y/o archivo, incluyendo la fecha de creación
- **-r, -R**: Hace una copia recursiva, para poder copiar todo un directorio y sus subdirectorios
- **-v**: modo "verbose" en donde muestra todo lo que está haciendo

Entonces, lo que podemos hacer es copiar un archivo para que en el destino tenga otro nombre; también podemos copiar varios archivos a un directorio.

Comando mkdir
Se utiliza para crear directorios.

Sintaxis:

```
# mkdir prueba
```

Ejemplo para crear un directorio y una cadena de subdirectorios:

```
# mkdir -p es/otro/directorio/creado
```

Comando mv
Se utiliza para mover o renombrar archivos y directorios.

Sintaxis:

mv [opciones] origen destino

Opciones más usadas:

- **-f**: Si ya existe lo fuerza a sobrescribir
- **-i**: Activa el modo interactivo para aceptar los cambios uno por uno

Ejemplo para cambiar de nombre:

```
# mv archivo1.txt nuevonombre.txt
```

Ejemplo para mover:

```
# mv archivo1.txt /directorio
```

Comando cd
Se utiliza para avanzar o retroceder niveles de directorios.

Ejemplo:

```
# cd dir # (cambio a un directorio hijo)
# cd .. # (retrocedo)
# cd # (me lleva a mi home)
```

Comando rm
Sirve para borrar tanto directorios como archivos

Sintaxis:

rm [opciones] archivos

Opciones más usadas:

- **-f**: Hace un borrado sin preguntar nada
- **-i**: En modo interactivo
- **-r, -R**: Para borrar recursivamente

Ejemplo para borrar un directorio y TODO su contenido:

```
# rm -rf /directorio
```

Comando rmdir
Se utiliza solamente para borrar directorios vacíos.

rmdir [opciones] directorio

```
# rmdir dir
```

Nota: Generalmente se utiliza el comando rm -rf

Comando touch
Cambia la fecha/hora de acceso o modificación de los archivos.

touch [opciones] archivo

Opciones:

- **-a**: Solo cambia el tiempo de acceso
- **-m**: Cambia el de modificación
- **-t**: Timestamps definir la fecha a utilizar [[CC]YY]MMDDhhmm.[.ss]

Ejemplo:

```
# touch -t 200101121845 archivo1
```

Comodines
Los comodines o meta caracteres (file blobbing) nos van a servir para tomar acción sobre múltiples archivos/directorios que contengan cierta cadena de caracteres en su nombre. Tener en cuenta que los comodines son interpretados por la Shell y no por los comandos, cuando un

comando es puesto con comodines, la Shell los interpreta a estos y otros tipos de expansiones, para luego ejecutar el comando. Este proceso es invisible para el usuario.

Comodín	Descripción	Ejemplo
*	Coincide con cualquier cadena de texto, incluso con la cadena vacía. De manera predeterminada se excluyen los nombres de archivos que comienzan con "."	`rm *` (borrará todos los archivos en el directorio en el que estemos parados) rm "ho (borrará todo lo que comience con ho) `rm *la` (borrará todos los archivos que terminen con la cadena de texto 'ho')
?	Coincide con un único carácter	`ls -l /bin/z???` va a coincidir con archivos que comienzan con z seguidos exactamente 3 caracteres
[lista o rango]	Coincide con cualquier de los caracteres encerrados entre corchetes	`ls -l /bin/[a-c][ar]??` va a coincidir con los archivos cuyo nombre empieza con a, b, ó c, el segundo carácter puede ser una a o una r. Los dos últimos caracteres pueden ser cualquiera.
[!lista o rango]	Coincide con cualquiera de los caracteres no encerrados entre corchetes	`ls -l /bin/[a-c][ar]??` no va a coincidir con los archivos cuyo nombre empieza con a, b, ó c, el segundo carácter puede ser una a o una r y con dos caracteres finales
{ cadena1, cadena2... }	Se usa para generar cadenas de texto arbitrarias	`mkdir dir{1,2,3,4}` generará los directorios dir1, dir2, dir3 y dir4.

Comando gzip / gunzip

Con estos comandos vamos a poder comprimir y descomprimir archivos (utiliza el algoritmo Lempel-Ziv). Gzip es una de las herramientas más antiguas de compresión en Linux.
La sintaxis es:

gzip [opciones] [archivos]

Las opciones más comunes son:

- **-d**: Para descomprimir un archivo (equivale al comando gunzip)
- **-1**: La compresión más rápida
- **-9**: La mejor compresión

Comando bzip2 / bunzip2

Con este comando vamos a poder comprimir y descomprimir archivos (utiliza el algoritmo

Burrows-Wheeler y Huffman). Bzip2 es considerado uno de los mas eficientes programas de compresión disponibles para Linux y su sintaxis y opciones es prácticamente idéntica a gzip/gunzip.
Generalmente tiene la extensión bz2. Tener en cuenta que el archivo mantendrá el nombre y se le agregará dicha extensión.
Ejemplos de cómo usarlo:

La opción -d es para descomprimir, también se puede usar el comando bunzip2.
Con la opción -9 (va de 1 a 9) se define el nivel de compresión. El 9 será la compresión máxima y el 1 la minina.

Comando xz
Esta herramienta usa el algoritmo LZMA, ofrece altos grados de compresión de archivos, así como también, velocidad para la compresión y descompresión.
Para comprimir un archivo se debe hacer lo siguiente:

```
$ xz lxle.iso
```

Para descomprimir un archivo:

```
$ xz -d lxle.iso.xz
```

Al igual que los comandos gzip y bzip2, xz se puede combinar con **tar** para generar un archivo con extensión .tar.xz.

Comando tar
Los comando bzip, gzip y xz pueden comprimir un único archivo a la vez. Este comando sirve para poder agrupar archivos y directorios, y comprimirlos con esas herramientas. El archivo resultante suele llamarse **tarball**.

Opciones:

- **c**: Para crear
- **x**: Para extraer
- **v**: Modo verbose, muestra lo que está haciendo
- **f**: Define el archivo.tar
- **z**: Comprime/Descomprime con gzip
- **j**: Comprime/Descomprime con bzip2
- **J**: Comprime/Descomprime con xz
- **a**: Usa la extensión del archivo para elegir la herramienta de compresión
- **t**: Muestra el contenido del tarball

Generalmente las opciones de un comando van precedidas por guiones medios (-), sin embargo, para el comando tar podemos prescindir de ellos en muchos casos.

Comando cpio
Se usa para crear y extraer archivos o copiar archivos desde un lugar hacia otro. No utiliza compresión por si solo, para comprimirlo hay que hacerlo con bzip/bzip2. Tener en cuenta que nos referimos a archivos en el sentido de paquetes de archivos.

Sintaxis:

cpio -o [opciones] < [filenames …] > [archivo]
cpio -i < [archivo]
cpio -p [destino-directorio] < [filenames…]

Opciones:

- **-o**: Este modo se usa para crear un archivo
- **-i**: Este modo se usa para copiar todo lo que creamos antes
- **-p**: No crea un archivo, solo copia archivos de un lado a otro
- **-u**: Sobrescribe archivos existentes al extraer
- **-m**: Preserva los permisos
- **-d**: Como copio extrae de manera "plana" los archivos, con esta opción respeta la estructura original de directorios del paquete.

La ventaja de este comando con respecto al tar es que se puede tener mayor control, cuando realizamos scripts, sobre los archivos o tipos de archivos que no se desean copiar. Suele combinarse con el comando **find**. No obstante, su uso es menos frecuente que el de tar ya que este último es mucho más sencillo de usar.

Comando dd
Es muy útil para realizar copias de seguridad/clonado de dispositivos de almacenamiento porque copia en bloques.

Sintaxis:

dd [opciones]

- **if=archivo**: Toma como entrada un archivo o dispositivo
- **of=archivo**: Salida a un archivo en vez de la salida estándar
- **ibs=n**: Leer de a "n" bytes
- **obs=n**: Escribir de a "n" bytes
- **conv=list**: Convertir ciertos aspectos de lo que copio

Ejemplos:

Crear una imagen de un CD/DVD:

```
# dd if=/dev/cdrom of=/tmp/cd.iso
```

Crear una imagen de un disco rígido:

```
# dd if=/dev/sda1 of=/directorioparticion/disco2.img
```

Recuperar el contenido de un disco:

```
# dd if=disco1.img of=/dev/sda
```

Recuperar el contenido de una partición:

```
# dd if=disco2.img of=/dev/sda1
```

Copiar el archive file a file2 convirtiendo todo su contenido a minúsculas:

```
# dd if=/tmp/file of=/tmp/file2 conv=1case
```

Copiar una determinada cantidad de bytes de un archivo:

```
# dd if=/dev/zero of=/root/bigfile bs=100M count=10
```

Este último caso se crea un archivo de 1000M con bytes nulos

Comando file
Este comando nos va a devolver que tipo de archivo es el que le preguntemos, sin importar su extensión, lo detecta por su contenido.

Sintaxis:

File [opciones] [file]

Opciones:

- **-f**: Busca por cada línea que hay en el archivo que se le invoque
- **-z**: Intenta ver adentro de los archivos comprimidos

Páginas del Manual

Las páginas del manual o páginas de **man** de GNU/Linux son el mejor lugar para resolver cuestiones relativas a la sintaxis y opciones de los comandos y utilidades del sistema. Los documentos de las paginas **man** están almacenados en un formato comprimido. El comando **man** descomprime y da formato a estas paginas para su visualización. Se accede a estas paginas utilizando el comando **man** seguido del nombre del comando que se quiere consultar. Un ejemplo de la sintaxis de este comando es el siguiente:

```
$ man ls
```

Este comando buscara todas las paginas del manual relativas al comando ls. Cuando abras las paginas del manual, lo primero que se visualizara es un banner con el comando y la pagina de man que esta siendo consultada. También se muestra aquí el logo FSF de Free Software Foundation.
Una vez utilizado el comando, podemos avanzar paginas en el man, utilizando la barra espaciadora, que avanzara una pagina por cada pulsación. La tecla “q” provoca la salida del man. Si queremos buscar algún texto dentro de la pagina man, puedes utilizar las expresiones regulares. Ejemplo de como buscar la palabra “option”.

/option

Encontrando las paginas man
Las paginas man se almacenan en el sistema. La variable MANPATH indica la ubicación de estos archivos. Las paginas man se guarda en los siguientes lugares:

/usr/share/man/man1
/usr/share/man/man2
/usr/share/man/man3
/usr/share/man/man4
/usr/share/man/man5
/usr/share/man/man6
/usr/share/man/man7
/usr/share/man/man8
/usr/share/man/man9

El usuario puede especificar un **MANPATH** diferente. Esto permitiría utilizar un conjunto diferente de páginas man. Esto es practico porque algunos comandos podrían almacenar sus páginas man en lugares distintos a los estándares.

Opción	Descripción
-C configuración	Indica un fichero de configuración distinto a /etc/man.conf.
-M ruta	Indica en qué directorios se buscarán las páginas man.
-P paginador	Indica el paginador (Programa que da formato y visualiza las páginas). Por defecto es el indicado en la variable de entorno **PAGER**. Los paginadores **more** y **less** son los más utilizados.
-S lista	Indica una lista de las secciones a buscar separadas por dos puntos (**:**)
-a	Indica que han de mostrarse todas las entradas coincidentes y no solo la primera.
-c	Indica que la página fuente ha de ser reformateada.
-d	Mostará información de debug en lugar de las páginas man.
-f	Indica que el programa man debe comportarse como el programa **whatis** (se explicará mas adelante).
-h	Muestra información sobre el comando man.
-k	Indica que el programa man debe comportarse como el programa apropos (se explicará mas adelante).
-K	Busca una cadena especificada en las páginas man. Por cada entrada encontrada se le pregunta al usuario si desea verla
-m	Indica un conjunto alternativo de páginas man basado en el sistema especificado.
-w	Indica que ha de visualizarse el path de las páginas man

Secciones de las páginas de man

Sección	Descripción
1	Comandos y aplicaciones del usuario.
2	Llamadas del sistema y errores del Kernel.
3	Llamadas a librerías.
4	Drivers de dispositivos y protocolos de red.
5	Formatos estándar de archivos.
6	Juegos y demos.
7	Ficheros y documentos misceláneos.
8	Comandos de administración del sistema.
9	Especificaciones e interfaces oscuros del kernel.

Cuando se le pasa un argumento al comando man, este busca dentro de las secciones siguiendo un orden especifico y se devuelve la primera coincidencia. El orden de búsqueda por defecto es el siguiente:

1, 8, 2, 3, 4, 5, 6, 7, 9.

También es posible indicar la sección en la que queremos buscar. Si quisiéramos buscar información sobre la utilidad “crontab” en la sección cinco utilizarías el siguiente comando:

```
# man 5 crontab
```

Si usamos la opción -a podremos examinar todas las entradas coincidentes. Siguiendo el ejemplo anterior, tendríamos que utilizar el siguiente comando:

```
# man -a crontab
```

Whatis

La utilidad whatis se usa para buscar entradas coincidentes en la base de datos de whatis. Esta base de datos se crea utilizando el comando /usr/bin/makewhatis. Esta base de datos contiene las descripciones cortas que se encuentran en las páginas man de los comandos del sistema. Un ejemplo de su uso es el siguiente:

```
# whatis passwd
passwd (1) – Update a user’s authentication token(s)
passwd (1ssl) – compute password hashes
passwd (5) – password file
passwd.nntp [passwd] (5) – passwords for connecting to remote NNTP servers
```

Como Podemos ver en el ejemplo, el comando passwd tiene entradas en las secciones 1 y 5 de las paginas man. También ha sido encontrado en sección 1 de las páginas de man del comando ssl.

El comando **man -f** busca en esta base de datos entradas coincidentes con la palabra clave indicada. Por ejemplo:

```
# man -f passwd
```

Podremos observar un resultado igual al anterior. Pero este puede ser practico para localizar secciones de páginas man y variantes de los comandos.

Apropos

Este comando se emplea para buscar tanto los nombres de comando como las descripciones para la palabra clave indicada. Ejemplo:

```
# apropos password
chpasswd (8) – update password file in batch
gpasswd (1) – administer the /etc/group file
htpasswd (1) – Create and update user authentication files
nwpasswd (1) – Change a user’s password
… (Salida del comando recortada)
```

Un ejemplo del comando man -k:

```
# man -k password
chpasswd (8) – Update password file in batch
gpasswd (1) – Administer the /etc/group file
htpasswd (1) – Create and update user authentication files
nwpasswd (1) – Change a user’s password
passwd (1) – Update a user’s authentication tokens
```

passwd (1ssl) – Compute password hashes
passwd (5) – Password file
passwd.nntp [passwd] (5) – Passwords for connecting to remote NNTP servers
… (Salida del comando recortada)

Como Podemos ver, el comando también nos da el mismo resultado, nuevamente, esto puede ser practico cuando buscamos comandos a partir de determinadas palabras claves.

Búsqueda y ubicación de archivos

¿Qué es FHS?
Antes de empezar a explicar, hay que introducir el termino FHS (Filesystem Hierarchy standard), un estándar a seguir para las distribuciones de Linux, que se refiere a la forma en que se utiliza el sistema de archivos en Linux.

Estructura básica según FHS
Hay doy grupos independientes de archivos, shareables/unshareables y variables/static.

Shareable: Los datos pueden ser usados por múltiples sistemas en una red. Los archivos que se pueden compartir y que son información de propósito general, en donde existe ninguna atadura en ningún host. Un ejemplo pueden ser archivos de datos, de programas ejecutables y documentación del sistema.

Non-Shareable: Aquella información que no se puede compartir, dado que esta atada a un equipo en particular, por ejemplo, archivos de configuración.

Variable: Los datos son considerados variables cuando estos cambian sin intervención del usuario/root. Por ejemplo: Archivos de usuarios, registros del sistema (logs).

Static: Son aquellos archivos que no interactúan con nadie y permanecen siempre iguales durante el día a día o años enteros. Un ejemplo puede ser los binarios de determinados comandos (ls, bash) en donde solo se cambian cuando se actualiza el sistema.

El FHS organiza las categorías de datos anteriores en una matriz de 2 x 2.

	Shareable	Unshareable
static	/usr /opt	/etc /opt
variable	/var/mail /var/cache/fonts /home	/var/run /var/log

Directorios esenciales

El sistema de archivos raíz /
El FHS ofrece un significativo nivel de detalle que describe la localización exacta de los archivos, utilizando la lógica derivada de la definición static/variable y shareable/unshareable. El sistema de archivos raíz se encuentra en la parte superior de la jerarquía de directorios. El FHS define los objetivos para el sistema de archivos raíz:

- Arrancar el sistema, deben estar el software y los datos suficientes en la partición raíz para montar otros sistemas de archivos. Esto incluye herramientas, configuración,

información del cargador de arranque, y otros datos del arranque. Los directorios /usr, /opt y /var están diseñados de modo tal poder ubicarse en otras particiones o sistemas de archivos
- Posibilitar la recuperación y/o reparación de un sistema, aquellas herramientas necesarias para que un administrador experimentado diagnostique y reconstruya un sistema dañado
- Restaurar un sistema, aquellas herramientas necesarias para recuperar backup's
- Las aplicaciones ni la distribución deberían crean archivos o subdirectorios adicionales en el directorio raíz
- Su partición debería ser lo más pequeña posible, siempre que permita realizar las funciones mencionadas anteriormente

Aunque un sistema Linux con todo en una sola partición raíz se puede crear, el hacerlo así no cumpliría con estos objetivos. En cambio, el sistema de archivos raíz debe contener solo los esenciales directorios del sistema, junto con los puntos de montaje para otros sistemas de archivos.

Directorios no esenciales

El resto de directorios de nivel superior en el sistema de archivos raíz se consideran no esenciales para los procedimientos de emergencia:

- /boot
- /home
- /opt
- /tmp
- /usr
- /var

El sistema de archivos /usr

Contiene utilidades de sistema y programas que no pueden no estar en la partición raíz. Por ejemplo, los programas de usuarios, tales como **less** y la **tail** se encuentran en /usr/bin. El directorio /usr/bin contiene comandos de administración del sistema, tales como **adduser** y **traceroute**, y una serie de demonios solo necesarios cuando el sistema funciona normalmente. No hay datos específicos del host o datos variables que se almacenen en /usr. También es rechazada la colocación de directorios directamente en /usr para los paquetes de software grandes. Una excepción a esta regla está hecha para X11.
Los siguientes subdirectorios deben encontrarse en /usr:

- /usr/bin
- /usr/include
- /usr/lib
- /usr/local
- /usr/sbin
- /usr/share

El sistema de archivos /var

Contiene datos tales como, el de la cola de impresión y los archivos de registro de bitácoras que varían con el tiempo. Puesto que los datos variables siempre están cambiando y creciendo, /var es por lo general contenida en una partición separada para evitar el llenado de

la partición raíz.
Los siguientes subdirectorios se requieren en /var:

- /var/cache
- /var/lib
- /var/local
- /var/lock
- /var/opt
- /var/run
- /var/spool
- /var/tmp

Binarios

Los directorios **bin** y **sbin**, se llaman así porque poseen archivos ejecutables, los cuales la mayoría están ya compilados como binarios.
En esta tabla, se enumeran los directorios y muestra como se usa cada uno:

	Comandos de usuario	Comandos de administración del sistema
Suministrados por la distribución (sistema de archivos raíz)	/bin	/sbin
Suministrados por la distribución (sistema de archivos /usr)	/usr/bin	/usr/sbin
Suministrados localmente (por ejemplo, compilados a mano), no son esenciales (sistema de archivos /usr/local)	/usr/local/bin	/usr/local/sbin

Herramientas de búsqueda

FHS sirve como una guía para ubicar y encontrar archivos y directorios, sin embargo, eso no nos garantiza encontrar cualquier archivo. Linux proporciona varias herramientas para buscar archivos, las principales son:

- which: Muestra la ruta de los comandos usando la variable entorno PATH
- find: Buscar archivos en un árbol de directorios
- whereis: Muestra la ruta de los comandos junto con las de sus páginas del manual
- locate: Ofrece buscar archivos indexados en una base de datos en lugar del propio sistema de archivos

Comando find
Es una herramienta muy potente que permite buscar entre otras cosas:

- En un determinado directorio en la jerarquía del sistema de archivos
- Archivos por nombre
- Archivos por tipo
- Archivos por marcas de tiempo
- Archivos por permisos
- Archivos de acuerdo a su dueño

Sintaxis:

find [opciones] [path…] [expresión]

Opciones:

- -mount: No va a seguir buscando recursivamente por directorios que fueron montados
- -maxdepth X: Le indico cual es la cantidad máxima de subdirectorios que quiero que recorra
- -perm: Para buscar por permisos (-perm 777)

Buscar por nombre:

```
# find /home/user/scripts/ -name "*.sh"
```

Busca todos los archivos .sh en el directorio mencionado.
Es conveniente encerrar entre comillas para que la Shell no reemplace automáticamente los meta caracteres en el directorio actual.

Buscar aquellos archivos que se hayan creado hace mas de siete días, y máximo un subdirectorio de profundidad:

```
# find /tmp -maxdepth 1 -type f -daystart -ctime +7
```

Buscar archivos .tmp sin importar mayúsculas o minúsculas y ejecutar una acción sobre los archivos encontrados:

```
# find /home -iname "*.tmp" -exec rm -i '{}' \;
```

La cadena de texto {} es reemplazada por el archivo procesado por find, y el \; significa que no hay más argumentos para el comando después del exec (en este caso rm). Tanto las comillas simples como la contra barra se ponen para que bash interprete de manera literal esos caracteres.

Buscar por tamaño:

```
# find /usr -size + 110M
```

Buscará los archivos con un tamaño mayor a 110 MB

Más ejemplos:

Buscar por nombre de archivo	find /etc -name "*.sh"	Usar comillas al usar metacaracteres para evitar que la shell realice autocompletado de rutas
Buscar por tipo y nombre de archivo	find / -type d -name images	
Buscar por fecha de modificación	find /var -mtime -2	Buscará todos los archivos que fueron modificados hace menos de 2x24hs exactamente
	find /var -mtime +2	Buscará todos los archivos que fueron modificados hace más de 2x24hs exactamente
	find /var -mtime 2	Buscará todos los archivos que fueron modificados hace 2x24hs exactamente

Comando locate
Sintaxis:

locate patron.

La búsqueda se realiza en un índice, creado por el comando **updatedb**, busca los archivos cuyos nombres coincidan con uno o más patrones.

Buscar el patron sh:

```
# locate zyxel
/usr/share/Wireshark/radius/dictionary.zyxel
```

El comando **locate** debe tener una base de datos reciente para la búsqueda, y la base de datos debe ser actualizada periódicamente para incorporar los cambios en el sistema de archivos. Si la base de datos esta demasiado desactualizada, el comando puede advertirnos como se muestra a continuación:

```
# locate tcsh
locate: warning: database /var/lib/slocate/slocate.db' is more than 8 days old
```

Para actualizarla hay que ejecutar el comando **updatedb**

Updatedb
Sintaxis:

updatedb [opciones]

Actualizar (o crear) la base de datos slocate en /var/lib/slocate/slocate.db

Type
Sintaxis:

type [opciones] nombre del archivo

Type no es realmente un programa independiente, sino una parte integrada de la Shell bash. Type le dira como un nombre de archivo se interpretaría si se usa como un nombre de comando. Ejemplo:

```
# type ls
ls is aliased to 'ls –color=tty'

# type grep
grep is /bin/grep
```

Whereis
Sintaxis:

Whereis [opciones] nombre del archivo

Whereis localiza source/binarios y manuales para los archivos especificados. Ejemplo:

```
# whereis ls
ls: /bin/ls /usr/share/man/man1p/ls.1p.gz /usr/share/man/man1/ls.1.gz
```

Which
Sintaxis:

which comando

Determina la ubicación del comando y muestra la ruta completa del programa ejecutable que la Shell lanzaría para ejecutarlo. Este solo busca las rutas definidas en la variable PATH.

```
# which bash
/bin/bash
```

Editores de texto

La edición de archivos de texto es fundamental tanto para administrar Linux como para el desarrollo de aplicaciones en distintos lenguajes. En Linux contamos con una variedad de editores, algunos de ellos son:

- **Emacs**: Es un programa muy adaptable que puede realizar muchas más tardeas además de editar texto
- **Nano**: Surgió como un reemplazo libre del editor de texto Pico
- **Vi y VIM**

Editores Vi y Vim

Vi (visual) es programa informático que entra en la categoría de los editores de texto. Esto se debe a que, a diferencia de un procesador de texto, no ofrece herramientas para determinar visualmente como quedara el documento impreso. Es por esto que carece de opciones como centrado o justificación de párrafos, pero permite mover, copiar, eliminar o insertar caracteres con mucha versatilidad.
Este tipo de programas es frecuentemente utilizado por programadores para escribir código fuente de software.
Vi fue originalmente escrito por Bill Joy en 1976, tomando recursos de ed y ex, dos editores de texto para Unix, que trataban de crear y editar archivos, de ahí, la creación de vi.
Hay una versión mejorada que se llama Vim, pero Vi es un editor de texto que se encuentra en (casi) todo sistema de tipo Unix, de forma que conocer rudimentos de Vi es un salvaguarda ante operaciones de emergencia en diversos sistemas operativos.
Vim (del inglés Vi IMproved) es una versión mejorada del editor de texto Vi.
Su autor, Bram Moolenaar, presento la primera versión en 1991, fecha desde la que ha experimentado muchas mejoras. La principal característica tanto de Vim como de Vi consiste en que disponen de modos entre los que se alterna para realizar ciertas operaciones, lo que los diferencia de la mayoría de editores comunes, que tienen un solo modo en que se introducen las ordenes mediante combinaciones de teclas o interfaces gráficas.

¿Por qué Vi y Vim?

La importancia de este editor de texto se basa en que lo vamos a encontrar en todos los sistemas Unix, por lo cual se transforma en una herramienta fundamental del administrador. Este editor de texto posee características que nos van a permitir mucha versatilidad a la hora de editar rápidamente archivos de texto; por eso, aprendiendo y practicando todas las funciones, podemos hacer de Vim nuestro editor favorito.

10 diferencias entre Vi y Vim

Vim proporciona muchas funcionalidades de las que Vi en muchos casos carece, las más importantes son:

- Múltiples niveles para deshacer cambios
- Pestañas, múltiples ventanas y buffers
- Se puede usar las flechas para desplazarse por el documento

- Resultado de sintaxis
- Modo visual
- Operaciones en bloques
- Sistema de ayuda integrado
- Modo diff
- Se pueden formatear grandes porciones de texto usando un par de teclas
- Plugins

Modos de uso de Vi y Vim

Vi es un editor con diferentes modos. En el modo de edición, el texto que ingresemos será agregado al texto; en modo de comandos las teclas que oprimamos pueden representar algún comando de vi. Cuando comencemos a editar un texto, estará en modo para dar comandos. Por ejemplo, el comando para salir es : seguido de q y ENTER (:q); con ese comando saldrá si no se han hecho cambios al archivo o los cambios ya están salvados; para salir ignorando cambios, presionamos :q! seguido de ENTER.

Modo Comandos: Siempre que se inicia vi, comienza en modo de comandos, que, como su nombre lo indica, permite indicar comandos que ejecuten una acción específica, como buscar, copiar, pegar, eliminar líneas, mover el cursor, posicionarse en partes del documento, etc. Varios comandos están disponibles directamente, con solo apretar una o dos teclas, y otros están disponibles en modo "last line" o última línea, que se accede presionando la tecla dos puntos ":", y, seguido a esto, se indica la acción o comando a ejecutar. Para salir del modo de última línea presionamos ESC.

Inserción: Este modo se utiliza para trabajar en el documento, donde se puede escribirlo y editarlo. Para ingresar al modo inserción se una generalmente la letra "i". Estando en el modo de inserción, presionamos ESC para regresar al modo comandos.

Modo Visual (solo Vim): Pulsando la "v" y utilizando las flechitas para moverse por el documento e ir seleccionando texto, para luego ser copiado, borrado, tabulado, etc.

Editando un archivo

Para pasar del modo inserción al modo de comandos, se emplea la tecla **ESC**; para desplazarse en este modo sobre el archivo, se emplean las teclas **j** (abajo), **k** (arriba), **h** (izquierda) y **l** (derecha). También se pueden usar las flechas, si nuestra terminal lo permite. Avanzar pagina **Ctrl+U** (PgUp), retroceder página **Crtl+D** (PgDn).
Para ir a una línea específica, podemos escribir el numero de la línea, seguido de **gg** o **G**; por ejemplo **25G**, o utilizar "**:**" seguido del número de línea (**:25**) y **ENTER**.
Para mostrar el número de las líneas, se puede ejecutar **:set number**, y para quitar los números **:set nonumber**.
Para ir al final de la línea en la que esta el cursor, presionamos **$**; para ir al comienzo **0**.
Para llegar al inicio del archivo, **1G** o **gg**, y, para llegar al final del archivo, **G**.
Para buscar un texto: **/texto-a-buscar**, y **ENTER**. Luego podemos presionar **n** o **N** para el siguiente o anterior resultado de la búsqueda. Para guardar los cambios **:w**; para guardar y salir, **:wq** o **:x** o **ZZ**.
Para ejecutar un comando del interprete de comandos utilizamos **:!** Seguido del comando y **ENTER**. Podemos teclear **:set all** para ver las opciones disponibles.
Una de las utilidades mas comunes es el uso de **:wq**, que corresponde a la unión de las opciones guardar (w) y salir (q); o bien el modo forzado es **:q!**, que sale de vi sin guardar cambios.

Comandos de desplazamiento

Comando	Descripción
h	Una posición a la izquierda (flecha izquierda)
l	Una posición a la derecha (flecha derecha)
k	Una línea hacía arriba (flecha arriba)
j	Una línea hacía abajo (flecha abajo)
0	(cero) Inicio de la línea
$	Fin de la línea
w	Adelante una palabra
W	Adelante una palabra incluyendo puntuación
b	Atrás una palabra
B	Atrás una palabra incluyendo puntuación
e	Al final de la palabra actual
E	Al final de la palabra actual incluyendo puntuación
n-	Arriba n líneas, primer caracter no espacio
n+	Abajo n líneas, primer caracter no espacio
H	Primera línea de la pantalla actual
M	Línea a mitad de la pantalla actual
L	Última línea de la pantalla actual

Copiar, cortar y pegar

Comando	Descripción
yy	Copia la línea actual
Y	Copia la línea actual
nyy	Copia n líneas
yw	Copia la palabra actual
dd	Corta la línea actual (también se usa para borrar)
p	Pega después del cursor
P	Pega antes del cursor

Mas comandos para realizar inserciones

Comando	Descripción
x	Borra el carácter en el que está el cursor
X	Borra el carácter antes del cursor
nx	Borra n caracteres
dd	Borra la línea actual
ndd	Borra **n** líneas
dw	Borra la palabra actual
ndw	Borra **n** palaras
D	Borra desde el cursor hasta el final de la línea
dL	Borra desde el cursor hasta el final de la pantalla
dG	Borra desde el cursor hasta el final del documento
cw	Reemplaza la palabra actual con nuevo texto
J	Junta la línea actual con la siguiente
~	(tilde) Cambia may/min del carácter actual
u	Deshacer el último cambio de texto
U	Deshacer los cambios en la línea actual
.	(punto) Repite el último cambio de texto realizado
>	Tabula la línea actual hacia la derecha
<	Tabula la línea actual hacia la izquierda

Comandos de inserción y reemplazo

Comando	Descripción
a	Inserta texto después del cursor
A	Inserta texto al final de la línea actual
i	Inserta texto antes del cursor
I	Inserta texto antes del primer carácter no espacio de la línea actual
o	Abre una nueva línea después de la actual
O	Abre una nueva línea antes de la actual
r	Remplaza el carácter actual
R	Remplaza el carácter actual y los siguientes hasta presionar ESC

Explorando un archivo

Cuando estamos en modo de comandos, también podemos utilizar ciertos comandos para poder realizar tareas especiales, como, por ejemplo:
Buscando y reemplazando texto (en modo comandos)

/texto1 – Busca texto1 hacia adelante en el documento
?texto1 – Busca texto 1 hacia atrás en el documento

n – Repite búsqueda a la siguiente ocurrencia
N – Repite búsqueda a la siguiente ocurrencia, invierte dirección
:s/viejo/nuevo – Sustituye la primera ocurrencia de “viejo” a “nuevo” en la línea actual
:s/viejo/nuevo/g – Sustituye todas las ocurrencias de “viejo” a “nuevo” en la línea actual
:%s/viejo/nuevo/g – Sustituye todas las ocurrencias de “viejo” a “nuevo” en todo el documento
:s/viejo/nuevo/g/c – Sustituye todas las ocurrencias de “viejo” a “nuevo” en todo el documento y pregunta por confirmación
& - Repite el ultimo comando de sustitución “s”

Vim avanzado
Mas comandos en modo Ex:

:wq nuevoarch – Termina y guarda documento con el nombre “nuevoarch”
:x – Termina y guarda documento
:h – Palabra busca ayuda sobre la palabra definida
:e archivo2 – Abre “archivo2” mientras se está en un archivo
:n – Avanza al siguiente archivo abierto
:p – Avanza al archivo abierto previo
:r otroarchivo – Inserta, desde la posición actual del cursos, “otroarchivo”
:r !cmdlinux – Inserta, desde la posición actual del cursor, la salida del comando de :!cmdlinux, se ejecuta el comando de Linux indicado
:w – Guarda el documento actual

Opciones de la línea de comandos

vim -r - muestra archivos rescatados después de un cierre incorrecto
vim -r archivo – Recupera “archivo”
vi arch1 arch2 – Abre archivos “arch1” y “arch2”
vi +45 archivo – Abre el “archivo” y posiciona el cursor en el renglón 45
vi +/cadena archivo – Abre el “archivo” y posiciona el cursor en el primer renglón donde encuentre “cadena”

Comparar archivos
Se pueden comparar dos archivos usando vim de la siguiente manera:

vim -d archivo1.txt archivo2.txt

En cuyo caso se mostrarán en colores las diferencias entre ambos archivos.

Dividir en ventanas
Dividir ventanas (no tiene límites), se puede copiar y pegar de una ventana a la otra con los comandos **yy** y **p**

:split – Divide el documento en dos ventanas horizontales con el mismo archivo y las mismas líneas en cada ventana
:vsplit – Divide el documento en dos ventanas verticales con el mismo archivo y las mismas líneas en cada ventana
:split archivo2 – Divide el documento en dos ventanas horizontales con el archivo inicialmente cargado en una ventana y el “archivo2” en la otra
:8split archivo2 – Divide el documento en dos ventanas horizontales con el archivo inicialmente cargado en una ventana y el “archivo2” en la otra con una altura de 8 líneas

:e nombredearchivo – Se carga un archivo en la ventana actual
Ctrl + W (dos veces) – Cambia el foco entre ventanas
:close – Cierra la ventana actual
:only – Cierra todas las ventadas, dejando solo la actual

El intérprete de comandos

Empezando a usar la Shell
La Shell es una interfaz de línea de comandos. Cuenta con un indicador para ingresar comandos llamados prompt, estos son interpretados por la Shell y enviados al sistema. El Shell configura su prompt correspondiente. La mayoría de los sistemas Linux tienen como predeterminado el Shell bash, y generalmente esta configurado para mostrar el nombre de usuario, nombre del servidor y directorio actual de trabajo en el prompt. Por ejemplo:

nombreDeUsuario@debian /home: $

nombreDeUsuario es el nombre del usuario, debian es el nombre del servidor y /home es el directorio actual de trabajo (llamado pwd – present working directory). El prompt para el Shell de bash es el símbolo $ para un usuario común y # para el usuario root. Al prompt lo configuramos a través del archivo /etc/bashrc o el archivo /etc/bash.bashrc (este ultimo presente en distribuciones basadas en debian). Por medio de este archivo, podemos cambiar la configuración de lo que se muestra en el prompt. Cada usuario puede personalizar el prompt a su gusto, editando el archivo ~/.bashrc o el archivo ~/.profile (para debian).

Arquitectura y sistema operativo
Es importante saber que tipo de hardware y software estamos utilizando antes de entender y monitorear procesos. El comando **uname** permite obtener esta información. Así que, luego de loguearnos, ejecutaremos esta vez un comando tan básico como esencial:

```
# uname -s -r -o
Linux 4.9.5-200.fc25.x86_64 GNU/Linux
```

Proporciona respectivamente:

- El nombre del kernel
- La version del kernel
- El nombre del Sistema operativo

Para obtener información del hardware:

```
# uname -m -p -i
x86_64 x86_64 GNU/Linux
```

- Tipo de maquina
- Tipo de procesador
- Tipo de Plataforma del hardware

Tipos de Shell
Existen varios tipos de Shell, las cuales difieren ligeramente (en general) de sintaxis, funcionalidades y configuración. La mayoría de las distribuciones de Linux usan de manera predeterminada el Shell bash
Para ver cuales tenemos instaladas, podemos chequear el siguiente archivo:

```
$ cat /etc/shells
/bin/sh
/bin/bash
/sbin/nologin
/bin/dash
/bin/tcsh
/bin/csh
/usr/bin/irssi
```

Comandos y sintaxis

Los comandos que escribimos en la línea de comandos no son enviados al sistema hasta que no se haya pulsado la tecla Enter. Esto permite editar los comandos antes de que se pasen al sistema. Los comandos escritos en la línea de comandos deben seguir una sintaxis especifica. La primera palabra es el comando que se debe ejecutar; esta puede incluir una ruta absoluta, que es la ruta completa al comando, que debe terminar con el nombre del comando. A continuación, deben figurar las opciones que son usadas por el comando llamadas parámetros. Los argumentos del comando van a continuación de las opciones. Cada uno de estos elementos los separaremos en la línea de comandos por medio de un espacio en blanco. El formato de un comando es el siguiente:

```
$ comando opciones argumentos
```

Las opciones son códigos de una letra precedidos por un guion (-), y modifican la acción del comando. Podemos combinar y usar varias opciones con un mismo comando. Las opciones son sensibles a mayúsculas, y, en muchas ocasiones, una letra minúscula significara una opción diferente que su correspondiente mayúscula. Una opción muy importante con muchos comandos es la **-R**, que significa que el comando se debe ejecutar de forma recursiva a través del árbol de directorios (recursivo significa listar el contenido de los subdirectorios contenidos en el directorio del cual queremos ver el contenido).

Algunos ejemplos:
Si lo que queremos es listar los archivos (ls) y los directorios, y que los directorios aparezcan seguidos de una barra inclinada (/), se puede usar la opción **-F**

```
$ ls -F
```

Vamos a suponer que deseamos mostrar el contenido del directorio /etc desde otra localización. Además, queremos que se muestre una barra inclinada tras todos los nombres de directorios que estén dentro de /etc. El directorio /etc se puede utilizar como argumento para el comando ls. Un argumento es otra opción que se puede utilizar con un comando. En el siguiente ejemplo, el argumento es un directorio que será examinado por el comando. Podríamos probar ejecutar el comando de la siguiente manera:

```
$ ls -F /etc
```

Otro carácter especial que podemos utilizar cuando introducimos comandos es la barra invertida (\). Cuando se introduce a la derecha, antes de pulsar la tecla Enter, la barra invertida permite extender el comando a lo largo de varias líneas. La barra invertida provoca que el sistema ignore la pulsación de la tecla Enter, y trata todos los comandos como si estuvieran en una única línea. Esta función es muy útil si tecleamos comandos muy largos, para poder partirlos en varias líneas. Por ejemplo:

$ ls -F /etc \ ls -F /usr

Para ejecutar comandos muy largos podemos utilizar scripts, en lugar de teclearlos directamente sobre la línea de comandos.
Otras opciones que se pueden utilizar con el comando ls pueden ser:

$ ls -l

Devuelve la lista de archivos y directorios en formato extenso, incluyendo nombre, privilegios de acceso, propietario, tamaño, fecha de ultima modificación, etc.

$ ls -a

Muestra todos los archivos y directorios del pwd, incluyendo los archivos ocultos.
También pueden combinarse:

$ ls -la

Concatenación de comandos
En Linux, no estamos limitados a ejecutar un único comando a la vez.
Podemos ejecutar varios comandos, sin que están conectados entre sí de ningún modo, tecleándolos en la misma línea y separándolos por punto y coma "**;**". Por ejemplo, es posible listar todos los archivos del directorio actual y la fecha de hoy tecleando:

$ ls ; date

El punto y coma es un carácter especial que siempre significa que hay varios comandos en la misma línea. Debido a que este ultimo tiene un sentido global, podemos prescindir de los espacios en blanco a ambos lados del punto y coma para obtener el mismo resultado (ls;date). Lo importante a destacar es que un comando termina independientemente de como finaliza y se ejecuta a continuación el otro.
También podemos ejecutar varios comandos a la vez, pero con los operadores lógicos **&&**, **||**. En este caso, solo ejecuta el comando si cumple con la función lógica dado que, cuando un comando se ejecuta, devuelve un estado 0 (bien) u otro valor mayor que 0 que se interpreta como error.

$ ls /pepe && date
ls: cannot access /pepe: No such file or directory

Como ven ahí no muestra la fecha porque fallo el primer comando.

$ ls -ld /etc && date
Drwxt-xr-x. 171 root root 12299 Oct 27 14:49 /etc
Thu Oct 27 16:06:38 ART 2020

Podemos ver que, al funcionar el primer comando, posteriormente se ejecuta el otro.

Variables
Son parámetros con nombre que contienen valores. Se pueden crear, modificar, referenciar y hacerlas parte del entorno de ejecución.
Por ejemplo, la personalización del prompt bash se puede realizar mediante la modificación de la variable **$PS1**
Ejemplo:

$ PS1="[soy \u en \h dir actual \W]\$"
[soy nombreDeUsuario en debian dir actual etc]$

Secuencias de escape con "\" y sus significados

\u – Nombre de usuario
\h – Nombre del host
\W – Directorio de trabajo
\A – Hora actual

Comando History
Este comando se utiliza para ver los comandos ejecutados con anterioridad en el Shell, por ejemplo:

$ history
1 ls -ltr
2 cd /tmp/
3 ltr
4 cat onlycomm
5 vi onlycomm

Los archivos ejecutados antes se guardan además en un archivo, comenzando por los más antiguos hasta los más recientes.
La variable **HISTSIZE** define el número de comandos que recordara la Shell durante la sesión actual; esta variable puede estar definida tanto en /ect/profile como en ~/.profile (~ equivale al directorio home del usuario) y su valor por defecto es de 1000 entradas. El comando history muestra todas las entradas del archivo del historial que se guarda en ~/.bash_history.
La variable **HISTCMD** proporciona el índice dentro del historial de comando que se está ejecutando. La variable **HISTFILE** especifica el nombre del fichero que contendrá el historial (~/.bash_history por defecto). La variable **HISTFILESIZE** especifica el máximo número de líneas que contendrá el archivo especificado en **HISTFILE** y, aunque suele tener el mismo valor que **HISTSIZE**, podría ser diferente ya que esta última se refiere solo al histórico de la sesión actual y no al tamaño total del archivo histórico.

Operadores de expansión del historial de comandos

Operador	Descripción
!!	También conocido como bang-bang, este operador hace referencia al comando más reciente del historial.
!n	Hace referencia al comando número **n** del historial. Puedes utilizar el comando **history** para conocer estos números.
!-n	Hace referencia al comando actual menos **n** en el historial.
!cadena	Hace referencia al comando más reciente que comience por cadena.
!?cadena	Hace referencia al comando más reciente que contenga cadena.
^cadena1^cadena2	Sustitución rápida. Repite el último comando reemplazando la primera aparición de cadena1 por cadena2.

Comando fc
La utilidad **fc** proporciona otra opción para editar los comandos del fichero histórico antes de ejecutarlos. El comando **fc** abre el editor de textos por defecto con el comando especificado y

ahí podemos editarlo y salvarlo antes de ejecutarlo disponiendo de toda la potencia de un editor de textos.
Podemos llamar a **fc** utilizando como parámetro el número del comando que queremos editar o, también, con un rango de comandos para, de esta forma, editarlos y ejecutarlos en conjunto. También podemos especificar el editor de textos a utilizar. Una vez que llamamos al **fc** el archivo de comandos puede ser editado como cualquier otro archivo de texto y los comandos editados se ejecutaran al salir del editor.
La opción **-l** la utilizaremos para mostrar una lista de los valores especificados a continuación, podemos escribir, por ejemplo, **fc -l 50 60** y obtendremos la lista de los comandos del 50 al 60 en el historial.

Comportamiento de bash
Bash usa un conjunto de funciones llamado readline al ingresar comandos. Existen dos modos de operaciones de readline: **vi** y **Emacs**. El nombre Emacs se debe al nombre de un editor que en algún momento fue muy popular en Linux. A continuación, veremos algunas de sus características.
Algunos atajos de teclado que provee este modo son:

Ctrl + t: Intercambia la posición del carácter
Alt + f: Mueve el cursor a la siguiente palabra
Alt + b: Mueve el cursor a la anterior palabra
Ctrl + u: Borra la línea actual

Completando comandos
El Shell bash incluye una característica denominada “completado de comandos”. Esto nos permite teclear las primeras letras de un comando, pulsar la tecla tabulador, y dejar que el sistema complete el comando por nosotros. Si queremos ejecutar el comando dmesg para mostrar el buffer del kernel, podríamos teclear:

```
$ dm
```

Y pulsar la tecla Tabulador, de forma que el sistema completara el comando:

```
$ dmesg
```

Si existe mas de una coincidencia con la cadena tecleada antes de pulsar la tecla Tabulador, el sistema hará sonar un beep. Pulsando de nuevo la tecla Tabulador se mostrarán todas las posibilidades que coincidan con lo tecleado. Pulsar la tecla **Esc** dos veces produce el mismo efecto que pulsar la tecla Tabulador.

Búsqueda incremental inversa
Existen muchos atajos de teclado que posee bash en modo Emacs. Uno de ellos es la búsqueda incremental inversa. Es muy útil para repetir comandos complejos y/o largos que fueron ejecutados con anterioridad. Por ejemplo, si ejecutamos con anterioridad el comando:

```
$ vi .local/share/kwin/scripts/Shimmer/contents/code/main.js
```

En lugar de tipearlo de vuelta, usamos el atajo del teclado **Ctrl + r** y al tipear **scri** aparecerá algo así:

(reverse-i-search) ‘scri’: vi .local/share/kwin/scripts/Shimmer/contents/code/main.js

Se llama incremental porque de haber mas coincidencias se puede seguir buscando presionando nuevamente el atajo.

Aliases de comandos

Aunque el sistema operativo y el Shell nos ofrecen multitud de comandos y utilidades, podemos crear alias con nombres que tengan mas sentido para nosotros o que sean mas pequeños y así teclear menos caracteres. Por ejemplo:

```
$ alias dir="ls -l"
```

Esto ocasionara que el comando dir ejecute el comando ls -l.
La sintaxis será siempre el alias seguido del comando que queremos que se ejecute cuando se teclee el alias separado por el signo igual.
Con el comando **alias** listamos todos los que están activados.

```
$ alias
```

Con el comando **unalias** comando, sacamos un alias que tenemos programado.

```
$ unalias dir
```

Generación de nombres de rutas de archivos

Los comodines son caracteres que vamos a utilizar en lugar de otros caracteres que el sistema rellena. Los dos comodines mas frecuentes son el asterisco y la interrogación. Aunque en ocasiones se confundan, su significado es diferente y producirán resultados totalmente distintos. Pueden ver un resumen ejecutando **man 7 glob**

```
$ ls s*
```

Este comando mostrar todas las entradas (archivos o directorios) dentro del directorio actual que comiencen con la letra s, y que tengan cualquier numero de caracteres a continuación (incluyendo ninguno). Un posible ejemplo puede ser:

s sa sam sample sample.gif

Hay que prestar atención a que el comando encuentra la "s" sola y la "s" seguida de cualquier numero de caracteres a continuación.
En contraste, el signo de interrogación (?) es un contenedor para un y solo un carácter.
Utilizando las mismas posibilidades que antes, el comando:

```
$ ls s?
```

Encontraremos entradas dentro del directorio actual que comiencen por la letra "s" y que únicamente tengan una letra más. El resultado sería:

sa

Si quisiéramos encontrar las entradas que comiencen por s y cuyo nombre tenga 5 caracteres en total, utilizaremos:

```
$ ls s????
```

En resumen, el asterisco significa todos o ninguno, y el interrogante siempre significa uno. Estos dos comodines no son excluyentes, de modo que se pueden combinar según las necesidades. Por ejemplo, para encontrar solo los archivos que tengan la extensión de tres letras dentro del directorio actual utilizaremos:

$ ls *.???

Para complicar un poco más las cosas, también podemos utilizar los corchetes "[]" para especificar posibles valores. Todos los valores posibles deben estar dentro de los corchetes, y el Shell los tratara individualmente:

$ ls [de]*

En este ejemplo, encontraremos todas las entradas que comiencen por "d" o por "e" y que contengan un número ilimitado de caracteres. Para encontrar las entradas de longitud de 3 caracteres que comiencen por "d" o por "e", utilizaremos:

$ ls [de]??

El número de caracteres que podemos incluir dentro de los corchetes es teóricamente ilimitado. Sin embargo, si lo que queremos es encontrar todas las entradas que comiencen por una letra minúscula pero no por un numero u otro carácter, podemos utilizar [abcdefghijklmnopqrstuvwxyz]. Debido a que esto es un rango, una forma mucho más simple de obtener el mismo resultado es poniendo:

$ ls [a-z]*

Los rangos no tienen que ser series completas de números o caracteres, podemos expresar subconjuntos de ellos. Por ejemplo, si queremos buscar entradas que comiencen por alguna letra entre la "d" y la "t", podemos utilizar indistintamente [defghijklmnopqrst] o [d-t]. Si la entrada puede comenzar por esas letras, tanto en mayúsculas como en minúsculas podemos usar [D-Td-t].
Otros ejemplos son:

[A-z] Todas las letras (mayúsculas y minúsculas)
[A-z] Es lo mismo que [A-Z] y [a-z]
[0-9] Todos los números
[!0-9] Cualquier carácter que no sea un numero
[!A-z] Cualquier carácter que no sea una letra.

Librerías y paquetes

ADMINISTRANDO BIBLIOTECAS COMPARTIDAS

¿Qué es una librería?
Es un archivo que contiene uno o más programas llamados funciones.
Cuando un programa es compilado (pasado de código fuente a código maquina) bajo Linux, muchas de las funciones que se necesitan provienen de otras funciones ya existentes como las de manejo de disco, memoria, entre otras, por esta razón no se van a compilar nuevamente, puesto que ya están instaladas. Debido a esto, los programas van a compartir librerías.
Para evitar problemas relacionados con la memoria, estos programas apuntan a las librerías que necesitan, estando así dinámicamente enlazados.
Si un programa detecta que no puede acceder a estas librerías, lo más probable es que falle en su ejecución, teniendo que cumplir primero con este requerimiento para poder seguir su funcionamiento.

Viendo dependencias de librerías
ldd

Muestra las librerías compartidas que requiere el programa. Devuelve el nombre de la librería y donde se supone que debe estar

```
# ldd /usr/bin/nmap
linux-gate.so.1 => (0x00a9600)
libpcre.so.0 => /lib/libpcre.so.0 (0x0045f000)
libpcap.so.0.9.4 => /usr/lib/libpcap.so.0.9.4 (0x00252000)
libssl.so.6 => /lib/libssl.so.6 (0x00966000)
libcrypto.so.6 => /lib/libcrypto.so.6 (0x0074f000)
libdl.so.2 => /lib/libdl.so.2 (0x00458000)
libstdc++.so.6 => /usr/lib/libstdc++.so.6 (0x06e96000)
libm.so.6 => /lib/libm.so.6 (0x00b35000)
…
```

Ubicación de las librerías compartidas

Las librerías suelen encontrarse en los siguientes directorios:

- /lib
- /libx32 (librerías de arquitectura 32 bits)
- /lib64 (librerías de arquitectura amd64 en distribuciones estilo Red Hat)
- /usr/lib
- /usr/lib64 (librerías para arquitectura amd64 en distribuciones estilo Red Hat)
- /usr/lib/x86_64-linux o /usr/libx64 (librerías para arquitectura amd64 para distribuciones basadas en Debian)
- /usr/lib/i386-linux o /usr/libx32 (librerías para arquitectura 32 bits para distribuciones basadas en Debian)

El enlazador dinámico es la herramienta que busca las librerías de acuerdo a las opciones que se usaron para compilar un programa.
Uno de los primeros lugares que buscara el enlazador esta determinado por la variable **LD_LIBRARY_PATH**. Generalmente no suele asignarse algún valor a esta variable, ya que es preferible que utilice las librerías predeterminadas. Solamente cuando se está desarrollando una aplicación tendría sentido declararla, por ejemplo:

```
# export LD_LIBRARY_PATH=/home/Leo/mylibs
```

Además, puede buscar en directorios adicionales determinado por el archivo /etc/ld.so.conf. Podemos ver el contenido de ese archivo así:

```
# cat /etc/ld.so.conf
include ld.so.conf.d/*.conf
```

El contenido nos informa que leerá los archivos que terminen con .conf en el directorio /etc/ld.so.conf.d, por ejemplo:

```
# cat /etc/ld.so.conf.d/*.conf
/usr/lib64//bind9-export/
/usr/lib64/qt5-qtwebengine-freeworld
```

Administración de librerías compartidas

idconfig

Este comando configura los enlaces en tiempo de ejecución de los distintos programas. Dicha

tarea es realizada por el enlazador dinámico.
Cuando se instala una librería en el sistema es necesario actualizar esos enlaces para que los programas dependientes puedan funcionar bien. Esto se hace sencillamente ejecutando ldconfig.

```
# ldconfig
```

No obstante, esto generalmente no hace falta realizarlo ya que los empaquetadores de las librerías suelen agregar dicha orden en el script de instalación.
En el siguiente ejemplo el comando muestra las librerías de la cache.

```
# ldconfig -p
1054 libs found in cache `/etc/ld.so.cache'
lib3dkit.so.1 (libc6) => /usr/lib/lib3dkit.so.1
libzvbi.so.0 (libc6) => /usr/lib/libzvbi.so.0
libzvbi-chains.so.0 (libc6) => /usr/lib/libzvbi-chains.so.0
libz.so.1 (libc6) => /usr/lib/libz.so.1
libz.so (libc6) => /usr/lib/libz.so
libx264.so.60 (libc6) => /usr/lib/libx264.so.60
libx11globalcomm.so.1 (libc6) => /usr/lib/libx11globalcomm.so.1
…
```


Programas en distribuciones basadas en Debian
Entre las distribuciones mas conocidas que usan el tipo de paquete DEB, se destacan Debian, Ubuntu y Mint.

Nomenclatura de un paquete
Los nombres de los paquetes siguen la siguiente nomenclatura:

<nombre>_<Nro.Version>-<NúmeroDeRevisión>_<arquitectura>.deb

<nombre> Es el nombre del paquete
<Nro.Version> Es el número de versión establecido por el desarrollador, no sigue un estándar.
<NúmeroDeRevisión> Es el número de revisión que normalmente implica cambios en el archivo de control, los scripts de instalación o desinstalación, o en los archivos de configuración usados en el paquete. Este número lo agrega la distribución.
<arquitectura> Indica para qué arquitectura es el paquete.

Características de un paquete
Los paquetes en sistemas Debian/Ubuntu tienen la extensión ".deb", y son manejados por un programa llamado "**dpkg**". Este es una herramienta para instalar, eliminar y manipular paquetes de Debian/Ubuntu. La herramienta principal de Debian para gestionar paquetes es **apt-get**.
Los paquetes "deb" contienen tres archivos:

Debian-binary – Contiene el número de la versión del paquete .deb
control.tar.gz – Contiene la información de control del paquete en una serie de ficheros de texto, por ejemplo, dependencias del paquete, prioridad, mantenedor, arquitectura, conflictos, versión, md5sum, etc.
data.tar – Contiene todos los archivos que se instalaran, con sus rutas de destino

Instalando y desinstalando paquetes con dpkg
Mantiene información sobre los paquetes en el directorio /var/lib/dpkg
Hay dos archivos que son los más relevantes:

available – Contiene la lista de todos los paquetes disponibles e información sobre los mismos
status – Contiene los atributos de los paquetes como el Status (instalado o marcado para eliminar).
Estos archivos también son modificados por **apt-get** y **dselect** (esta última herramienta prácticamente no se usa).
Sintaxis:

dpkg [opciones] acción paquete

Opciones frecuentes:

--configure – Configura un paquete que esta desempaquetado, pero no configurado
-i (--install) – Instala el paquete. Si se especifica la opción --recursive o -R, el fichero-paquete debe ser un directorio
-l (--list) – Lista todos los paquetes. Si se le agrega -l [patrón] lista el o los paquetes que coinciden con el patrón.
-L paquete – Lista los archivos instalados por el paquete
-s paquete – Obtiene información del paquete, como el estado, versión, dependencias, etc.

Ejemplos:

Instalar el programa bzip2
dpkg -i bzip2_1.0.5-6_i386.deb

Desinstalar el programa quota
dpkg -r quota

Para desinstalar un paquete y borrar incluso sus archivos de configuración
dpkg -P exim

Obtener información mediante dpkg
Opciones frecuentes:

-L paquete – Lista los archivos instalados por el paquete
-s paquete – Obtiene información del paquete, como el estado, versión, dependencias, etc.

Ejemplos:

Listar información del paquete
dpkg -s bzip2

Obtener la versión y estado de un paquete instalado
dpkg -l vim

Verificar la integridad del paquete (si no devuelve nada es porque el paquete devuelve el hash correcto
dpkg -V vim

Ver información de un paquete no instalado
dpkg -I (i latina mayúscula) sed_4.7-1_amd64.deb

El sistema APT

El sistema de gestión de paquetes APT (Advanced Packaging Tool), fue creado por el proyecto Debian y se utiliza para la instalación y eliminación de programas en sistemas GNU/Linux. APT fue rápidamente utilizado para funcionar con paquetes .deb

Sources.list

El archivo /etc/apt/sources.list indica desde donde deberán ser descargados los paquetes. Este archivo contiene una lista con repositorios. La lista de repositorios esta diseñada para gestionar cualquier numero de fuentes y distintas procedencias de paquetes.

```
# cat /etc/apt/sources.list
deb http://deb.debian.org/debian/ buster main contrib non-free
deb-src http://deb.debian.org/debian/ buster main
deb http://security.debian.org/ buster/updates main contrib non-free
deb-src http://security.debian.org/ buster/updates main
deb http://deb.debian.org/debian/ buster-updates main contrib non-free
deb-src http://deb.debian.org/debian/ buster-updates main
```

Descripción:
deb: repositorio con los paquetes compilados
deb-src: repositorio con el código fuente de los paquetes

En la línea se hace mención a los parámetros **búster**, **main**, **contrib** y **non-free**.
Buster es una de las ramas de la distribución, solo buscara paquetes y actualizaciones para dicha versión. Si quisiera tener la posibilidad de poder hacer un upgrade a futuras versiones, hay que cambiar dichos parámetros a stable o al nombre que corresponde al de la nueva versión.

Ramas de la distribución

Es importante recordar que Debian tiene 3 ramas en el desarrollo de la distribución:

Rama	Nombre en código	Características
stable	buster	Indicada para usar en producción
testing	bullseye	Sirve para probar paquetes más nuevos pero aún no aceptados en 'stable'
unstable	sid	Sirve para tener los paquetes más actualizados, pero tiene la desventaja de no estar testeados suficientemente

NOTA: Los nombres en código de las versiones son correspondientes a mayo de 2019.
Se puede verificar el nombre de la última versión estable en pagina https://www.debian.org/releases/.
Los paquetes se dividen en tres secciones, **main** (libres y predeterminado), **contrib** (programas libres, pero con dependencias no libres) y **non-free** (paquetes no libres)

El comando apt-get

apt-get no trabaja directamente con los paquetes .deb como lo hace **dpkg**, sino que utiliza los nombres de los paquetes, por ejemplo, en dpkg utilizo "**dpkg -i bzip2_1.0.5-6_i386.deb**", mientras que con apt-get utilizo "**apt-get install bzip2**". apt-get tiene una base de datos con información que le permite a la herramienta actualizar automáticamente paquetes y sus dependencias, como también instalar nuevos paquetes disponibles.
Sintaxis:

apt-get [opciones][comando][nombre paquete…]

Opciones frecuentes:

-d – Descarga los archivos, pero no los instala
-s – No realiza ninguna acción, simula lo que hubiese ocurrido, pero sin hacer cambios en el sistema
-y – Responde que si, a todas las preguntas que nos realiza la herramienta

Comandos frecuentes:

install – Instala o actualiza uno o más paquetes
remove – Remueve los paquetes seleccionados
update – Sincroniza el listado de paquetes disponibles en los repositorios (configurados en el archivo sources.list)
upgrade – Realiza una actualización de todos los paquetes
dist-upgrade – Realiza la misma acción que upgrade, pero, además, puede llegar a quitar paquetes si las nuevas condiciones de los repositorios lo requieren. Se puede usar para actualizar de una versión estable a una mas nueva del mismo tipo. Para ello hay que modificar el archivo sources.list como indico anteriormente
clean – Borra los paquetes de instalación descargados (/var/cache/apt/archives)

Ejemplos:
Actualizar base de datos contra los repositorios:

```
# apt-get update
```

Instalar programas mc, sin confirmación:

```
# apt-get -y install mc
```

Eliminar el programa mc:

```
# apt-get remove mc
```

Elimina el programa mc y toda su configuración:

```
# apt-get --purge remove mc
```

Simular la operación. En este caso se utiliza la opción --dry-run:

```
# apt-get --dry-run upgrade
```

El comando apt-cache
Esta herramienta se utiliza para consultar la cache local de la base de datos de paquetes de Debian.
Comandos frecuentes:

showpkg – Muestra información acerca del paquete y sus dependencias entre otras cosas
show – Muestra descripción acerca del paquete y paquetes sugeridos
search – Busca un paquete por su nombre o descripción

Ejemplos:
Busca paquete que contenga en su nombre o descripción la palabra midnight

```
# apt-cache search commander
```

Además, podemos buscar para que coincida solamente con el nombre del paquete

```
# apt-get search --names-only music
```

Los comandos apt y apt-file
La herramienta apt es una herramienta mas sencilla equivalente a apt-get y apt-search combinadas.
Ejemplos:

```
# apt search mc
```

```
# apt install mc
```

El comando apt-file no viene instalado de manera predeterminada, pero sirve para buscar paquetes conteniendo un determinado archivo. Por ejemplo:

```
# apt-file update && apt-file search bin/nvim
libnvtt-bin: /usr/bin/nvimgdiff
neovim: /usr/bin/nvim
neovim-qt: /usr/bin/nvim-qt
```

INSTALANDO Y ADMINISTRANDO PAQUETES RPM

Distribuciones que usan RPM
RPM es un acrónimo recursivo que significa Red Hat Package Manager. Este sistema de paquetes, que vinee desde Red Hat, es utilizado por varias distribuciones (Fedora, CentOS, SUSE, etc). Con estos paquetes vamos a poder instalar nuestros programas de manera fácil.

Nomenclatura de un paquete
Un paquete RPM consta de cuatro componentes, al igual que Debian:

Firefox-3.6.9-2.el5

Nombre: Nombre del paquete (ejemplo firefox)
Versión: Cada paquete contiene una versión. La forma que se usa es (major.minor.patchlevel).
Revisión: Se trata de la versión release del paquete, dado que, cuando tienen mismo número de versión, se determina cuál es la más nueva por este campo; ya que, algunas veces, los cambios hechos en los paquetes son menores.
Arquitectura: Muestra cuál es la arquitectura que determina el paquete (i386, i586, i686, ia64, sparc, ppc, noarch)

Ejemplo:

Installed Packages
Name: firefox
Arch: i386
Version: 3.6.9
Release: 2.el5

Paquetes RPM: Instalación manual (Instalar, actualizar y desinstalar paquetes con rpm):
Este comando nos permite instalar, remover, actualizar y verificar paquetes entre otras cosas.
Sintaxis:

rpm opciones

Opciones:

-i – Instala el paquete propiamente dicho
-e – Desinstala el paquete
-U – Hace un //update// del paquete a su versión más nueva
-F – o freshen, actualiza solo aquellos paquetes que están instalados en el sistema, pero en este caso, no va actualizar las dependencias de ese paquete, aunque este mencionado en la línea de comandos.
--force – Fuerza la actualización de los paquetes ya sea viejo por nuevo o viceversa
-h – Marca un string de progreso de la tarea a realizar
--nodeps – No buscar las dependencias del paquete; esto puede generar problemas en la base de dependencias
--test – Hace una simulación de la instalación o de la tarea que fuera; no puede ser utilizado con -h pero si con --v
-v – Muestra más información
--v – Muestra información más ampliado que solamente con -v

Ejemplo de instalación:

```
# rpm -ivh zsh-5.5.1-6.el8_1.2.x86_64.rpm
```

Realizar consultas con rpm
-a Nos devuelve todos los paquetes que tenemos instalados, podemos filtrarlos con **grep** para buscar alguno en particular

```
# rpm -qa |grep grub
grub-0.97-13.5
```

Preguntar por pertenencia de un archivo
Con la opción -f vamos a poder preguntar que paquete esta usando el fichero que demos como argumento.

```
# rpm -qf /etc/passwd
setup-2.5.58-7.el5
```

Solicitar información del paquete

```
# rpm -qi grub-0.97-13.5
Name: grub Relocations: (not relocatable)
Version: 0.97 Vendor: Red Hat, Inc.
Release: 13.5 Build Date: Tue 30 Jun 2009 02:34:35 PM ART
Install Date: Thu 29 Apr 2010 04:37:57 PM ART Build Host: hs20-bc1-2.build.redhat.com
Group: System Environment/Base Source RPM: grub-0.97-13.5.src.rpm
Size: 1057883 License: GPL
Signature: DSA/SHA1, Wed 15 Jul 2009 05:53:04 AM ART, Key ID 5326810137017186
Packager: Red Hat, Inc.
URL: http://www.gnu.org/software/grub/
Summary: GRUB - the Grand Unified Boot Loader.
Description: GRUB (Grand Unified Boot Loader) is an experimental boot loader
capable of booting into most free operating systems - Linux, FreeBSD, NetBSD, GNU Mach, and
others as well as most commercial operating systems.
```

Preguntar por archivos de configuración

```
# rpm -qc at
/etc/at.deny
/etc/pam.d/atd
/etc/rc.d/init.d/atd
```

Preguntar por documentación

```
# rpm -qd at
/usr/share/doc/at-3.1.8/ChangeLog
/usr/share/doc/at-3.1.8/Copyright
/usr/share/doc/at-3.1.8/Problems
/usr/share/doc/at-3.1.8/README
/usr/share/doc/at-3.1.8/timespec
….
```

Consultar los archivos pertenecientes a un RPM

```
# rpm -ql at
```

Preguntar por dependencias
-R Muestra los paquetes que este necesita (dependencias):

```
# rpm -qR at
```

Preguntar por archivos de un paquete

```
# rpm -qlp shuutter-0.85.1-1.fc10.noarch.rpm
```

Verificación de paquetes
Una opción muy importante es la de verificar los paquetes para poder saber que fue cambiando. Se realiza con la opción **-V** (o **--verify**)
Opciones que se pueden usar con -V:

--nofiles – Ignora archivos perdidos
--nomd5 – Ignora los chequeos de error de md5
--nopgp – Ignora los chequeos de error de pgp

```
# rpm -V postfix
.......T. c /etc/pam.d/smtp.postfix
.......T. c /etc/postfix/access
.......T. c /etc/postfix/canonical
.......T. c /etc/postfix/generic
S.5....T. c /etc/postfix/header_checks
S.5....T. c /etc/postfix/main.cf
S.5....T. c /etc/postfix/master.cf
.......T. c /etc/postfix/relocated
S.5....T. c /etc/postfix/transport
S.5....T. c /etc/postfix/virtual
.......T. c /etc/sasl2/smtpd.conf
```

Si nos da nada de resultado significa que los archivos del paquete están intactos. En cambio, si aparece alguna de las letras como el ejemplo de arriba significa que algún archivo fue modificado. En el caso de que falte algún archivo también nos informara.

Caracter	Significado
S	Difiere el tamaño del archivo
M	Difiere el modo de acceso y/o el tipo de archivo
D	No coincide el número principal y/o seundario del dispositivo
L	No coincide con la ruta del enlace
U	Difiere el usuario dueño
G	Difiere el grupo dueño
T	Difiere la fecha de modificación
P	Difiere las capacidades del archivo
c	Archivo de configuración
d	Archivo de documentación
g	Archivo que pertenece al paquete al aunque no se instala
l	Archivo de licencia
r	Archivo de readme

Realizar tareas de mantenimiento de rpm

Si apareciera un error con la base de datos se puede regenerar con:

```
# rpm --rebuilddb
```

También se puede generar una base de datos nueva, aunque no es lo aconsejado y rar vez se necesita:

```
# rpm --initdb
```

Extracción de archivos de un paquete RPM

Aquí veremos como extraer el contenido del archivo rpm con la herramienta **rpm2cpio**:

```
# rpm2cpio nxcliente-3.4.0-7.i386.rmp |cpio -t
/etc/profile.d/nx.csh
./etc/profile.d/nx.sh
./usr/NX
./usr/NX/bin
./usr/NX/bin/nxclient
……
```

Ahora extraigo lo que necesito:

```
# rpm2cpio nxcliente-3.4.0-7.i386.rpm |cpio -ivmd ./usr/NX/share/images/wizard.png
./usr/NX/share/images/wizard.png
19913 blocks
```

Yum

El gestor de pagues YUM (YellowDog Updater Modified) nos ofrece una manera rápida de instalar los paquetes. Podemos actualizar, instalar y remover paquetes. Tiene funciones muy similares a las de rpm, pero con la particularidad que puede administrar toda la resolución e instalación de dependencias de paquetes.
Además, YUM, nos permite cargar múltiples repositorios de paquetes de manera muy sencilla.

Configuración de YUM

Se realiza mediante el archivo /etc/yum.conf
A continuación, un ejemplo:

Archivos de configuración:
Configuración /etc/yum.conf

[main]
cachedir=/var/cache/yum/$basearch/$releasever
keepcache=0
debuglevel=2
logfile=/var/log/yum.log
exactarch=1
obsoletes=1
gpgcheck=1
plugins=1
installonly_limit=5
bugtracker_url=http://bugs.centos.org/set_project.php?project_id=19&ref=http://bugs.centos.org/bug_report_page.php?category=yum
distroverpkg=centos-release

cachedir		Directorio de caché	Contiene una ruta
keepcache		¿Guardar los archivos y encabezados luego de una instalación exitosa?	0 es no, 1 es sí
debuglevel		El nivel de detalle en los mensajes	El rango es de 0 (mínimo) a 10 (máximo)
logfile		Archivo de log	Contiene una ruta
exactarch		¿Actualizar solamente paquetes de la arquitectura actual?	0 es no, 1 es sí
obsoletes		Incluye a los paquetes obsoletos en los cálculos al actualizar	0 es no, 1 es sí
gpgcheck		¿Verificar firma GPG?	0 es no, 1 es sí
plugins		¿Habilitar plugins de yum?	0 es no, 1 es sí
installonly_limit			

Repositorios

Estos archivos están en el directorio /etc/yum.repos.d

```
# ls
adobe-linux-i386.repo  rhel-debuginfo.repo
convirt.repo           rpmforge.repo
dsmerror.log           rpmforge-testing.repo
dsmwebcl.log           rpmfusion-free-updates.repo
epel.repo              rpmfusion-free-updates-testing.repo
epel-testing.repo      rpmfusion-nonfree-updates.repo
google.repo            rpmfusion-nonfree-updates-testing.repo
mirrors-rpmforge       yum.conf
openfusion.repo.bkp
```

Instalar, actualizar y desinstalar paquetes con YUM

Instalar un paquete con yum:
En el siguiente ejemplo se instala el paquete postgresql

```
# yum install postgresql
```

De manera predeterminada, yum pedirá confirmación de la instalación, se puede evitar con el parámetro **-y**

```
# yum -y install paquete
```

Desinstalar un paquete con yum
Para desinstalar un paquete incluyendo todas sus dependencias:

```
# yum remove postgresql
```

Actualizar un paquete
Si tenemos una versión vieja de un paquete, podemos utilizar "**yum update paquete**" para actualizarlo a la ultima versiona. Esto también busca e identifica todas las dependencias requeridas:

```
# yum update postgresql
```

Actualizar el sistema entero
Si no especificamos el paquete, yum actualizara todos los paquetes:

```
# yum update
```

Realizar consultas con YUM

Ver el listado de actualizaciones disponibles

```
# yum check-update
```

Buscar un paquete con YUM
Si no sabemos exactamente el nombre del paquete a instalar, podemos usar el comando "**yum search palabra**", que realizara una búsqueda de todos los paquetes que coincidan con la palabra indicada.
La búsqueda que realicemos devolverá como resultado todos los paquetes que coincidan con la palabra "firefox":

```
# yum search firefox
```

Solo buscar en los campos nombre y resumen; si quisiéramos que busque por todos los campos tendríamos que usar la opción "**search all**"

Mostrar información adicional acerca de un paquete
Una vez que buscamos y poseemos el nombre del paquete con "**yum search**", podemos obtener información adicional acerca del paquete utilizando el siguiente comando: "yum info paquete":

```
# yum info samba-common.i686
```

Ver todos los paquetes disponibles
El siguiente comando nos mostrara una lista de todos los paquetes disponibles que hay en nuestra base de datos yum.

```
# yum list | less
```

Listar paquetes instalados
Muestra una lista de todos los paquetes instalados en nuestro sistema.

```
# yum list installed
```

Visualizar a que paquete le pertenece un archivo
Utilizar el comando "**yum provides**" para identificar a que paquete le corresponde determinado archivo.

```
# yum provides /etc/sysconfig/nfs
```

Listar los grupos de programas instalables
En yum, una gran cantidad de paquetes están agrupados. En lugar de tener que buscar paquetes individualmente, con esta opción se instalaran los que correspondan a un grupo específico.
Para poder ver los grupos de paquetes disponibles, usaremos el comando "**yum grouplist**". La salida de este comando mostrara los grupos disponibles y los instalados

```
# yum grouplist
Installed Groups:

Administration Tools
Base
Design Suite
....

Installed Language Groups:
Arabic Support [ar]
Armenian Support [hy]
Bengali Support [bn]
....

Available Groups:
Authoring and Publishing
Books and Guides
Clustering
DNS Name Server
Development Libraries
Development Tools
……
```

Realizar tareas adicionales con yum

Instalar un grupo de programas específicos

Para instalar un grupo especifico de programas, utilizamos la opción "**groupinstall**". Se instalará el grupo "DNS Name Server", el cual trae por dependencia el paquete bind-chroot.

```
# yum groupinstall 'DNS Name Server'
```

Actualizar un grupo instalado

Si ya disponemos de un grupo, podemos realizar una actualización de este con el comando "**yum groupdate**"

```
# yum groupdate 'Graphical Internet'
```

Desinstalar un grupo de programas

Para borrar un grupo de programas instalados, se utiliza el comando "**yum groupremove**".

```
# yum groupremove 'DNS Name Server'
```

Mostrar los repositorios instalados

Para poder ver los repositorios que tenemos configurados en nuestro sistema se usa el comando "**yum repolist**".

El siguiente comando muestra solo los repositorios activos.

```
# yum repolist
```

Para mostrar todos los repositorios:

```
# yum repolist all
```

Limpiar la cache

Yum puede guardar en una cache:

- Paquetes descargados antes de instalarlos
- Encabezados
- Metadatos de paquetes
- Metadatos de la cache sqlite
- Datos de la base local de RPM

Para borrar toda la cache se realiza lo siguiente:

```
# yum clean all
```

Este tipo de operación se suele realizar cuando alguno de los elementos de la cache se ha dañado.

DNF

Es un gestor de paquetes que utiliza las librerías **hawkey** y **libdnf**

En versiones recientes de Fedora dnf ha reemplazado a yum como herramienta predeterminada para administrar paquetes. La sintaxis de dnf y yum son muy similares.

Usa como archivo de configuración /etc/dnf/dnf.conf y usa los mismos archivos de repositorio que yum.

Algunos subcomandos que vienen separados en yum, ya vienen integrados en dnf, por ejemplo 'download':

```
# dnf download mc
```

Se puede encontrar más información en la página del manual yum2dnf.

Zypper
Es el gestor de paquete predeterminado de la distribución SUSE. Está basado en la librería libzypp. Algunas características son:

- Archivos .repo compatibles con YUM
- Trata las dependencias entre los paquetes como un problema de satisfacibilidad booleana
- Usa metalink al igual que yum para mejorar las descargas

Instalar paquetes

```
# zypper install ranger
```

Reinstalar paquetes

```
# zypper install -f ranger
```

Actualizar todos los paquetes

```
# zypper update
```

Actualizar un paquete

```
# zypper update ranger
```

Obtener información de un paquete

```
# zypper info --provides --requires ranger
```

Archivo de configuración de zypper
El archivo principal de zypper es /etc/zypp/zypp.conf, uno de los parámetros más importantes es:

repo.refresh.delay = 10

Lo que significa que cada 10 minutos zypper actualizara la información que posee de los repositorios.

Archivos de repositorios
Los archivos de repositorios están en el directorio /etc/zypp/repos.d, cada uno de ellos puede tener un parámetro autorefresh, si esta puesto en 1, zypper actualizara la información del repositorio de acuerdo a la configuración dada por repo.refresh.delay en el archivo principal. Por ejemplo:

```
[openSUSE-Leap-15.0-1]
name=openSUSE-Leap-15.0-1
enabled=1
autorefresh=1
baseurl=http://download.opensuse.org/distribution/leap/15.0/repo/oss
path=/
type=rpm-md
keeppackages=0
```

Aptitude

Es un gestor de paquetes para sistemas Debian y su utilización es muy similar al **apt**, la diferencia esta en su interfaz en modo texto para el manejo del sistema de paquetes, y, además, utiliza un algoritmo distinto para manejar dependencias, por lo que debe usarse con precaución.
Estos son algunos de los archivos más importantes de aptitude:

/var/lib/aptitude/pkgstates: Se almacenan el conjunto de estados y algunas banderas de los paquetes
/etc/apt/apt.conf, /etc/apt/apt.conf.d/*, ~/.aptitude/config: Son los archivos de configuración de aptitude.
/var/log/aptitude: Logs de aptitude

Tanto apt como aptitude comparten el mismo archivo **sources.list**

Sintaxis:

aptitude [opciones] [comando] [paquetes]

Ejemplos:

aptitude search mc (igual que apt-cache search)

aptitude update

aptitude install mc

Para acceder a la interfaz de texto hay que ejecutar aptitude sin ningún parámetro adicional.

Alien

Es un programa que convierte principalmente paquetes rpm a deb o paquetes deb a rpm. Si se desea utilizar un paquete de una distribución GNU/Linux diferente a la que esta instalada en nuestro sistema, alien, puede convertir el paquete de acuerdo a la distribución que tengamos en nuestro sistema.
Sintaxis:

alien [-to-deb] [-to-rpm] [opciones] paquete

Opciones frecuentes:

-i – Instala automáticamente el paquete generado y elimina el paquete convertido
-r – Convierte un paquete a RPM
-t – Convierte un paquete a un archivo tar gzip

Dselect

Funciona como una interfaz de **dpkg**, la cual ofrece una pantalla completa de selección de paquetes con un solucionador de dependencias y conflictos. Permite instalar, actualizar y eliminar paquetes. Habitualmente, dselect se invoca sin parámetros. Se presenta un menú interactivo que ofrece al usuario una lista de acciones. Si se da una acción como argumento se ejecutará inmediatamente.

El comando yumdownloader

Este agregado sirve para descargar paquetes para uso posterior, junto con sus dependencias. Algunas veces, quizás necesitemos descargar paquetes con sus dependencias, pero no para instalarlos, sino para luego copiarlos e instalarlos en otra PC.

Yumdownloader (que viene con el paquete yum-utils) es muy útil para eso, porque nos sirve para construir grupos de paquetes reutilizables para un grupo de computadoras idénticas y luego distribuirlos.

Descargar paquete fuente

```
# yumdownloader --sources packagename
```

Descargar un paquete y dependencias

```
# yumdownloader --resolve packagename
```

Descargar un paquete y sus dependencias para una diferente instalación

```
# yumdownloader --resolve packagename --root /path/to/chroot
```

Autor: ***David Fernando Suarez Zambrana***