GNU/Linux

Básicamente

Antonio Perpiñan

Fundación Código Libre

http://www.codigolibre.org

Padre Pina #102 • Santo Domingo • 809-476-7758 • Avenida Estrella Sadhalá #5 • Santiago • 809-724-5543
República Dominicana • http://www.codigolibre.org info@codigolibre.org

GNU/Linux

Básicamente

Padre Pina #102 • Santo Domingo • 809-476-7758 • Avenida Estrella Sadhalá #5 • Santiago • 809-724-5543
República Dominicana • http://www.codigolibre.org info@codigolibre.org

Padre Pina #102 • Santo Domingo • 809-476-7758 • Avenida Estrella Sadhalá #5 • Santiago • 809-724-5543
República Dominicana • http://www.codigolibre.org info@codigolibre.org

Introducción

Hola a Todos lo que utilizan Minix, estoy creando un sistema operactivo gratuito es solo un hobby, no se trata de nada enorme ni profesional como GNU, es para los clones 386, 486, me gustaria recibir todos los comentarios repecto a lo que la gente piensa de minix, por que mi sistema operativo se parece un poco al suyo.
Todas las sugerencias serán bien recibidas aunque no prometo que la vaya a incluir todas.
.....LinusTorvalds@columna.helsinki.fi

Bienvenidos a GNU/Linux

Este libro es producto de una jornada de cursos, talleres y charlas sobre el Software Libre y en particular su sistema operativo y columna vertebral GNU/Linux, ofreciendo una orientación para ayudar a obtener las certificaciones ofrecidas por la nuestra fundación que pronto deberá estar disponible en todos los países que la adopten ya que será la primera verdadera certificación LIBRE.

Este libro contienen todas las informaciones necesarias para prepararle en sus estudios para poder empezar a estudiar para estas certificaciones. GNU/Linux Ejercicios es para ayudarle a prepararse para poder continuar con los siguientes volúmenes diseñados para prepararle para el examen GNU CERTIFIED, y una carrera como Administrador de Sistemas GNU/Linux y un amplio conocimiento de los sistemas basados en Software Libre en general. En este le introduciremos a la historia, los comandos básicos como los son **ls, cp y mv**, el uso del shell **bash**, el sistema de archivos, en fin la parte básica de GNU/Linux que todo aspirante a titulo de SysAdmin debe dominar.

Al completar este libro usted debe tener una iniciación sólida y bien encaminado para introducirle en aprender los quehaceres básicos del administrador de sistemas GNU/Linux.

Al final de la Serie GNU CERTIFIED (4 en total) usted poseerá todo el conocimiento necesario para convertirse en un verdadero Administrador de Sistemas GNU/Linux y ostentar para las certificaciones GNU CERTIFIED ofrecidas por nuestra fundación.

¡Envíenos su Opinión!

Como todo en Software Libre, el lector de este libro, es un colaborador y puede aportar a que este libro mejore y que con sus críticas se desarrolle. Queremos saber que opinas, si te gusto también y si no con más razón, déjanos saber que está mal y lo mejoraremos y si crees que tienes algo que aportar ayúdanos.

Como autor, le damos bienvenidas a sus comentarios. Puedes enviarnos sus emails directamente de que le gusta y que no le gusta de este libro – y su opinión de como mejorarlo. Tome nota que no podemos responder todas las preguntas acerca de aprendizaje de GNU/Linux ya que recibimos un volumen muy alto pero tratamos de responder las más que podemos, pero sus preguntas técnicas deben ser dirigidas al forum en nuestra paginas web de nuestro portal de Software Libre.

http://www.codigolibre.org

Cuando nos contacte favor recordar incluir su nombre y email en el cual les podemos contactar en caso de que tengamos algunas preguntas de seguimiento.

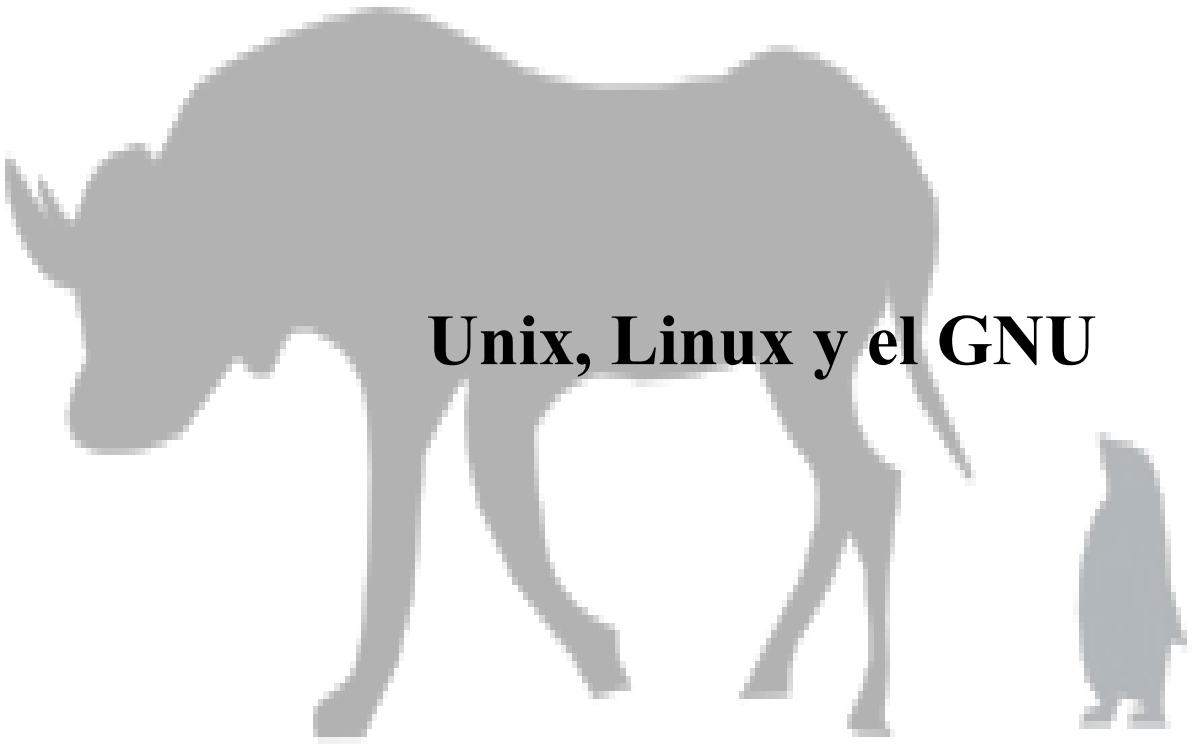
TEL: 809-476-7758
Email: aperpinan@codigolibre.org
Fundación Código Libre Dominicano
Padre Pina #102
Zona Univ. Santo Domingo, República Dominicana

INDICE


FCLD
http://www.codigolibre.org

Capítulo 1

Hazlo simple: tan simple como sea posible, pero no más.
—A. Einstein



Unix, Linux y el GNU

Los Objetivos de este Capítulo son:

1. **Qué es Unix, BSD y los derivados**
2. **Filosofía Unix**
3. **Filosofía GNU**
4. **El Kernel Linux**
5. **El sistema operativo GNU/Linux**
6. **Las características del GNU/Linux**

Unix, Linux y el GNU

¿Qué es UNIX?

UNIX es un sistema operativo multitarea, multiusuario, creado en 1969 por los investigadores Thompson y Ritchie de los Laboratorios Bell, en los Estados Unidos. Las primeras versiones fueron escritas en assambler, pero muy pronto fue re-escrito en lenguaje C. En sus primeros años, no se lo utilizó comercialmente, sino que se usaba para proyectos de investigación en laboratorios y se distribuya gratuitamente en las universidades, donde tuvo mucha aceptación.

En 1975, Bell decidió comercializarlo. Dado que el sistema se vendía con una licencia que permitía modificarlo y redistribuirlo, a lo largo del tiempo fueron surgiendo una gran variedad de sistemas derivados del UNIX original. Los más conocidos, actualmente, son: Solaris, AIX, HPUX, SCO, BSD.

Por esta razón, varias veces se hizo necesario normalizar estos sistemas, para que cumplan con determinadas normas (POSIX, UNIX95, etc.), para permitir la compatibilidad entre los diferentes sistemas. De estas normas, el sistema operativo GNU/Linux satisface la norma POSIX-1, y casi completamente la POSIX-2.

¿Qué es BSD?

La Universidad de Berkeley estuvo relacionada con el desarrollo de los sistemas operativos UNIX. Recibió de AT&T una versión gratuita de UNIX, y a partir de entonces comenzó a promover el desarrollo de aplicaciones para UNIX dentro de la universidad. Más adelante, desarrolló su propio sistema operativo UNIX, sin utilizar el código fuente de AT&T.

El kernel fué creado desde Berkeley, pero las herramientas utilizadas son en su mayoría GNU, es decir las mismas que en el sistema GNU/Linux. Existen actualmente 3 sistemas operativos libres, derivados del BSD: FreeBSD, OpenBSD y NetBSD.

Filosofía Unix

¿Por qué tuvo tanto éxito el enfoque de UNIX? Aparentemente, su simplicidad fue un factor decisivo. En su diseño, sus creadores antepusieron la facilidad de comprensión a la eficiencia, de manera que era fácil entender el código y, por ende, adaptarlo a las necesidades de otros. UNIX no es una reliquia del pasado; de hecho, la mayor parte de los sistemas operativos actuales son una evolución de UNIX. Por eso conviene conocer los principios en los que se fundamenta, puesto que esos mismos principios estarán presentes (de una u otra manera) en los sistemas que hoy podamos manejar.

Todo es un Archivo

Esta idea, propia de la orientación a objetos (si bien la precede), consiste en que la unidad básica para la interacción con el sistema es una entidad llamada archivo que, como los archivos en papel, puede abrirse, leerse, avanzar hojas hacia delante y hacia atrás, escribir en él, y cerrarse. Este modelo tan sencillo puede parecer ingenuo, pero ha probado ser extremadamente valioso. Permite a un programa acceder transparentemente a un documento de texto o a un puerto de comunicaciones.

Multi-Usuario

En un sistema multiusuario, cuando alguien quiere acceder a la máquina, debe identificarse, para poder ser reconocido por ésta y permitirle la entrada al sistema, si se trata de un usuario autorizado. Este proceso es el que se conoce como logging in. Durante este proceso, la máquina nos preguntará nuestro nombre de usuario (login) y nuestra contraseña (password). Es el administrador del sistema (root) quien debe crearnos una cuenta en la máquina y quien nos dará los datos. Una vez entremos a nuestra cuenta, nos encontraremos dentro de nuestro directorio HOME (el directorio que nos asigna el administrador para que guardemos nuestros trabajos), y se nos aparecerá el prompt del sistema: esto es un símbolo que nos indica que la máquina está lista para recibir comandos. Puede ser algo parecido a:

Linux: /home/Usuario#

o bien

Linux: ~#

o cambiando # por $, pero en todos los sistemas aparecen de forma parecida a esta. El nombre anterior a los dos puntos es el nombre de la máquina en la que trabaja. El símbolo ~ en la ruta significa que estamos en nuestro directorio HOME. Hay sistemas UNIX que muestran el directorio completo, y los hay que no, para saber si estamos en nuestro HOME, existe ese símbolo de ~.

Multi-Tarea

La palabra multitarea describe la habilidad de ejecutar varios programas al mismo tiempo. GNU/LINUX utiliza la llamada multitarea preventiva, la cual asegura que todos los programas que se están utilizando en un momento dado serán ejecutados, siendo el sistema operativo el encargado de ceder tiempo de microprocesador a cada programa.

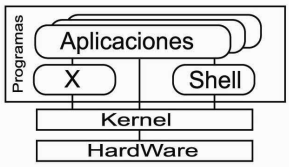
La navaja Suiza

UNIX incorpora un conjunto de herramientas que guardan cierta analogía con una navaja multiusos. Son simples, pero hacen muy bien su trabajo. En lugar de construir programas muy complejos, UNIX proporcionaba muchas pequeñas herramientas, y un esquema para poder combinarlas de forma efectiva. Este diseño escala muy bien, permitiendo al sistema crecer, incorporar nuevas herramientas y, a la vez, ser compatible hacia atrás.

Manual en Línea

Cuando Thompson y Ritchie estaban desarrollando UNIX, solicitaron a sus jefes un computador más potente (DEC PDP-11) a cambio de desarrollar un sistema completo de tipografía (no les dijeron nada acerca de UNIX). Con el nuevo ordenador desarrollaron UNIX sobre C y, Joe F. Ossanna desarrolló troff (de typesetting run-off). Este sistema fue incluido en el propio UNIX, de manera que el manual del sistema fue escrito con él, estando disponible en línea desde entonces (a través del programa man).

Arquitectura de los Sistemas Unix



- El shell y el Xwindow son programas
- Programas solo pueden acceder el hardware vía el kernel

¿Qué es GNU?

La sigla GNU simplemente significan GNU is Not Unix.

En 1984, Richard Stallman fundó el Proyecto GNU con el objetivo de conseguir un sistema operativo libre y abierto. Esto es, un sistema operativo tal que los usuarios puedan usarlo, leer el código fuente, modificarlo, y redistribuirlo. A partir de ese momento, un gran número de colaboradores se fueron sumando al proyecto, desarrollando software libre para reemplazar cada una de las herramientas del sistema UNIX.

La filosofía GNU apoya el crecimiento de la sociedad como un conjunto, haciendo especial hincapié en la valoración de las libertades personales, aún cuando esto puede estar en conflicto con intereses empresariales.

¿Qué es Linux?

En 1991, Linus Torvalds completó el sistema con su kernel (la aplicación encargada de comunicar los procesos con el hardware de la computadora). A este kernel se le bautizó Linux. De esta manera, se formó el sistema GNU/Linux.

Algunas de las características de GNU/Linux son:

- **Multitarea**: La palabra multitarea describe la habilidad de ejecutar varios programas al mismo tiempo. Linux utiliza la llamada multitarea preventiva, la cual asegura que todos los programas que se están utilizando en un momento dado serán ejecutados, siendo el sistema operativo el encargado de ceder tiempo de microprocesador a cada programa.
- **Multiusuario**: Muchos usuarios usando la misma maquina al mismo tiempo.
- **Multiplataforma**: Las plataformas en las que en un principio se puede utilizar Linux son 386-, 486-. Pentium, Pentium Pro, Pentium II, Amiga y Atari, también existen versiones para su utilización en otras plataformas, como Alpha, ARM, MIPS, PowerPC y SPARC.
- **Multiprocesador**: Soporte para sistemas multiprocesador estan disponible para Intel y SPARC.
- **Monolítico**: Se basa en un gran núcleo que se encarga de la gestión y control de todo el sistema. A diferencia de estos, los micronúcleos reparten sus tareas entre varios segmentos de código dedicados a menesteres más particulares, gozándose de gran flexibilidad y versatilidad.
- Funciona en modo protegido 386.
- Protección de la memoria entre procesos, de manera que uno de ellos no pueda colgar el sistema.
- Carga de ejecutables por demanda: Linux sólo lee del disco aquellas partes de un programa que están siendo

usadas actualmente.

- Política de copia en escritura para la compartición de páginas entre ejecutables: esto significa que varios procesos pueden usar la misma zona de memoria para ejecutarse. Cuando alguno intenta escribir en esa memoria, la página (4Kb de memoria) se copia a otro lugar. Esta política de copia en escritura tiene dos beneficios: aumenta la velocidad y reduce el uso de memoria.
- Memoria virtual usando paginación (sin intercambio de procesos completos) a disco: A una partición o un archivo en el sistema de archivos, o ambos, con la posibilidad de añadir más áreas de intercambio sobre la marcha Un total de 16 zonas de intercambio de 128Mb de tamaño máximo pueden ser usadas en un momento dado con un límite teórico de 2Gb para intercambio. Este límite se puede aumentar fácilmente con el cambio de unas cuantas líneas en el código fuente.
- La memoria se gestiona como un recurso unificado para los programas de usuario y para el caché de disco, de tal forma que toda la memoria libre puede ser usada para caché y ésta puede a su vez ser reducida cuando se ejecuten grandes programas.
- Librerías compartidas de carga dinámica (DLL's) y librerías estáticas.
- Se realizan volcados de estado (core dumps) para posibilitar los análisis post-mortem, permitiendo el uso de depuradores sobre los programas no sólo en ejecución sino también tras abortar éstos por cualquier motivo.
- Compatible con POSIX, System V y BSD a nivel fuente.
- Emulación de iBCS2, casi completamente compatible con SCO, SVR3 y SVR4 a nivel binario.
- Todo el código fuente está disponible, incluyendo el núcleo completo y todos los drivers, las herramientas de desarrollo y todos los programas de usuario; además todo ello se puede distribuir libremente. Hay algunos programas comerciales que están siendo ofrecidos para Linux actualmente sin código fuente, pero todo lo que ha sido gratuito sigue siendo gratuito.
- Control de tareas POSIX.
- Pseudo-terminales (pty's).
- Emulación de 387 en el núcleo, de tal forma que los programas no tengan que hacer su propia emulación matemática. Cualquier máquina que ejecute Linux parecerá dotada de coprocesador matemático. Por supuesto, si el ordenador ya tiene una FPU (unidad de coma flotante), esta será usada en lugar de la emulación, pudiendo incluso compilar tu propio kernel sin la emulación matemática y conseguir un pequeño ahorro de memoria.
- Soporte para muchos teclados nacionales o adaptados y es bastante fácil añadir nuevos dinámicamente.
- Consolas virtuales múltiples: varias sesiones de login a través de la consola entre las que se puede cambiar con las combinaciones adecuadas de teclas (totalmente independiente del hardware de video). Se crean dinámicamente y puedes tener hasta 64.
- Soporte para varios sistemas de archivo comunes, incluyendo minix-1, Xenix y todos los sistemas de archivo típicos de System V, y tiene un avanzado sistema de archivos propio con una capacidad de hasta 4 Tb y nombres de archivos de hasta 255 caracteres de longitud.
- Acceso transparente a particiones MS-DOS (o a particiones OS/2 FAT) mediante un sistema de archivos especial: no es necesario ningún comando especial para usar la partición MS-DOS, esta parece un sistema de archivos normal de Unix (excepto por algunas restricciones en los nombres de archivo, permisos, y esas cosas). Las particiones comprimidas de MS-DOS 6 no son accesibles en este momento, y no se espera que lo sean en el futuro. El soporte para VFAT, FAT32 (WNT, Windows 95/98) se encuentra soportado desde la versión 2.0 del núcleo y el NTFS de WNT desde la versión 2.2 (Este ultimo solo en modo lectura).
- Un sistema de archivos especial llamado UMSDOS que permite que Linux sea instalado en un sistema de archivos DOS.
- Soporte en sólo lectura de HPFS-2 del OS/2 2.1
- Sistema de archivos de CD-ROM que lee todos los formatos estándar de CD-ROM.
- TCP/IP, incluyendo ftp, telnet, NFS, etc.
- Appletalk.
- Software cliente y servidor Netware.
- LAN Manager / Windows Native (SMB), software cliente y servidor.

- Diversos protocolos de red incluidos en el kernel: TCP, IPv4, IPv6, X.25, IPX, Netrom, etc.

Qué son las distribuciones

El código fuente del sistema GNU y del kernel Linux está accesible a todo el mundo, sin embargo, hacer funcionar un sistema a partir del código fuente es bastante difícil. Por eso, un sistema operativo se distribuye (normalmente) en formato binario, es decir ya compilado. Poco después de que apareciera el kernel Linux, comenzaron a aparecer las primeras distribuciones, que agrupaban versiones probadas de varios programas, junto con el kernel, de tal manera que formaban un sistema operativo listo para usar.

A medida que fue pasando el tiempo, algunas distribuciones se fueron haciendo más sofisticadas, otras desaparecieron, otras se hicieron comerciales y aparecieron mucha más. Existen distribuciones de muchos tipos: distribuciones que ocupan 1 disquete y distribuciones que llegan a ocupar 10 CDs; distribuciones orientadas a una finalidad en especial (redes, seguridad, etc) y distribuciones de uso general.

Cada usuario de GNU/Linux suele elegir la distribución con la que se siente más cómodo, y no tiene sentido entrar en discusiones acerca de cuál es mejor. A menos que aclaremos lo contrario, lo que se enseña en este curso es aplicable a la gran mayoría de los sistemas UNIX, y a cualquiera de las distribuciones de GNU/Linux.

Software Libre

A lo largo de todo este curso, siempre utilizamos Software Libre. Por lo general, este software lo hemos obtenido gratuitamente, sin embargo, debemos entender que el hecho de que el software sea libre está relacionado con la libertad que nos otorga a los usuarios de utilizarlo, modificarlo y distribuirlo, no con el precio al cual lo podemos obtener.

Al hablar de software libre se suelen clasificar los distintos grados de libertad a los que podemos tener acceso los usuarios.

- **Libertad 0:** El software se puede usar. Es la libertad que nos otorga casi cualquier software.
- **Libertad 1:** El software se puede modificar. Es decir, se puede personalizar, mejorar, adaptar para las necesidades particulares de un determinado usuario.
- **Libertad 2**: El software se puede distribuir. Es decir, se puede copiar, vender, prestar o compartir a las personas que el usuario desee, sin tener que pedir permiso al autor del software.
- **Libertad 3:** El software se puede distribuir modificado. Se trata de una suma de la 1 y la 2. Permite que las mejoras que un usuario le haya hecho a un determinado software puedan compartirse con otros usuarios.

Para poder considerar que una determinada aplicación cumple con los requisitos de software libre es necesario que estén dadas estas cuatro libertades para cualquier usuario. En particular para poder tener la libertad de modificar el software, es necesario tener acceso al código fuente del programa en cuestión, y no solamente al código binario (llamamos código binario a aquél que es entendido por la computadora) ya que para poder modificar correctamente el software es necesario poder acceder al código fuente original.

Estas ideas de software libre como las conocemos hoy fueron desarrolladas y trabajadas durante mucho tiempo por Richard Stallman y sus seguidores, miembros de la “Free Software Fundation” (Fundación del Software Libre).

En general las ideas del software libre buscan promover la generación de mejor software (a través de la suma de los pequeños aportes de cada persona), y colaborar para que toda la sociedad se vea beneficiada con los avances del software. Es decir, logramos mejorar la sociedad al tener disponibles más y mejores herramientas.

Para más información sobre el software libre pueden visitar el sitio de Internet del Proyecto GNU (http://www.gnu.org), que tiene una gran cantidad de documentos relacionados con la filosofía del software libre.

Usando GNU/Linux

Modo Gráfico - Modo Consola

Como ya dijimos anteriormente, GNU/Linux puede utilizar el Modo Gráfico, si utiliza la aplicación XFree86. Por otro lado, llamamos Modo Consola, al modo que es puramente texto. Gran cantidad de los temas que se enseñan en este curso se pueden probar en modo consola, o bien en una consola gráfica, dentro del modo gráfico.

Comenzando la Sesión

Dado que UNIX es un sistema multiusuario, para poder comenzar a utilizarlo debemos ingresar el usuario y password que nos identifica. Esto lo podemos hacer tanto en modo gráfico como en modo consola. Muchas veces, cuando ingresemos nuestra contraseña a un sistema UNIX, no veremos los caracteres (ni siquiera un '*'). Esto se debe a que de esta manera es más difícil que un observador sepa la cantidad de caracteres que contiene nuestra contraseña.

Una vez que hayamos ingresado, tendremos acceso a nuestros archivos, y podremos ejecutar una gran cantidad de aplicaciones, según los permisos que se le hayan dado a nuestro usuario. Es decir que, cada usuario que utilice el sistema tendrá un tratamiento distinto. A esto nos referimos Cuando decimos que todo sistema derivado de UNIX es multiusuario.

En particular, existe un usuario que es el encargado de administrar el sistema, es el usuario que tiene acceso a los archivos de configuración, a instalar y desinstalar el sistema. Este usuario suele tener el nombre de root, o también superusuario.

Terminar la sesión

Para salir del shell, use el comando exit, o presione las teclas **CTRL+D** desde el PROMPT. Si se encuentra en el modo gráfico deberá ejecutar desde el menú de inicio => logout => el sistema le presentara una ventana de dialogo y elegirá salir o reiniciar. El modo de terminal o consola:

 # exit
$ logout

Serán suficientes. Después de salir del sistema, el sistema presentará de nuevo un login PROMPT si está en modo Shell o un desktop si en el modo gráfico.

Comandos Unix

Los comandos se ejecutan desde un shell. Se escriben en el PROMPT del shell y luego se presiona ENTRE. El shell trata de interpretar el comando, si es reconocido se ejecuta; si no devuelve un error (precedido por un $ si es usuario normal o # si es la cuenta de root). Algunos comandos requieren parámetros, también conocidos como argumentos.

Ejemplos:

$ date
Thu Jun 14 12:28:05 BST 2001

El símbolo de $ es el prompt y claro está no lo escriba en la línea de comandos.

$ echo Saludo Todos
Saludo Todos

Los comandos diferencian entre mayúscula y minúscula, y casi siempre son escritos en minúscula

$ echo repítelo
repítelo
$ ECHO REPITELO
bash: ECHO: command not found (comando no encontrado)

Estándares y convenciones Unix

- Existen diferencias entre los sistemas operativos GNU/Linux y Unix
- Especialmente en lo que concierne la administración del sistema
- A menudo cosas específicas de GNU/Linux en estas áreas

Nombres de Archivo

En Unix, cada archivo tiene un nombre que le da el usuario para poder utilizarlo. Un nombre de archivo puede ser casi cualquier cadena de caracteres, donde el único carácter ASCII que no se puede usar es el slash (/), que en Unix se usa como separador de directorios y archivos. Existen otros caracteres que se recomienda no usarlos pues tienen significado especial para el intérprete de comandos de Unix, como son (! # & ( ) * ‘“ `; | < > @ $ ^ { } ¿?: \) espacio backspace tab + - y los caracteres de control. Las mayúsculas y minúsculas son consideradas distintas en Unix y, por ejemplo, los nombres CARTA, carta y Carta corresponden a tres archivos distintos.

Se recomienda que los nombres de archivo no tengan más de 14 caracteres porque, salvo casos especiales, Unix considera iguales a dos archivos si coinciden en los primeros 14 caracteres (esto no es el caso para versiones modernas de Unix, las que pueden aceptar nombres de archivo de más de 80 caracteres).

Directorios

Los directorios son la base del sistema jerárquico de archivos de Unix. Son grupos de archivos que sirven para clasificarlos y organizarlos de acuerdo a las necesidades de los usuarios. Un directorio puede contener otros directorios y archivos, y así sucesivamente. En teoría, no existe limitación del número de archivos y directorios que se puedan crear en un directorio, con excepción del tamaño del dispositivo donde se almacena. El sistema de directorios y archivos se puede graficar en lo que se conoce como estructura de árbol.

Práctica1

Ejercicio 1

1) ¿Qué es Unix?
2) Nombre Cinco variedades de Unix
3) ¿Qué es POSIX?
4) Nombre 5 Personajes de Software Libre.
5) Nombre 5 proyectos del Software Libre.

Ejercicio 2

6) Ingrese al Sistema (Log in).
7) Salga del Sistema (Log out).
8) Ingrese de nuevo (Login). Abra un terminal en el XWindow.
9) Salga del shell; la ventana del terminal debe cerrarse.
10) Inicie otro shell. Ejecute los siguientes comandos en este orden.

```
# date
# whoami
# hostname
# uname
# uptime
```


FCLD
http://www.codigolibre.org

Capítulo2

Todos los niños aprenden sobre todo jugando, por eso fue de verdad muy importante que Linus entrara al mundo de la informática, en un momento en que los computadores eran todavía muy simples. Para un niño de 10 o 12 años era fácil darse cuenta de lo que había en el interior de los computadores, comprender su esencia. Para los niños y las niñas de hoy día, es mucho más difícil comprender el mismo tipo de cosas, por que hay demasiado niveles y elementos complicado en los computadores actuales, para que puedan adquirir el tipo de instinto y compresión que " Linus " adquirió a través del juego.

Nils Torvalds ... Padre de Linus Torvalds

Los Inicios

Los Objetivos de Este Capítulo son:

1. **Manejo de los comandos básicos**
2. **Completar comandos con TAB**
3. **Uso del comando history**
4. **Archivos y Directorios**
5. **Encadenamiento de comandos**
6. **Tuberías y Redireccionamiento**

Los Inicios

Primeros Comandos Básicos

Crear archivos con el comando cat

Ejecutando **cat archivo** podremos ver el contenido de archivo. Este comando puede recibir una serie de archivos, y el resultado será que nos mostrará un archivo a continuación del otro. Un caso especial se produce cuando ejecutamos **cat** sin ningún nombre de archivo. En este caso, el comando esperará a que nosotros le demos una entrada, y la irá reproduciendo línea por línea. Hasta que presionemos la combinación **Ctrl+d**, que indica que la entrada ha terminado.

Una de las maneras más sencilla de crear un archivo es con el comando cat:

```
$ cat > listado.txt
Felipe
Carlos
Luisa
Manuel
```

Note el símbolo de (>) - es necesario para crear el archivo. El texto que escribas será escrito al archivo especificado como argumento en este caso listado.txt Presione Ctrl+d después de la última entrada para denotar el fin de archivo.

Desplegar archivos con el comando cat

Existen muchas maneras de desplegar un archivo a pantalla para leer, una de las maneras más fácil es con el comando cat:

```
$ cat listado.txt
Felipe
Carlos
Luisa
Manuel
```

Note que no se esta utilizando el símbolos (>). El contenido es desplegado de inmediato.

Borrar archivos con rm

Para borrar archivos utilizamos el comando **rm**. Hay que usarlo cuidadosamente, porque una vez que los archivos han sido borrados, no pueden recuperarse de ninguna forma. Si deseamos que **rm** nos pregunte si queremos borrar o no un archivo, debemos utilizar la opción **-i**, mientras que si deseamos que no nos pregunte utilizamos la opción **-f**. Dependerá de la configuración del sistema cual de estas dos opciones es la que está seleccionada por omisión.

$ rm archivo_borrar.txt

Al ejecutar este comando el archivo **archivo_borrar.txt** es eliminado y:

- No existe un SAFACON
- No existe el comando unrm

Mensajes de los comandos Unix

Típicamente, comandos exitosos no devuelven ningún mensaje. Mensajes son desplegados solo en caso de error. Tomando por ejemplo el comando anterior de **rm** si encontró el archivo y lo logro borrar con éxito no nos informa de nada, pero si por alguna razón el comando falla nos devuelve un mensaje

Copiando archivos con cp

El comando **cp** es el que se utiliza para copiar archivos.

Si escribimos **cp viejo nuevo**, copiaremos el archivo viejo con el nombre nuevo. Es decir, el archivo origen se escribe primero y a continuación el archivo que se va a crear. Una vez hecha la copia, tendremos dos archivos diferentes, con el mismo contenido. Por otro lado, también podemos ejecutar: **cp archivo1 archivo2 directorio** de forma que los archivos archivo1 y archivo2 se copiarán dentro de directorio.

$ cp archivo.pdf otro-nombre-archivo.pdf

Renombrar y mover archivos con mv

Muy similar a cp, el comando mv es el que se utiliza para mover archivos de un lugar a otro, o para cambiarle el nombre a un archivo. Si ejecutamos, **mv viejo nuevo**, el archivo viejo habrá pasado a llamarse nuevo.

$ mv viejo nuevo

Por otro lado, si ejecutamos **mv archivo1 archivo2 directorio**, los archivos archivo1 y archivo2 se moverán dentro de directorio.

$ mv archivo1 archivo2 directorio

Completando nombres en el Shell Bash

Otro instrumento para evitar perder tiempo es el de completar los comandos. Si teclea parte de un archivo, un comando o una ruta y después pulsa la tecla Tab, la bash le mostrará o la parte del nombre del archivo/ruta que falta o emitirá un bip. Si escucha un bip, bastará con que pulse la tecla Tab para obtener una lista de archivos/rutas que se corresponden con lo que está tecleando.

Por ejemplo, si se olvida del comando updatedb, pero recuerda parte del mismo, puede utilizar su para convertirse en root, en el prompt de la shell teclee up, y pulse la tecla Tab dos veces, entonces verá una lista de posibles comandos que empiezan con la sílaba “up”, como **updatedb** y **uptime**. Añadiendo la letra “d” a up y pulsando otra vez la tecla Tab, el comando será completado.

De esta manera aunque la máquina sea apagada al final del día, no resulta difícil actualizar la base de datos **slocate**: Existen muchas posibilidades de que el comando sea salvado en el archivo history o bien puede utilizar la tecla Tab para completar el nombre del comando (siempre y cuando recuerde al menos cómo

empieza el nombre del comando). Por ejemplo:

$ rm arch

Presionar Tab puede retornar algo así:

$ rm archivo.txt

También funciona con nombres de comandos

Por ejemplo, **startk** puede ser completado a **startkde** si ningún otro comando empieza con **"startk"**

Historial de los comandos "history"

Si usted teclea **history**, verá una lista numerada, que le mostrará los últimos 500 comandos que ha utilizado. Usted probablemente no necesitará ver todos los últimos 500 comandos, por eso el comando **history 20** podrá resultarle útil. De esta manera, sólo los 20 últimos comandos introducidos serán visualizados (puede usar cualquier número con este comando).

Otras combinaciones de teclas

Aquí se muestran otras combinaciones de teclas que le serán útiles:

"**Bang, bang**": Si teclea **!!** (Llamado "bang bang") ejecutará el último comando.

"**Bang *número***": Si teclea **!*número*** (como !302) ejecutará el comando etiquetado con el número 302 en el archivo histórico.

"**Bang *string***": Si teclea **!*string*** (as in !rpm) ejecutará el comando más reciente del archivo histórico en el que aparezca la cadena especificada.

Up arrow y **down-arrow**: En el indicador de comandos (Bash), podrá ver los comandos previamente usados, simplemente pulsando la tecla de cursor con la flecha hacia arriba, (la tecla flecha hacia abajo le moverá hacia adelante a través de los comandos) hasta que encuentre el que desea. Pulse Enter para ejecutar el comando, como si lo hubiese tecleado en la línea de comandos.

Los comandos también pueden ser editados antes de ser ejecutados, lo cual es algo muy útil en caso de error ortográfico en comandos largos. Se puede mover con las teclas de ***Derecha*** e ***Izquierda*** para llevar el cursor hasta donde se cometió la falta. Una vez colocado el cursor en posición se puede insertar texto adicional, con la tecla de retroceso (Backspace) borrar caracteres a la izquierda del cursor y con Del o **Ctrl+D** borrar los caracteres a la derecha.

Archivos y Directorios

Cada sistema operativo tiene su propio método para almacenar datos en los archivos y en los directorios de manera que detectan cuando se agrega, modifica o se efectúan cambios. En Linux, cada archivo se almacena en un directorio. Los directorios pueden a su vez contener directorios; estos subdirectorios pueden también contener archivos u otros subdirectorios.

Se puede pensar en el sistema de archivos como una estructura similar a un árbol con los directorios como ramas. Estos directorios, pueden contener o ser los "padres" de directorios dentro de ellos (llamados subdirectorios) los cuales mantienen archivos y pueden contener otros subdirectorios al mismo tiempo. Sabemos que los árboles no pueden vivir si no tienen raíces y lo mismo le ocurre al sistema de archivos de Linux. No importa lo lejos que se encuentre un directorio dentro del árbol porque todo está conectado al directorio root, el cual se representa con el símbolo de la barra hacia adelante (/).

- Como cada directorio puede contener otros directorios, se genera una **jerarquía** de directorios
- El nivel más alto se conoce como el directorio **root (/)**
- Archivos y directorios pueden ser nombrados por su ruta o **path**

- Le muestra a los programas como encontrar un archivo
- Al directorio root se le refiere com ***(/)***
- Los otros directorios son referidos por un nombre, sus nombres son separados por un barra /

- Si una ruta se refiere a un directorio entonces puede terminar en /
  - Normalmente una barra "/" al final de una ruta no efectúa diferencia alguna.

Rutas/Paths Absolutos y Relativos

Una ruta **absoluta** empieza en la raíz de la jerarquía del directorio y nombra los directorios debajo del, ejemplos: **/etc/hostname**

Que significa que un archivo llamado ***hostname*** en el directorio ***etc*** en el directorio root

Podemos utilizar el comando **ls** para listar los archivos en un directorio en específico dando su ruta absoluta así:

$ ls /usr/share/doc/

Directorio Actual

El comando pwd es muy sencillo, nos muestra la ruta de directorios en la que estamos situados en este momento. Por ejemplo, /home/user.

- Su shell contiene un **(current directory) directorio actual** - directorio cual es el que usted se encuentra trabajando ahora mismo.
- Comandos como el **ls** usan el directorio actual como parámetro si ninguno es especificado
- Use el comando **pwd** (print working directory) para ver cual es su directorio de trabajo actual así:

$ pwd
/home/usuario

Cambie de directorio actual a otro con el comando cd:

$ cd /mnt/cdrom
$ pwd
/mnt/cdrom

Para retornar a su directorio de usuario use el comando **cd** sin especificar una ruta o directorio.

Creando y removiendo directorios

Comando mkdir: Utilizamos el comando **mkdir** directorio para crear directorios. Pueden utilizarse rutas absolutas o relativas. Es decir que si queremos crear el directorio /home/user/temp, y estamos situados dentro del directorio /home/user, podremos ejecutar mkdir temp o mkdir /home/user/temp indistintamente.

Comando rmdir: Para borrar directorios utilizamos el comando **rmdir** directorio. Solamente funcionará si el directorio está vacío. En caso contrario, habrá que borrar primero los archivos, para luego borrar el directorio.

- El comando **mkdir** crea un directorio nuevo y vació.
- Por ejemplo, para crear un directorio que almacene las cuentas de la compañía hágalo así:

$ mkdir Cuentas

Para borrar un directorio que actualmente está vacío, use rmdir así:

$ rmdir Cuentas_Invalidas

- Use rm con la opción -r (recursiva) para borrar directorios y todo su contenido así:
- $ rm -r Cuentas_Viejas
- Sea extremadamente cauteloso al utilizar el comando **rm** puede ser muy peligrosa herramienta si se utiliza incorrectamente.

Ruta (path) Relativa Paths

- Rutas no tienen que empezar desde el directorio raíz (root)
- Una ruta la cual no empieza con una barra "/" es una ruta **relativa**
  - Las rutas son relativas a algún otro directorio, casi siempre al directorio actual
- Por ejemplo, los cambios siguientes nos llevan al mismo directorio:

  $ cd /usr/share/doc
  $ cd /
  $ cd usr
  $ cd share/doc
- Rutas Relativas especifican archivos dentro de los directorios en la misma manera que las absolutas.

Directorios de Dot (.)(..) Especiales

- Todo directorio contiene dos nombres de archivos especiales que le asisten en hacer rutas relativas:
  - El directorio **(..)** apunta al directorio padre. Así es que el comando **ls ..** Listará los archivos en el directorio padre.
  - Por ejemplo, si empezamos desde el directorio ***/home/usuario***:

  $ cd ..
  $ pwd
  /home
  $ cd ..
  $ pwd
  /
- El directorio especial **(.)** apunta al directorio actual
  - Así es que **./carta.txt** es el mismo archivo que el archivo **carta.txt**

Utilizando los Directorios Dot (.) en su Ruta

Los directorios ocultos, especiales . y .. pueden ser utilizados en rutas al igual que cualquier otro nombre de directorio:

$ cd ../dir-anterior/

El cual significa "ir al directorio ***dir-anterior*** en el directorio padre del presente directorio"

Es común ver "../" utilizado para navegar hacia atrás varios directorios desde el directorio actual:

$ ls ../../../../directory-bien-retirado/

Archivos Ocultos

El directorio (.) se utiliza comúnmente para denotarse a si mismo, para significar "directorio actual"

Archivos Ocultos

- Los directorios especiales **.** y **..** no son visibles cuando se ejecuta el comando de listar **ls** Son archivos ocultos
- Regla Simple: archivos que sus nombres empiezan con un . son archivos ocultos
- Para hacer que **ls** despliegue todos los archivos, hasta los ocultos, ejecútelo con la opción **-a** (all):

  . .. .bashrc .profile reportes.doc
- Archivos ocultos a menudo se utilizan para los archivos de configuración colocados en el directorio home del usuario
- Tienes acceso a leer los archivos ocultos - solo no se listan con el comando **ls** por defecto

Ruta a los Directorios home

- El símbolo ~ (tilde) es una abreviación para su directorio home
  - Así es que para el usuario "**usuario**", las dos sentencias que siguen son equivalente:

    $ cd /home/usuario/documentos/
    $ cd ~/documentos/
- La ~ se **expande** por el shell, y los programas solo ven la ruta completa
- Puedes accede a los directorios home de los otros usuarios utilizando la ~, por ejemplo:

  $ cat ~silvia/cartas.txt
  Las sentencias siguientes son todas equivalentes para el usuario "mike"
  $cd
  $ cd ~
  $ cd /home/mike

Buscando archivos en el Sistema

- El comando **locate** lista los archivos que contienen el texto que se suple como argumento.
- Por ejemplo, para encontrar archivos que sus nombres contienen la palabra "mkdir":

  $ locate mkdir
  /usr/man/man1/mkdir.1.gz
  /usr/man/man2/mkdir.2.gz
  /bin/mkdir
  ...
- El comando **locate** es útil para encontrar archivos cuando no estas seguro de como se llama el archivo que buscas, o donde esta guardado
- Para muchos usuarios, herramientas graficas hace de navegar por los archivos del sistema una tarea menos difícil.
  - Y ayudan en el la tarea de la administración de los archivos

Ejecutando Programas

Los Programas en GNU/Linux son archivos, y se almacenan en directorios como ***/bin*** y ***/usr/bin***

- Los programas se ejecutan desde el Shell, simplemente escriba el nombre y presione ENTRE
- Muchos programas aceptan opciones, las cuales se le añaden al nombre precedidas por "-". Por ejemplo, la opción **-l** aplicada al comando **ls** nos devuelve más información, incluyendo el tamaño del archivo y la fecha en el cual fue modificado por última vez:

  $ ls -l

drwxrwxr-x	**2**	**mike**	**users**	**4096**	**Jan 21 10:57**	**Cuentas**
-rw-rw-r--	**1**	**mike**	**users**	**345**	**Jan 21 10:57**	**notas.txt**
-rw-r--r--	**1**	**mike**	**users**	**3255**	**Jan 21 10:57**	**reportes.txt**

- Muchos programas aceptan nombres de archivos después de las opciones
  - Puedes especificar múltiples archivos separándolos con espacios

Especificar Múltiples Archivos

- A la gran mayoría de programas se les puede pasar una lista de archivos como argumentos
- Por ejemplo, para borrar más de un archivo a la vez:

  $ rm notas-viejas.txt tmp.txt cosas.doc
- Para crear varios directorios en un solo comando:

  $ mkdir Cuentas Reportes
- El uso original de **cat** fué para soldar múltiples archivos juntos
  - Por ejemplo, para listar dos o más archivos, uno después del otro:

  $ cat notas.txt más-notas.txt
- Si el nombre de un archivo contiene espacio en blanco, caracteres que son interpretados por el shell (ejemplo *), se le colocan comillas sencillas alrededor del nombre del archivo:

$ rm 'Villalona - Dominicano Soy.mp3'
$ cat '* notas importantes.txt *'

Buscar la Documentación de los Programas

Comando man: El comando más importante es **man**. Este comando nos mostrará las hojas del manual del programa que estamos queriendo buscar. Por ejemplo:

$ man date

Nos mostrará el manual del comando **date**, que ya sabemos que sirve para ver y configurar la fecha, aquí está explicado como utilizarlo. Podemos movernos dentro de las páginas de los manuales utilizando la barra espaciadora, Enter, los cursores y el mismo sistema de búsqueda que utilizamos en less. Para salir, utilizamos **q**.

Comando info: Un comando muy similar a **man**, es el comando **info**. Las páginas que nos muestra este comando suelen tener una mayor cantidad de información acerca de la aplicación sobre la cual estamos consultando. Por ejemplo

$info sh-utils

Contiene información detallada sobre algunas de las utilidades del intérprete de comandos (shell), que se verá más adelante.

Comando help: Algunos comandos (como fg), son parte interna del intérprete de comandos, y por esta razón no tienen una página del manual que los explique. Para saber de qué manera utilizar estos comandos, usamos help. La ayuda que nos da este comando es más sintética que la de **man**. Por ejemplo:

$ help jobs

Nos informará sobre el uso del comando **jobs** visto anteriormente.

Archivos con información: Dentro del directorio **/usr/share/doc**, encontramos una gran cantidad de documentos que tratan las distintas aplicaciones que tenemos instaladas en nuestro sistema. En particular, el directorio **/usr/share/doc/HOWTO**, contiene artículos sobre cómo hacer determinadas cosas dentro de nuestro sistema.

- Otras cosas como formatos de archivos y librerías también tienen sus páginas **man.**
  - Como hemos visto para leer la página **man** de un programa sólo necesitas saber su nombre:

$ man mkdir

Para salir del **man** presione la tecla **q**

- Las páginas **Man** de un programa contienen la siguiente información:
  - Una descripción de lo que hace
  - Una lista de las opciones que acepta
  - Informaciones adicionales como el nombre del autor, etc.

Especificando Archivos con Metacarácteres (Wildcards)

Además de ejecutar los comandos que nosotros le indicamos, el shell interpreta ciertos caracteres especiales, a estos caracteres los llamamos metacaracteres. Cuando nosotros utilizamos algún metacarácter, los comandos no lo reciben, sino que el shell lo reemplaza por lo que corresponda, y le pasa al comando ejecutado el resultado de ese reemplazo.

Eso es lo que entendemos por interpretar: reemplazar el carácter por otro carácter o por una cadena de caracteres, según corresponda.

Metacarácteres relacionados con archivos

Cuando el shell encuentra un **(*)**, lo reemplaza por una lista de los archivos que concuerdan con la expresión indicada.

El (*): echo * nos mostrará todos los archivos. **echo a*** nos mostrará todos los archivos del directorio que comiencen con a. **echo *o** nos mostrará todos los archivos que terminen con **o**.

$ echo /usr/local/*

Nos mostrará todos los archivos que estén en ese directorio.

En el caso de que no hubiera ningún archivo que concuerde con la expresión, generalmente, nos mostrará la expresión que hayamos escrito.

El (?): Al encontrar un ? el shell lo reemplaza por cualquier otro carácter. Es decir que la expresión que escribamos se reemplazara por todos los archivos que en esa posición tengan cualquier carácter, y en el resto de la cadena tengan lo que hemos escrito.

Por ejemplo: echo ?ola nos podría mostrar archivos como hola, sola, Pola. echo a??a, podría mostrar allá, arca, hacia.

Al igual que con el *, si ningún archivo concuerda con el patrón, generalmente, nos muestra la misma expresión que hemos escrito.

[] Encerrados por los corchetes, podemos escribir un rango de caracteres con los cuales queremos que el shell concuerde. Por ejemplo,

$ ls [af]*

Nos mostraría todos los archivos que comienzan con **a** o con **f**.

Podemos además especificar un rango de caracteres, con un guión en el medio. Por ejemplo, a-z (letras minúsculas), 0-9 (números), etc. y combinarlos con caracteres individuales siempre que no sea ambigua la interpretación. (Considerar la concordancia con el carácter -).

Por ejemplo, podemos querer sólo los archivos que comienzan con números seguidos de un -, en ese caso escribiríamos

$ ls [0-9]-* **o** **$ ls [0-9][0-9]-***

si comienzan con dos números seguidos de un -.

[^] Cuando al comienzo de la cadena que está encerrada por los corchetes encontramos el carácter ^, estamos indicando que debe concordar los caracteres que no se encuentran en el rango. Por ejemplo:

$ ls [^0-9]*

Nos listará todos los archivos que no comiencen con un número.

Metacarácteres relacionados con comandos

Ejecutar un comando es tan sencillo como escribir el comando y oprimir la tecla ENTER. Sin embargo, utilizando algunos de los metacaracteres de shell podemos combinar los comandos entre sí, y lograr resultados mucho más importantes.

El **";"** es un separador de comandos, nos permite ejecutar un comando a continuación de otro, equivalente a lo que sucedería si ejecutáramos primero uno, y al terminar ejecutáramos el siguiente. Es decir si escribimos:

$ ls; echo Hola

Veremos la sálida del **echo** a continuación de la del comando **ls**.

( ) Los paréntesis sirven para encerrar grupos de comandos, y tratarlos como si fueran uno solo.

El & manda el comando a background, esto quiere decir, que nos devuelve la línea de comandos inmediatamente después de oprimir Enter, mientras el comando sigue ejecutándose en segundo plano.

La ejecución de tareas en segundo plano ya se ha estudiado anteriormente, cuando se vieron los comandos relacionados con procesos. Este metacarácter funciona de manera equivalente, y sus resultados pueden corroborarse utilizando el comando jobs.

Para ver un ejemplo, vamos a usar un nuevo comando, **sleep**, (un comando simple que espera una determinada cantidad de segundos). Por ejemplo:

$ sleep 5

Espera 5 segundos antes de devolvernos la línea de comandos.

Ahora, utilizando &:

$ (sleep 20; echo Hola) &

Al escribirlo nos mostraría el PID del comando que estamos ejecutando, y nos devolvería el shell; 20 segundos después veremos aparecer "Hola" en nuestra línea de comandos. Antes de que termine de ejecutarse, podemos ejecutar **jobs** y observar que el proceso se está ejecutando, o bien **ps** y observar que el comando en ejecución es sleep. Además, el & nos puede servir para separar comandos: cada vez que lo utilizamos para separar comandos, mandaría al comando que esté a su izquierda a background.

Otros metacarácteres

'...' Al encontrar una cadena encerrada entre ' ', el shell tomaría el contenido de la cadena literalmente, es decir, sin interpretar los metacaracteres contenidos en ella.

Por ejemplo, echo '* ?* [A-Z-]*' nos mostraría * ?* [A-Z-]*.

Note que si no cerramos las comillas y presionamos ENTER, el shell nos mostraría una línea en blanco esperando que sigamos ingresando nuestro comando, hasta que cerremos las comillas.

La \ - Utilizamos una \ para escapar el siguiente carácter. Escapar significa que el shell no lo interpretaría como un metacarácter.

Por ejemplo echo * nos mostraría un *.

El #- Es el señalador de comentarios. Si el shell encuentra un # al comienzo de una palabra, descartaría todos los caracteres hasta el final de la línea. Por ejemplo, echo 3.1416 # Pi con un error de 0.0001 nos mostraría únicamente3.1416.

- Use el comodín **(*)** para especificar más de un nombre de archivo como argumento de un programa o utilitario, por ejemplo:
  $ ls -l *.txt
  -rw-rw-r— 1 miguel users 108 Nov 16 13:06 report.txt
  -rw-rw-r— 1 miguel users 345 Jan 18 08:56 notes.txt
- El shell expande el comodín, y le pasa la lista completa al programa o utilitario.
- Con solo utilizar el * expandirá a todos los archivos en el directorio (Menos los ocultos):
  $ rm *
- Nombres con comodines se les llama **globs**, y el proceso de expandirlos es conocido como **globbing.**

Entrada y Salida

UNIX tiene un extenso manejo de entrada y salida, es una de las características principales que nos permite combinar pequeñas herramientas para lograr resultados más complejos. La mayoría de los comandos UNIX que nosotros utilizamos tienen una entrada estándar, una salida estándar y una salida para errores estándar. Las denominamos **stdin**, **stdout** y **stderr** respectivamente.

La entrada estándar por omisión es el teclado, mientras que la salida estándar y la salida de errores son, por omisión, la pantalla.

Un comando genérico, lee datos de la entrada estándar, los procesa de alguna manera, y luego emite el resultado por la salida estándar. En el caso de que durante el proceso hubiera algún error, emitiría un aviso de ese error por la salida de errores.

El Shell se encarga de relacionar estos tres, lo cual no impide que un determinado programa maneje su entrada y su salida de una manera diferente.

El carácter > .- Nos permite direccionar la salida estándar de un comando a un archivo. Por Ejemplo:

ps ax > procesos.txt

Guardaría en el archivo procesos.txt la salida del comando **ps**.

El carácter < .-Nos permite direccionar la entrada estándar de un comando desde un archivo. Por ejemplo, el comando mail nos sirve para mandar mensajes a otros usuarios, si escribimos:

mail miguel_p < archivo.txt

Mandaría un mensaje con el contenido del archivo.txt al usuario **miguel_p**.

Usar un >> - En lugar de un > nos permite direccionar la salida estándar a un archivo, sin sobrescribirlo, sino que le agrega los datos que nosotros queramos al final. Si ahora hacemos:

ps ax >> procesos.txt

Tendremos el listado de procesos dos veces en un mismo archivo.

Utilizar 2> - Nos permite redirigir la salida de errores a un archivo. Por ejemplo, si no existe un **archivo.txt** y si ejecutamos:

ls archivo.txt 2> errores.txt

El error del comando **ls**, indicándonos que el archivo.txt no existe se almacenaría en **errores.txt**.

Usar | - Para relacionar la salida estándar de un comando, con la entrada estándar de otro comando, utilizamos el carácter |. Ejemplo, podemos relacionar la salida de **ls** con la entrada de **wc**.

Haciendo **ls | wc**, la salida de este comando será la cantidad de líneas, palabras y caracteres que produjo ls. Este comando recibe el nombre de pipe, que en español significa cañería o tubería. Es decir que es un comando que entuba la salida de un comando con la entrada de otro.

Es interesante observar lo que sucede cuando hacemos: ls > nuevo-archivo, esto es, el archivo nuevo aparece dentro del listado que hace ls. Esto se debe a que el shell, al hacer la relación entre el archivo y el comando, crea el archivo, y luego llama al ls.

Además es necesario tener en cuenta que un comando no puede utilizar como entrada y salida un mismo archivo. Por ejemplo, al ejecutar cat archivo > archivo, el intérprete de comandos nos indicaría que esto no es posible.

Encadenando Programas

- El comando **who** lista los usuarios actualmente utilizando el sistema
- El comando **wc** cuenta los bytes, palabras, y líneas en su entrada
    - Los combinamos para contar cuantos usuarios tienen sesión en el sistema:

  $ who | wc -l
- El símbolo | crea una tubería entre dos programas, le pasa la salida de uno a la entrada del otro.
    - La salida del comando **who** se le pasa al comando **wc**

La opción **-l** logra que el comando **wc** solo imprima el número de líneas y no toda la salida del comando **who**

Otro ejemplo es, para contar todas las palabras, líneas y caracteres de los archivos con extensión txt sólo

tenemos que ejecutar el siguiente comando:

$ cat *.txt | wc

Interfaces Grafica y Texto

Las distribuciones modernas de GNU/Linux proveen un interfase grafica al usuario (**graphical user interfase, (GUI))**

- Los sistemas GNU/Linux utilizan el sistema Xwindow para proveer graficas
- El X es solamente un programa más, no es parte de Linux

Las mayorías de distribuciones se inician automáticamente en el **X**

GNU/Linux se puede utilizar desde la línea de comandos sin un GUI

Pulse **Ctrl+Alt+F1** para entrar en una consola de texto – haga un login idéntico al del **X**

Use Ctrl+Alt+F2, Ctrl+Alt+F3, etc., para cambiar entre las terminales virtuales – los distros proveen 6 usualmente.

Use **Ctrl+Alt+F7**, o cualquiera que fuese la ultima terminal virtual, para regresar al **X**

Editores de Texto

- Los editores de Texto son para editar archivos de texto simple.
  - No proveen capacidad de formato avanzados como los procesadores de palabras.
  - De extrema importancia – saber manipular archivos de texto en Unix es FUNDAMENTAL.
- Los editores más usados son Emacs y Vim, ambos son sofisticados pero toman tiempo en dominar.
- Editores más simple son Nano, Pico, Joe, y gráficos son Kedit y Gnotepad.
- Algunos programas lanzan un editor de texto.
  - Ellos utilizan la variable **$EDITOR** para decidir cual.
  - Casi siempre por default es vi, pero puedes cambiarla.
  - Este ejemplo es parte de la filosofía de componente de Unix, programas pequeños haciendo cosas específicas.

Práctica2

Ejercicio 1

1) Use el comando ls para listar los archivos en el directorio actual.
2) Cree un nuevo archivo utilizando el comando cat de la manera siguiente:
3) $ cat > hola.txt
4) Hola Mundo!
5) Esto es solo una Prueba.
6) Presione Enter al final de la última línea, y entonces Ctrl+D para denotar el final del archivo.
7) Use el ls de nuevo para verificar que el archivo recién creado existe.
8) Despliegue el contenido del archivo.
9) Despliegue el mismo archivo de nuevo, pero esta vez utilice solo las teclas del cursor para ejecutar el comando sin tener que digitarlo de nuevo.

Ejercicio 2

1) Cree un segundo archivo. Llámelo nomina.txt, escríbale cualquier cosa.
2) Revise con el comando ls, que existe.
3) Despliegue el contenido del archivo. Minimice la digitación que se necesita escribir para lograrlo:
4) Revise el history para encontrar el comando que creo el archivo anterior.
5) Cambie el comando para que en ves de crear el archivo lo despliegue a pantalla.

Ejercicio 3

1) Después de cada uno de los siguientes pasos, use el comando **ls** y **cat** para verificar que ha sucedido.
2) Copie nomina.txt a un archivo nuevo llamado descuento.txt. Use el Tab para evadir tener que escribir el nombre completo del archivo.
3) Ahora copie hola.txt a descuento.txt. ¿Que sucedió?
4) Borre el archivo original, hola.txt.
5) Renombre descuento.txt a impuestos.txt.

6) Trate de borrar el archivo hola.txt con el comando rm. ¿Que sucede?

7) Trate copiar nomina.txt de nuevo, pero no especifique el nombre del archivo al cual se le copiará. ¿Que sucedió?

Ejercicios 4

1) Use el comando pwd para ver en que directorio se encuentra.

2) Si no estas en su directorio home (/home/NOMBRE_DE_USUARIO) use el comando cd sin ningún argumento, y ejecute pwd de nuevo.

3) Use cd para visitar el directorio root, y listar los archivos ahí dentro. Home debe ser uno de ellos.

4) Vaya al directorio home y liste el contenido de nuevo. Debe existir un directorio por cada usuario del sistema incluyendo el suyo (puedes utilizar el comando whoami para verificarlo).

5) Regrese a su directorio home para confirmar que esta de regreso donde empezó.

Ejercicios 5

1) Cree un archivo de texto en su directorio home y llámelo ***merengue***, conteniendo las siguientes oraciones:

   Baile compadre Juan

   Quisqueya

2) Renómbrelo clasicos.txt

3) Cree un nuevo directorio en su directorio home y llámelo ***merengue***.

4) Mueva el archivo ***clasicos.txt*** dentro del directorio ***merengue***.

5) Desde el escritorio grafico abra un manejador de archivos (explorador), y encuentre su directorio home, también confirme el directorio ***merengue*** y el archivo ***clasicos.txt***.

6) Con un editor de texto grafico edite el archivo ***clasicos.txt***.

Ejercicios 6

1) Desde su directorio home, liste los archivos en el directorio ***/usr/share***.

2) Cambie al directorio ***/usr/share***, confírmelo con ***pwd***. Liste los archivos en este directorio y los del directorio ***doc***.

3) Ahora liste los archivos en el directorio padre, los del directorio encima de este en la jerarquía.

4) Ejecute el siguiente comando, Asegúrese de entender el resultado: **$ echo ~**

5) Use el comando **cat** para desplegar el contenido de un archivo de texto el cual reside dentro de su directorio home (cree uno si no existe), usando el sintaxis "~/" para referirse a el. No debe importar desde que directorio usted se encuentre para ejecutar el comando.

Ejercicios 7

1) Use el comando ***hostname***, sin opciones, para imprimir el nombre del ***host*** de la maquina en uso.

2) Use el **man** para desplegar la documentación del comando **hostname**. Investigue como hacerlo que imprima la dirección IP de su maquina en vez del nombre de ***host*** de su maquina. Vaya a la sección de "Opciones" del ***man***.

3) Use el comando **locate** para encontrar en el sistema los archivos que contienen el texto 'hostname'. Cual de la lista de archivos desplegados es el programa ***hostname***? Ejecútelo ahora con su ruta y nombre absoluto para ver si es este realmente.

Ejercicios 8

1) Utilizando el comodín * (wildcard) solo como argumento de un comando es expandido por el shell como una lista de todos los archivos en el directorio actual. Use el comodín con el comando ***echo*** para probar el resultado (Asegúrese que solo sean archivos de texto que se encuentren en el directorio): **$ echo *

2) Use las comillas simple para que se imprima el símbolo * y que el shell no lo interprete.

3) Agréguele otro archivo al directorio ***merengue*** que usted creó anteriormente, llámelo ***modernos.txt***:

Dominicano soy - Fernando Villalona

Bachata Rosa - Juan Luís Guerra

4) Use el comando ***cat*** para desplegar ambos archivos, por nombre y con comodines (wildcard).

5) Copie el directorio ***merengue*** y todo su contenido, llámelo ***Merengue***, use ***cp***.

6) Finalmente, borre el directorio ***merengue*** con el comando ***rm***.

Ejercicios 9

1) Explicar en qué se diferencian ls * y echo *.

2) Explicar en qué se diferencian ls / y echo /.

3) Crear un archivo que contenga la cantidad de archivos en un directorio.

4) Crear dos archivos: a.txt, que contenga hola, y b.txt, que contenga chao. Luego concatenarlos en un archivo ab.txt.


FCLD
http://www.codigolibre.org

Capítulo3

Una analogía es comparar una distribución de Linux con una hamburguesa que compras en un restaurante de comida rápida. No tienes idea de lo que te estás comiendo. En cambio, la FSF no te da una hamburguesa, sino la receta para hacer la hamburguesa. Te permite revisarla, eliminar los ingredientes no deseados y añadir tus propios ingredientes para mejorar el sabor de tu hamburguesa. Cuando estés satisfecho con la receta entonces empiezas a prepararla. Tu la cocinas de la forma que prefieres: asada, cocida, frita, a la barbacoa, o comerla cruda.

Organización FHS

Gestión de Archivos desde la Línea de Comando

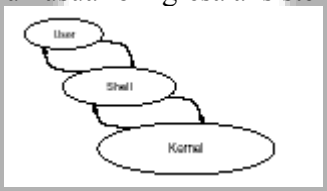
Los Objetivos de este Capítulo son:

1. **Qué son los Shells**
2. **Trabajar desde la línea de comandos**
3. **Comandos necesarios para manejarse desde el Shell**
4. **Variables del Shell**
5. **Localizar Archivos**

Trabajar en la Línea de Comandos

Los Shells

- Un shell provee una interfase entre el usuario y el kernel del sistema operativo
- Los shells o son GUIs (graphical user interfase) o CLI (command interpreter)
- Los Shells Tradicionales de Unix son Interfases de Líneas de Comandos (CLIs)
- Son iniciadas automáticamente cuando un usuario ingresa al sistema



El Shell Bash Shell

- El interprete de comandos más popular de GNU/Linux's es el bash (The **Bourne-Again Shell)**
- Es mucho más sofisticado que el ***sh*** original de Steve Bourne
- Puede ser ejecutado como ***sh***, y así reemplazar el Shell de Unix original
- Te presenta un prompt y espera que los comandos sean escritos
- Aunque nos concentramos solo en ***Bash***, el shell ***tcsh*** también es popular
- Basada en el diseño de la anterior Shell C (csh)

Comandos del Shell

- Los comandos del Shell consisten de palabras que se digitan en la línea de comandos
- Separadas por espacios en blanco
- Primero es el comando, seguido por opciones y luego los argumentos
- Por múltiples razones, algunos comandos son parte del Shell
- El número de comandos que son parte del Shell es pequeño

Argumentos de la Línea de Comandos

- Las palabras después del nombre del comando se pasan como argumentos
- La mayoría de los comandos agrupan estas palabras en dos categorías o grupos:
    - Las Opciones, casi siempre empiezan con uno o dos guiones
    - Seguido por nombres de archivos, directorios, etc., sobre los cuales se opera
- Las opciones casi siempre se colocan antes de los argumentos, pero para muchos comandos no es obligatorio
- La opción especial de '--' indica el fin de las opciones, y nada después de los guiones se conjuga como una opción, aunque empiece con un guión -

El Sintaxis de las Opciones de la Línea de Comandos

- La mayor parte de los comandos Unix tienen una sintaxis estandarizada para pasarle las opciones:
- Opciones de una letra empiezan con un guión, Ej., -B
- Existen opciones menos crípticas, que son palabras completas o frases, y empiezan con dos menos, por ejemplo --ignore-backups
- Algunas opciones mismas que toman argumentos
- Usualmente los argumentos son la próxima palabra: **sort -o *output file***
- Algunos programas usan estilos diferentes de opciones desde la línea de comandos
- Por ejemplo, opciones largas a veces empiezan con un solo – y no con dos --

Ejemplos de Opciones de los Comandos

- Lista todos los archivos en el directorio actual: $ **ls**
- Lista todos los archivos en el formato largo (dando más información): $ **ls -l**
- Lista la información completa sobre archivos en específico: $ **ls -l notas.txt reportes.txt**
- Lista toda la información de todos los archivos ***.txt***: $ **ls -l *.txt**
- Lista todos los archivos en formato largo, incluyendo los ocultos:

  $ ls -l -a
  $ ls -la

Variables del Shell

- Las **variables Shell** pueden ser utilizadas para almacenar valores temporarios
- Los valores de las variables del Shell se asignan de esta forma, ejemplo:
- $ archivos="notas.txt reportes.txt"
- Las dobles comillas se usan por el espacio en blanco, pero es mejor usarlas todo el tiempo
- Para imprimir el valor de una variable use el comando echo así:
- $ echo $archivos
- El símbolo ($) le dice al shell insertar el valor de la variable en la línea de comando
- Use el comando **set** (sin argumentos) para listar todas las variables del Shell

Variables de Ambiente

- Las variables del Shell son privadas de ese Shell
- Un tipo especial de variables del shell llamados **variables de ambiente** se les pasa a los programas ejecutados desde el **Shell.**

  El ambiente de un programa son el conjunto de variables de ambiente a las cuales el tiene.

  Desde Bash, use el comando **export** para exportar una variable del shell a su ambiente:

  $ archivos="notas.txt reportes.txt"
  $ export archivos

O combinándolo en un solo comando así:

$ export archivos="notas.txt reportes.txt"

El comando **env** lista todas las variables del ambiente

Donde están los Programas almacenados

La localidad de un programa puede ser especificada explícitamente:

../ejemplo

Ejecuta el programa ejemplo que se debe encontrar en el directorio actual

/bin/ls

Ejecuta el comando **ls** que se encuentra en el directorio ***/bin.*** Si no indicamos la ruta, el Shell busca en los sitios especificados por la variable **PATH.** La variable **PATH** almacena los directorios en donde buscar los ejecutables. Los nombres de directorios son separados por punto y coma, por ejemplo:

$ echo $PATH
/bin:/usr/bin:/usr/local/bin

Si ejecutamos el comando **whoami** el sistema ejecutara ***/bin/whoami*** o ***/usr/bin/whoami*** o ***/usr/local/bin/whoami*** (o el que encuentre primero).

Configuración de las Variables de Bash

Algunas variables contienen información la cual Bash utiliza. La variable llamada **PS1** (Prompt String1) especifica como desplegar el **prompt** del S**hell.** Use el comando echo con el símbolo $ antes del nombre de la variable para ver su valor:

$ echo $PS1
[\u@\h \W]\$

Los caracteres especiales **\u, \h** y **\W** representan variables del **Shell** que contienen, respectivamente, su nombre de usuario o login name, el nombre de su maquina y el directorio de trabajo actual, Ej.:

$USER, $HOSTNAME, $PWD

Usando el comando History

- Comandos previamente ejecutados pueden ser editados usando las teclas del cursor o **Ctrl+P**
- Esto permite reusar comandos anteriores sin tener que digitarlos de nuevo
- Bash almacena una **history** (historial) de los comandos ejecutados en memoria
- Use el comando **history** para desplegar las líneas guardadas en el historial de comandos.
- **History** se almacena en el archivo ***/.bash_history*** entre sección y sección del usuario.
- Bash usa la librería **readline** para leer la entrada del usuario
- Permite comandos de edición tipo Emacs desde la línea de comandos
- Las teclas del cursor Derecha e Izquierda y Delete trabajan normal

Rehusando los Ítems del History

Comandos anteriormente ejecutados pueden ser editados para formar nuevos, usando **expansión del history.** Use **!!** para referirse al comando anterior, por ejemplo:

$ rm cartas.txt
$ echo !!
echo rm cartas.txt

rm carta.txt

Uno muy importante es ***!cadena***, cual inserta el comando más reciente que empieza con ***cadena.***

Otros útiles para repetir comandos sin ninguna modificación:

$ ls *.txt
notas.txt reportes.txt
$!ls
ls *.txt
notas.txt reportes.txt

Extraer Argumentos desde el History

El comodín **!$** refiere al ultimo argumento del comando anterior, ejemplo:

$ ls -l carta con nombre muy largo.doc
-rw-r--r-- 1 Juan users 11170 Oct 31 10:47 carta con nombre muy largo.doc
$ cp !$
cp carta con nombre muy largo.doc

Similarmente, **!^** se refiere al primer argumento del ultimo comando. Un comando de la forma ***^buscar^reemplazar^*** reemplaza la primera ocurrencia de buscar por reemplazar en el último comando ejecutado, y se ejecuta así:

$ echo $htpp_proxy
$ ^pp^tp^
echo $http_proxy
http://192.18.2.7:8080

Resumen de las teclas de editar del Bash

Estos son los comandos básicos default del bash:

- **Right-** mueve el cursor a la derecha
- **Left-** mueve el cursor a la izquierda
- **Up-** comando previo en la línea del history
- **Down-** próxima línea del history
- **Ctrl+A-** moverse al inicio de la línea
- **Ctrl+E-** moverse al final de la línea
- **Ctrl+D-** borrar el carácter actualmente debajo del cursor

Existen teclas alternativas, como las del editor Emacs, en los casos que nos se puedan utilizar las teclas cursor. También existen otras teclas en combinación, un poco menos usadas, de las cuales se puede encontrar más información en las páginas man del bash, en la sección ¨Readline¨.

Combinando más de un Comando en una línea

- Puedes escribir múltiples comandos en una sola línea simplemente separándolos con “;”
- Muy útil si el primer comando toma mucho tiempo en ejecutar:
- programa_que_consume_tiempo; ls
- Alternativamente, use && para colocar los comandos que solo se ejecuten si el anterior tuvo éxito:

Comandos_de_largo_tiempo_de_ejecucion-alto_o_potencial_de_fracaso && ls

Repetir Comandos con for

- Comandos pueden ser repetidos las veces necesarias utilizando el bucle ***for***
- La estructura es: for nombre_variable in *lista*; do *comandos...; done*

Por ejemplo, para renombrar todos los archivos de extensión ***.txt*** a ***.txt.BAK***:

```
$ for file in *.txt;
> do
> mv -v $file $file.BAK;
> done
carta.txt -> carta.txt.old
reporte.txt -> reporte.txt.old
listado.txt -> listado.txt.old
```

- El comando también se pudiese escribir en una sola línea así:

```
$ for file in *.txt;do mv  -v  $file  $file.BAK; done
```

Substitución de Comandos

- **Substitución de Comandos** permite que la salida de un comando sea el argumento de otro
- Por ejemplo, use el comando ***locate*** para encontrar todos los archivos llamados ***carta.doc*** y imprime la información acerca de ellos con el comando ***ls***:
  $ ls -l $(locate carta.doc)
  $ ls -l `locate carta.doc`
- Las comillas en el segundo son las simples, también llamadas **backticks**
  - El estilo de **$()** es preferida, pero los ***backticks*** son muy usados
- Nuevas líneas son convertidas a espacios en la salida del comando primario
- Otro buen ejemplo: use el editor **vi** para editar el último archivo encontrado:
  $ vi $(locate carta.doc | tail -1)

Buscar archivos con locate

- Usar el comando **locate** es una manera simple y eficaz de encontrar archivos
- Por ejemplo, para encontrar archivos relacionados con el programa de correo ***mutt***:
- $ locate mutt
- El comando **locate** busca en la base de datos de los nombres de los archivos, esta base de datos es mantenida con el comando **updatedb**
  - La base de datos debe ser actualizada regularmente
  - Usualmente esta actualización se hace automáticamente con ***cron***
  - Debes tener cuenta que **locate** no encontrara archivos creados después de la última actualización.
- La opción **-i** hace que la búsqueda sea sensitiva a mayúscula y minúscula.

- La opción --r interpreta el argumento patrón como una expresión regular, y no como una cadena de caracteres simple.

Buscar archivos con más flexibilidad: find

- El comando **locate** solo busca archivos por su nombre, **find** puede encontrar archivos por una amplia combinación de criterios de búsqueda, el cual incluye por nombre.
- Estructura del comando: ***find directorios criterio***
- El find más simples por ejemplo es:

$ find .

- Buscar archivos por criterio simple singular:
-

$ find . -name reportes.txt

- Busca archivos en el directorio actual que su nombre es **reporte.txt**
- El ***criterio*** de búsquedas siempre empieza con un solo guión, aunque tengan un nombre largo

find pathnames search-expressions action-expressions
Find rutas búsqueda-expresiones acción-expresiones

El comando *find* busca en estructuras completas de directorio empezando con las rutas y ejecuta acciones especificadas por la *acción-expresión* en todos los archivos con atributos igualando el busqueda-*expresiones*.

Por ejemplo:

```
$ find . -name '*.config' -print
./prog1.config
./stat/mean.config
./stat/var.config
./math/matrix.config
```

Busca la estructura completa de directorio empezando por el directorio actual (especificado en este caso por un punto, **.**) por archivos con nombres que terminen en *.config e imprime el nombre de cada archivo encontrado*. En este ejemplo, *-name '*.config'* es una expresión de búsqueda (búsqueda-*expresiones*) y *-print* es una accion-expresión. Cualquier número de expresiones de búsqueda y accion puede ser usada con un solo comando find. Por ejemplo,

```
$ find . -name '*.config' -mtime 1 -print -cpio /dev/rmt1
```

Busca la estructura de directorio completo empezando por el directorio actual por archivos que su nombre terminan en *.config* que fueron modificados hace un día. La opción *-name* la cual es una expresión de búsqueda iguala archivos con nombres que terminan en *.config* y opción *-mtime* es otra expresión de búsqueda que iguala archivos que fueron modificados un día atrás. La expresión de accion *-print* imprime el nombre del archivo a pantalla y la expresión de accion *-cpio* escribe los archivos al dispositivo de cintas */dev/rmt1*. Las expresiones de búsqueda y accion más usadas se muestran más adelante.

Expresiones de búsqueda

-name '*pattern*'	Encuentra archivos con nombres igualando patrón (*pattern)*. El patrón puede incluir metacaracteres. El patrón debe estar entre comillas simples para que el shell no lo interprete.
-size [+|-]*n*[c]	Encuentra archivos que son por lo menos (+*n*) exactamente (*n*) o menos que (-*n*) *n* bloques de tamaño. En la mayoría de los sistemas el bloque es de 512 bytes o medio kilobyte. Si se le agrega una *c*, los tamaños son especificados en caracteres (e.j. bytes).
-atime +*n* | *n* | -*n*	Encuentra archivos que fueron accesado por ultima vez hace más de (+*n*) exactamente (*n*) o menos que (-*n*) *n* número de días antes. Un archivo es accesado cuando se le ejecuta un comando incluyendo el comando *find*.
-mtime +*n* | *n* | -*n*	Encuentra archivos que fueron modificados ultima vez hace más de (+*n*) exactamente (*n*) o menos de (-*n*) *n* días.
-ctime +*n* | *n* | -*n*	Encuentra archivos que fueron creados hace más de (+*n*) exactamente (*n*) o menos de (-*n*) *n* días.
-newer *filename*	Encuentra archivos que fueron modificados más recientemente que el archivo especificado por *filename*.
-type *c*	Encuentra archivos del tipo *c*. Los valores más útiles de *c* para la mayoría de los usarios es **d** para especificar un directorio, **f** para especificar archivos planos o ordinarios y **l** para especificar vínculos simbólicos. Otros valores son **b** para archivos especiales de block device, **c** para el archivo especial de caracteres y **p** para un archivo fifo o tuberías nombradas.
-perm [-]*perm-list*	Encuentra archivos con permisos igualando exactamente esos especificados por listado de permisos (*perm-list)*. Si el perm-list es precedido por un guión (**-**), encuentra archivos con por lo menos los permisos especificados.

Metacaracteres Soportados

El comando *find* soporta varios metacaracteres o wildcards (comodines) cuando busca con el criterio de nombre (e.j. usando opción -*name* '*patrón*' búsqueda-expresión).

*	Iguala cero o más ocurrencias de cualquier carácter.
?	Iguala cualquier carácter simple.
[...]	Iguala un carácter desde un conjunto de caracteres.
[*n*-*m*]	Iguala cualquier carácteres en el rango expresado por *n*-*m*.
[^...]	Iguala cualquier carácter no encerado en las llaves (brackets).
\	(\) Antes de cualquier meta carácter desactiva su interpretación.

Expresiones de Acción

-print	Imprime la ruta y nombre del archivo de cada archivo encontrado. Rutas son expresadas en relativa a la ruta de la búsqueda.
-exec *cmd*	Ejecuta comando indicado (*cmd)* por cada archivo encontrado. En el comando el archivo actual es especificado con \{\} (Backslash, llave cuadrada, Backslash, llave cuadrada cierra). El comando o *cmd debe terminar con un* **\;** (Backslash, punto y coma). Note que la expresión *exec* retorna verdadero si los comandos se completaron con éxito y falso si no. De esta manera *exec* funciona como una expresión de búsqueda además de una expresión de accion.
-ok *cmd*	Cuestionar el usuario antes de ejecutar el comando *cmd* a cada archivo encontrado. *OK* funciona idéntico al *-exec* excepto que el usuario es preguntado si confirma que desea ejecutar cada comando si imprime cada archivo encontrado seguido por un símbolo de pregunta (**?**). Si escribe **y** causara la ejecución. Escribir **n** causara que la ejecución se obvie del *cmd* y continua al próximo archivo.
-depth	Causa que las acciones que se van a efectuar a un archivo dentro de un directorio antes el mismo directorio.
-prune	Saltar el directorio iguala más recientemente.

Operadores

Expresiones de Búsqueda pueden ser combinadas para crear expresiones compuestas usando operadores. Los operadores nos permiten efectuar búsquedas más elaboradas o complejas. Se listan en la siguiente tabla en el orden que ellas son evaluadas.

\(*expresión* **\)**	Verdad si la expresión en el paréntesis es verdad. Expresiones entre los paréntesis son evaluadas primero. Los paréntesis son precedidos por una barra invertida (backslash) para no permitir que el shell las interprete como un carácter especial. Esto es necesario en los shells Bourne, Korn, c-shell y sus derivadas.
! *expresión*	El carácter de exclamación es el operador *NO*. Se evalúa como verdad si la expresión es falsa.
expresión **-a** *expresión* *expresión expresión*	El operador ***and*** evalúa a verdadero si ambas expresiones son verdaderas. La *-a* no tiene que ser especificada. Es implicada si usamos más de una expresión de búsqueda. La segunda no será evaluada si la primera es falsa.
expresión **-o** *expresión*	El operador **or**, ***-o***, evalúa a verdad si una de las dos expresiones es verdad. La segunda expresión no será evaluada si la primera expresión es verdadera.

Expresiones Avanzadas

Estas expresiones son útiles para usuarios avanzados o administradores de sistemas.

-inum *inode*	Busca archivos que su número de inodo es ***inode***.
-links *n*	Busca archivos con n número de Links o Vínculos.
-group *gname*	Busca archivos que pertenecen al grupo especificado en el argumento *gname*, *puede ser el nombre de un grupo o el número que representa su ID del grupo*.
-nogroup	Busca archivos que son de un grupo que no esta en */etc/group*.
-user *uname*	Busca archivos que pertenecen al usuario especificado por *uname*. *Uname puede ser el nombre de un grupo o el número que representa su ID del usuario.*
-nouser	Busca los archivos que pertenecen a un usuario que no esta en */etc/passwd*.
-cpio *device*	Escribe cada archivo encontrado al dispositivo *usando el formato cpio*. Para la gran mayoría de usuarios el dispositivo es el nombre físico de su cinta magnética o tape drive.
-xdev	No continué la búsqueda si se cambia a un sistema de archivos diferente. Buscar archivos que residen en el mismo sistema de archivos del de la ruta *dada como argumento*.
-follow	Sigue los links simbólicos y registra los directorios visitados. Esto no se debe de usar con la expresión *-type l*.

Ejemplos

- Buscar en todo el directorio home incluyendo todos los subdirectorios por un archivo de nombre *perdido.txt* y imprime la ruta a *perdido.txt* en la pantalla.

  $ find ~ -name 'perdido.txt' -print
  Note que la tilde (~) especifica su directorio home.

  Empezando por el directorio home, recursivamente busque todos los archivos que sus nombres terminan en extensión *.cpp* y imprima el resultado a la pantalla.

  $ find ~ -name '*.cpp' -print

  Busque todos los archivos empezando desde el directorio actual que su nombre empieza con *carta* y tiene un carácter más.

  $ find . -name 'carta?' -print
  Esta sentencia encontraría archivos nombrados carta*1*, carta*d* y carta*s*, pero no a *carta* o carta*12*. Note que el punto (**.**) especifica el directorio actual.

  Busque todos los archivos empezando con el directorio actual que su nombre empieza con una letra mayúscula y termina con un número.

  $ find . -name '[A-Z]*[0-9]' -print

  Busque todos los archivos empezando con el directorio */usr/local/install* de nombre R(*r)eadme*

  $ find /usr/local/install -name '[R,r]eadme' -print
  Esto también se puede lograr con la expresión compuesta.

  $ find /usr/local/install \(-name 'readme' -o -name 'Readme' \) -print

Busque todos los archivos empezando con el directorio actual que su nombre no termine en *.bak*.

$ find . -name '*[^.bak]' -print
También se puede lograr con el operador lógico *NOT*.

$ find . ! -name '*.bak' -print
Busque todos los archivos empezando desde el directorio home de nombre E*spec*ial*.

$ find ~ -name 'Espec*ial' -print
Note que el backslash (\) le dice a *find* que no trate a el asterisco (*) como un meta carácter.

Busque los archivos empezando desde el directorio home creados en los últimos cinco días.

$ find ~ -ctime -5 -print
Cree un listado de todos los archivos y subdirectorios contenido en los directorios *~/ccode* y *~/fortran* y guárdalo al archivo *lista-programas*.

$ find ~/ccode ~/fortran -print > lista-programas
Note que el signo más grande que (>) redirecciona la salida de *-print* desde la pantalla hacia el archivo *lista-programas*.

Cree listado de los directorios empezando por /home y guárdelo al archivo *lista-directorio*.

$ find ~ -type d -print > lista-directorio
Buscar archivos empezando en el /home que no han sido accesados en los últimos 30 días.

$ find ~ -atime +30 -print
Note que si usted ejecuta este comando una segunda vez, no encontrara ningún archivo porque el comando *find* accesa cada archivo en el directorio home cunado se ejecuta.

Busque todos los archivos empezando en el directorio home que su nombre termine en *.config* que fue modificado hace un día.

$ find ~ -name '*.config' -mtime -1 -print
Busque todos los archivos empezando en el directorio home más nuevo que el archivo *~/misdocumentos/reporte.txt*.

$ find ~ -newer ~/misdocumentos/reporte.txt -print
Busque todos los archivos empezando en el directorio home más nuevo que el archivo *~/misdocumentos/reporte.txt* y también nombrado *reporte.txt*.

$ find ~ -newer ~/misdocumentos/reporte.txt -name 'reporte.txt' -print
Busque todos los archivos empezando en el directorio home más nuevo que el archivo *~/misdocumentos/reporte.txt*, nombrado *reporte.txt* y cópielo al directorio de trabajo actual.

$ find ~ -newer ~/misdocumentos/reporte.txt -name 'reporte.txt' -exec cp \{\} . \;
Elimine todos los archivos y subdirectorios empezando con el directorio *dir-viejo*. haga que el usuario le confirme antes de ejecutar el comando de remover rm.

$ find dir-viejo -depth -ok rm \{\} \;
La opción *-depth* aquí es requerida. De otra forma, el *find* hiciera el intento de remover los directorios antes de vaciarlos y el comando rm fallará.

Busque todos los archivos empezando con su directorio home con permisos de lectura/read y escritura/write para el usuario y permisos de lectura/read solamente para el grupo y los otros.

$ find ~ -perm 644 -print
En este ejemplo los permisos los especificamos usando los números octales. Este método trabaja en las mayorías de sabores de Unix. Los Unixs modernos soportan un modo simbólico para especificar la lista

de los permisos. Por ejemplo,

$ find ~ -perm u=rw,go=r -print
Esta búsqueda hace lo mismo que el ejemplo arriba.

Busque todos los archivos en mi directorio home donde el grupo o los otros tienen permisos de escritura y use el comando **chmod** para remover ese permiso.

$ find ~ \(-perm -020 -o -perm -002 \) -exec chmod go-w \{\} \;
$ find ~ \(-perm -g=w -o -perm -o=w \) -exec chmod go-w \{\} \;
El primer ejemplo usamos números octales y el segundo usa modo simbólico de especificar la lista de permisos a el comando find.

Busque todos los archivos empezando con el directorio actual más grande que 1000 bloques (alrededor de 500 kilobytes en la mayoría de los sistemas).

$ find . -size +1000 -print

Busque todos los archivos regulares empezando por el directorio actual más grande de 1000 bloques que su nombre no termine en *.Z* y cuestione el usuario antes de comprimirlos con el comando **compress**.

$ find . ! \(-name '*.Z' \) -type f -size +1000 -ok compress \{\} \;

Busque todos los archivos empezando en el directorio actual que sus nombres terminen en *.ssd01* o *.sct01* y cópielo al directorio *~/saslib* y que pregunte antes de removerlos solamente si el comando *cp tubo éxito*.

$ find . \(-name '*.ssd01' -o -name '*.sct01' \) -exec cp \{\} ~/saslib \; -ok rm \{\} \;

Empezando por el directorio home, busque todo los archivos que sus nombres terminan en *.bak* pero no busque en el directorio de *backups*.

find ~ \(-name '*.bak' -o \(-name 'backups' -prune \) \) -type f -print

Ejemplos Avanzados

Busque el archivo en el directorio actual con inode número 1428846 y cuestióname antes de renombrar el archivo.

$ find . -inum 1428846 -ok mv \{\} newname \;
Esto es útil para renombrar archivos con caracteres especiales en su nombre. Note: Para encontrar el número de inode de un archivo con el comando ls use la opción *-i*.

Busque todos los archivos empezando en el directorio */usr/home* que no pertenecen a un grupo listado en el archivo */etc/groups*.

 # find /usr/home -nogroup -print
Note que el símbolo de número (#) se usa para denotar el prompt de Unix porque se necesita ejecutar desde la cuenta del superusuario.

Busque todos los archivos empezando en el directorio */usr/bin que tienen exactamente 5* links.

 # find /usr/bin -links 5 -print

- Busque todos los archivos empezando en el directorio /usr/home que le pertenecen al usuario miguel y cámbiele el dueño al archivo a root usando el comando chown.
  # find /usr/home -user miguel -exec chown root \{\} \;
- Busque todos los archivos regulares en el directorio */usr* que han sido modificado en los últimos 5 días y cópialos a un dispositivo de cinta tape. Siga los vínculos simbólicos pero no busque archivos en otro sistema de archivos.
  # find /usr -follow -xdev -mtime -5 -cpio /dev/rmt1

Práctica3

Ejercicios 1

1) Use el comando **df** para desplegar la cantidad de espacio en el disco duro usada y disponible.

2) Revise las paginas man del comando **df**, y encuentre la opción que permitirá que el comando desplegué la salida en más amistosa de leer para los humanos. Experimente con ambas las opciones de única letra y las de nombres largos.

3) Ejecute el shell, bash, y analice a ver que pasa. Recuerde que usted ya se encontraba ejecutando un shell bash. Trate de salir del shell que lanzo con el comando exit.

Ejercicios 2

1) Trate el comando **ls** con las opciones **-a** y **-A**. ¿Cual es la diferencia entre ellas?

2) Escriba un loop cual hace un recorrido de todos los archivos de un directorio y imprime los nombres de ellos con el comando **echo**. Si lo escribe todo en una línea, le será más fácil luego ejecutarlo desde la línea de **history**.

3) Cambie el loop para que saludo un número de gente en el aula (no cree archivos con estos nombres).

4) Claro esta, una manera más simple para imprimir una lista de los nombres de archivos es con **echo *.** Porque fuese esto útil, cuando casi siempre utilizamos el comando **ls**?

Ejercicios 3

1) Use el comando **find** para listar todos los archivos y directorios debajo de su directorio home. Experimente con **-type d** y **-type f** criterio para listar solo archivos y/o directorios.

2) Use el comando **locate** para encontrar los archivos que contienen la cadena ¨passwd¨. Intente la búsqueda ahora con el comando **find**, buscando en todo el sistema de archivos. Necesitaras utilizar el comodín * entre la cadena *passwd* en el patrón de búsqueda.

3) Investigue que hace el criterio de búsqueda del comando find -iname.

Ejercicio 4

1) Usando **cut** como despliegue una lista de usuarios ingresados en el sistema. (Verifique con **who**)

2) En el ejemplo de arriba imprima los usuarios sin duplicados y en orden alfabética.

3) Pruebe con el comando **last** para desplegar el record de quienes han ingresado al sistema, con el comando **tac** reverse el orden. Para que fuese esto útil? Si la salida es extensa como la direcciona al comando **less**?

4) Use **sed** para corregir el error ortográfico ‘sostema’ a ‘sistema’. Escriba un pequeño archivo en **nano**, para probar su comando. Que pasa si el error ocurre más de una ves, y que se puede hacer?

5) Use **nl** para enumerar las líneas que escribió en el ejemplo de arriba para corregir el error.

Capítulo4

Es la pregunta que nos impulsa NEO. Es la duda que te ha traído aquí.

Trinity, The Matrix

Manejo de Archivos de Texto

Los Objetivos de este Capítulo son:

1. **Editores de Textos Básicos, vi**
2. **Conceptos de Entrada/Salida**
3. **Redirección y Tuberías**
4. **Filtros y comandos de manipulación de texto**
5. **Sortear y ordenar**
6. **Cortar y pegar desde la línea de comandos**
7. **Manipulación básica como copiar, mover, de archivos**

Trabajar con Archivos de Texto

- Sistemas tipo Unix son muy eficiente en su manejo de archivos de texto
- Las mismas técnicas pueden ser utilizadas con archivos de texto simple o basado en formato de texto
  - La mayoría de los archivos de configuración de Unix son archivos de texto simple.
- El texto es mayormente en caracteres **ASCII**
  - Texto en idiomas otros que el Inglés pueden usar los caracteres ISO-8859
- Unicode es mejor, pero desafortunadamente muchas utilidades de GNU/Linux de línea de comandos no lo soportan aún.

Líneas de Texto

- Archivos de Texto son divididos en líneas
- En GNU/Linux una línea termina en un carácter de **line-feed**
  - Carácter número 10, hexadecimal 0x0A
- Otros sistemas operativos utilizan diferentes combinaciones
  - Windows y DOS usan el retorno del carro seguido por un line-feed
  - Sistemas Macintosh usan sólo un retorno de carro (Carriage-Return)
  - Existen programas que convierten entre estos formatos

Filtrar Texto y Tuberías

- La filosofía Unix es: usar pequeños programas, y combinarlos cuando sean necesarios
- Cada herramienta debe ser excelente en lograr un objetivo
- Se combinan programas utilizando las tuberías
  - El programa en la izquierda imprime texto a la **salida estándar**
  - Esa salida de texto es alimentada a la **entrada estándar** al segundo programa a la derecha
- Por ejemplo, para conectar la salida de **echo** a la entrada de **wc**:

```
$ echo "vamos a contar palabras" | wc
```

Desplegar Archivos con less o more

- Si un archivo no cabe en el espacio del terminal, despliegue con **less** o **more**:

```
$ less README
```

$ more README

- Además con less también es más fácil limpiar el terminal de todas las cosas allí presente
- Muy a menudo son utilizados a la derecha de las tuberías, para asistir en la lectura de archivos largos:
  $ wc *.txt | less **$ wc *.txt | more**
- No se bloquean con caracteres no interpretables como lo hace **cat**, y así no pierdes el terminal

Contar Palabras y Líneas con wc

- El comando **wc** cuenta caracteres, palabras y líneas en un archivo
- Si se utiliza para múltiples archivos genera su salida de cada archivo y un total combinado de todos
- Opciones:
  - **-c** salida cuenta los caracteres
  - **-l** salida cuenta las líneas
  - **-w** salida cuenta las palabras
  - Opción por defecto es: **-clw** salida es contar caracteres, líneas y palabras
- Por ejemplo: para desplegar el número de palabras en el archivo carta.txt:
  $ wc -w carta.txt
- Para desplegar el total número de líneas en todos los archivos de extensión txt en directorio **trabajos/** :
  $ wc -l trabajos/*.txt

Sortear Líneas de Texto con sort

El comando *sort* ordena líneas o columnas de un archivo en orden alfabética, numérica o orden reversa. Esto es otro de los comandos que debe manejar para incluirlo en su caja de herramientas para desempeñar su papel como administrador de sistemas *NIX. Sort es un comando muy versátil y poderoso; pero, si puede ser un poco difícil de aprender a un principio. Para que sea un poco menos difícil, lo vamos a dividir en tres categorías las funciones que sort puede llevar a cabo y entonces después de dividirlo lo conquistaremos.

- **Sort (ordenar) Simple.** Ordena las líneas de un archivo en orden alfabética, numérica o orden reversa.
- **Sort (ordenar) columnas.** Ordena usando uno o más de un campo separado en columnas. El orden del sorteado de cada columna puede ser especificado individualmente.
- **Fusionar archivos.** Pueden (pre-ordenado y sin ordenar) ser fusionado con el comando sort.

Sorteos Simples

sort [opciones] [*Archivos...*]

El comando *sort* ordena uno o más archivos en orden alfabética, numérica o orden reversa. Por defecto es ordenar alfabéticamente. Por ejemplo,

$ cat Archivo.txt	**$ sort Archivo.txt**
Susana	**Elizabeth**
Elizabeth	**Juan**
Juan	**Michael**
Michael	**Susana**

Si no se especifica un archivo, el comando *sort* lee desde la entrada estandar. Las opciones de la linea de

comandos que necesita manejar se muestran en la siguiente tabla:

Opciones General

Opción	Descripción
-o *filename*	Escribe la salida a un archivo, de nombre *filename*. Si no se especifica ningún archivo, la salida se envía a la salida estándar.
-u	(única) Líneas idénticas de entra se da salida solo una vez.
-c	Revisa a ver si los archivos ya están ordenados. Si esta ya sorteado, este no produce salida. Si no esta en orden este envía un mensaje de error a la salida del error estándar.

Opciones Orden de sort

Opción	Descripción
-d	Sortea en orden de diccionario. Ignora todos los caracteres excepto las letras, dígitos y líneas en blanco al determinar el orden del sorteado.
-n	Sortear en orden numérica (Por ejemplo: -2.5, -1, 0, 0.54, 3, 18). Orden Numérica ignora espacios en blanco al determinar el orden del sorteado e interpreta un símbolo de números negativos (-) correctamente. Números pueden incluir comas para separar los miles, millones, etc (e.j. 1,000 or 10,000). Entradas no-numéricas son sorteadas en orden alfabética entre números de cero y positivos. Líneas en blanco son sorteadas entre números negativos y cero. *Sort no* interpreta el símbolo de más (+) como número positivo, pero si al principio de una entrada no-numérica.
-f	Ignora distinción de mayúscula/minúsculas. (a y A son lo mismo).
-M	Ordenar los primeros tres caracteres como los meses. (e.j. jan < feb < mar...). Letras mayúsculas preceden las minúsculas del mismo mes (e.j. JAN < Jan < jan < FEB) Nombres no validos son sorteados en orden alfabética antes de los nombres validos. (e.j. mal-escritos < no-es-un-mes < jan).
-i	Ignorar los caracteres que no se imprimen. Los caracteres que no se imprimen incluyen caracteres de control como lo son tab, avance del carro, retorno del carro, etc. Caracteres no imprimibles son esos que no se incluyen el rango de los ASCII 040-176.
-r	Reversar el orden del sorteado.

Ejemplos Simples de sort

Sortear en Orden alfabética y de Diccionario

Los próximos ejemplos usan *Archivo1*.

```
$ cat Archivo1
.esta línea empieza con un punto
a esta línea la empezamos con minúscula a.
Esta es una línea.
abracadabra
1234
Donde ordenara esta línea?
  A esta línea la empezamos con mayúscula a.
```

- Sortear las líneas del *Archivo1* en orden alfabética.

```
$ sort Archivo1
.esta línea empieza con un punto
1234
A esta línea la empezamos con mayúscula a.
Esta es una línea.
Donde ordenara esta línea?
a esta línea la empezamos con minúscula a.
abracadabra
```

Note que los espacios y los caracteres de puntuaciones son ordenados primero antes de los números seguidos por las mayúsculas de la A a la Z entonces las minúsculas de la a a la z.

- Sortee el *Archivo1* en orden alfabética y escriba la salida a un archivo de nombre *Archivo1s*.

```
$ sort -o Archivo1s Archivo1
```

- Sortee *Archivo1* en orden alfabética inversa.

```
$ sort -r Archivo1
abracadabra
a esta línea la empezamos con minúscula a.
Donde ordenara esta línea?
Esta es una línea.
A esta línea la empezamos con mayúscula a.
1234
.esta línea empieza con un punto
```

- Sortee *Archivo1* en orden alfabética ignorando las mayúsculas/minúsculas.

```
$ sort -f Archivo1
.esta línea empieza con un punto
1234
a esta línea la empezamos con minúscula a.
```

A esta línea la empezamos con mayúscula a.
abracadabra
Esta es una línea.
Donde ordenara esta línea?

- Sortee *Archivo1* en orden de diccionario.
  $ sort -d Archivo1
  1234
  A esta línea la empezamos con mayúscula a.
  Esta es una línea.
  Donde ordenara esta línea?
  a esta línea la empezamos con minúscula a.
  abracadabra
  .esta línea empieza con un punto

  Orden de diccionario ignora todos los caracteres excepto los números, letras y espacios en blanco axial que "*.esta línea empieza con un punto*" es sorteada idéntica que si fuese sin el punto axial "*esta línea empieza con un punto*".

- Sortee *Archivo1* en orden de diccionario, ignore las mayúsculas/minúscula.
  $ sort -df Archivo1
  1234
  a esta línea la empezamos con minúscula a.
  A esta línea la empezamos con mayúscula a.
  abracadabra
  Esta es una línea.
  .esta línea empieza con un punto
  Donde ordenara esta línea?

- Sortee *Archivo1* en orden de diccionario inversa, ignore las mayúsculas/minúscula. .
  $ sort -dfr Archivo1
  Donde ordenara esta línea?
  .esta línea empieza con un punto
  Esta es una línea.
  abracadabra
  A esta línea la empezamos con mayúscula a.
  a esta línea la empezamos con minúscula a.
  1234

  Esta sentencia es el total inverso a usar el comando "*sort -df Archivo1*".

Sortear en Orden Numérica

Los próximo dos ejemplos usaran el archivo *Numero1*.

```
$ cat Numero1
-18
18
0
-1.4
0.54
0.0
3
0.1
```

- Sortee el archivo *Numero1* en orden numérica.

```
$ sort -n Numero1
-18
-1.4
0
0.0
0.1
0.54
3
18
```

- Sortee *Numero1* en orden alfabética.

```
$ sort Numero1
0
0.0
0.54
3
-1.4
18
-18
0.1
```

Note que esto no es sorteado matemáticamente.

- Por cada archivo en el directorio actual, liste el número de líneas en el archivo. Sortee los archivos en orden descendente de más líneas a menos.

```
$ wc -l * | sort -rn
```

El comando wc imprime el número de líneas en un archivo. La salida del comando *wc* se pasa por la tubería al comando *sort* donde la opción *-n* ordena los números de pequeño a grande, pero la opción *-r invierte el orden de sorteo*, ordenando los números de mayor a menor.

Sortear Meses

El próximo ejemplo usa el archivo *Meses*.

```
$cat Meses
FEB
mal-escrito
mar
MAY
january
May
No-mes
jan
may
```

Use la opción *-M* para sortear *Meses* en orden cronológica.

```
$ sort -M Meses
mal-escrito
no-mes
jan
january
FEB
mar
MAY
May
may
```

Note que los que no son meses son ordenados de primero y que los que tienen letras mayúsculas presiden los de letras minúsculas aunque sean meses idénticos.

Sortear con la opción de Única

El próximo ejemplo usa el archivo *log-de-error*.

```
$ cat log-de-error
error 01: /tmp directory not found
error 17: out of memory
error 01: /tmp directory not found
error 22: low disk space
error 01: /tmp directory not found
```

1. Sortee el archivo *log-de-error* en orden alfabética. líneas de entrada idénticas son interpretadas solo una vez.

$ sort -u log-de-error
error 01: /tmp directory not found
error 17: out of memory
error 22: low disk space

Sortear con la opción de Revisar/Check

Sortear un archivo grande puede ser una actividad extremadamente lenta. Irónicamente, es más lento ejecutar el comando sort en un archivo ya sorteado que en uno que no esta. La opción -c revisa para verificar que el archivo no este ya sorteado en un orden especifico. Si retorna que ya esta sorteado, el sort hace absolutamente nada. Si no esta, sort imprime un mensaje de error al error estándar. Por ejemplo, asumiendo que el archivo alfab-sorteado como su nombre indica ya esta ordenado alfabéticamente.

$ sort -c alfab-sorteado
Este proceso es mucho más rápido que sortear un archivo que ya esta correctamente sorteado.

- Asumamos que no esta sorteado en orden alfabética.
  $ sort -c no-sorteado
  sort: disorder on no-sorteado

- Ahora experimente con el comando.
  $ sort -fc alfab-sorteado
  sort: disorder on alfab-sorteado

La opción *-f* le instruye a *sort* que ignore la distinción de las letras mayúsculas/minúsculas. El archivo de nombre *alfab-sorteado* esta sorteado en un orden regular alfabético con distinción de mayúscula/minúscula tomada en cuenta axial que, la opción *-c* reporta desorden.

Sortear columnas con sort

Sort puede ordenar archivos por columnas (también llamados campos). Por ejemplo, el archivo *Archivo1* tiene dos campos, nombre y apellido.

$ cat Archivo1
Susana Perez
Jinette Diaz
John Foster
Andres Carter

El siguiente comando, ordena el *Archivo1* por el segundo campo.

$ sort -k 2 Archivo1
Andres Carter
Jinette Diaz
John Foster
Susana Perez

Las opciones de la línea de comando que debe saber para sortear archivos por columnas son mostradas más adelante. Estas opciones deben ser usadas después de las opciones generales y de orden de sorteo.

Opciones de Sortear Columna

opción	Descripción
-t*c*	Especifica el carácter, *c*, que separa los campos. Por ejemplo, "*-t,*" indica que los campos son separados por comas. Cada ocurrencia de *c* es significante axial que *cc* representa un campo vacío. Por ejemplo, si el carácter separador es una coma entonces en "a,,d" el campo uno la entrada es "a", campo dos esta vacío y campo tres es "d". El separador por defecto es espacio en blanco.
-b	Ignorar espacios en blanco (espacios y tabs) al determinar el carácter de las columnas. Cuando usamos espacio en blanco para separar las columnas, la opción *-b* elimina el significado de múltiple separadores de columnas. Por ejemplo, "c" es el primer carácter de el segundo campo "ab<space>cd" "ab<space><space>cd" y "ab<space><space><space>cd"
-k *START*[,*END*]	Define una clave de sorteado o una sección de cada línea usada para ordenar. La clave de sortear empezara con el campo S*TART* y termina con el campo *END*. Si END no se especifica, la clave empieza con START y termina hasta en fin de la línea. Más detalles de como especificar la llave a continuación.

Especificar la llave de Sort para la opción -k

El START y *END* son especificada usando el formato de *FNum*[.*CNum*][*tipo*] donde *FNum* es el campo número, empezando desde el 1, y *CNum*, si presente, es el carácter dentro del campo. El modificador tipo es descrito más adelante. Por ejemplo,

-k 1	Empezando por el primer carácter del primer campo y continuar hasta el final de la línea. Este es método es el mismo que el sort simple.
-k 1,1	Por el primer campo solamente. El ordenamiento de las líneas con primeros campos idénticos no es especificado (al azar/random). Note que esto es diferente al ejemplo anterior.
-k 1,3	Empezando con el primer carácter del primer campo y terminando con el último carácter del tercer campo.
-k 1.2	Empezando desde el segundo carácter en el primer campo y continuar hasta el fin de la línea.
-k 1.3,3.3	Empezando con el 3er carácter en el primer campo y terminar con el 3rd carácter en el 3er campo.

Cualquier número de especificación de campo puede ser usada con sort. Por ejemplo,

-k 3,5 -k 2,2	Sortea por el campo tres hasta el cinco y luego el dos.
-k 1,1 -k 2,2 -k 3,3	Sortea por el campo uno. Si el campo uno es idéntico, sortea por el campo dos. Si el campo uno y el dos son idénticos, sortea por el tres.
-k 1,3	Sortear por el campo uno hasta el tres. Note que este es diferente al ejemplo anterior.

Modificadores de Tipo

Un modificador de tipo puede ser agregado a un *START* o *END* para cambiar el orden de sorteo por defecto de la llave de sortear. Los modificadores de tipos son uno o más de las siguientes letras: d, f, i, M, n, o r. El efecto es el mismo como el correspondiente a la opción de orden de sortear (*-d*, *-f*, etc.) excepto que solamente el ordenamiento especificado por la llave de sortear después del *-k* es afectado. Estos modificadores de tipos pueden ser aplicados a START, *END* o ambos. El efecto es el mismo. Por ejemplo,

-k 1n	Sortear por la línea completa (campo uno hasta el final) usando orden de sorteo numérica.
-n -k 1	Idéntico al ejemplo anterior.
-n -k 3,3 -k 1,1	Sortear por el tercer campo. Si el tercer campo es idéntico, sortear por el primer campo. Ambos sorteos son en orden numérico.
-k 3,3n -k 1,1n	Idéntico al ejemplo anterior.
-k 3n,3 -k 1n,1	Idéntico al ejemplo anterior.
-k 3n,3n -k 1n,1n	Idéntico al ejemplo anterior.
-k 3,3n -k 1,1	Sortear por el tercer campo usando ordenamiento numérico. Si el tercer campo es idéntico, sortea por el primer campo usando el orden alfabético por defecto.

Una vez el modificador de tipo es parte de una especificación de llave de sortear, otras opciones de sortear simple son ignoradas por esa llave de sortear. Por ejemplo,

-df -k 2,2	Sortear por el campo dos en orden de diccionario ignorando la distinción de mayúscula/minúscula.
-k 2,2df	Igual que el ejemplo anterior.
-f -k 2,2d	Sortear por el campo dos en orden de diccionario, pero no aplica la opción *-f* a la llave de sortear *-k 2,2d*. Caso sensitiva cuando ordene por el segundo campo.
-f -k 2,2df -k 3,3	Sortear por el campo dos en orden de diccionario ignorando la distinción de mayúscula/minúscula. Si el campo dos es idéntico, sortear por el campo tres ignorando la distinción de mayúscula/minúscula, pero usando el sorteo por defecto alfabético y no el sorteo de diccionario.

El modificador de tipo b, como la opción *-b*, causa al comando *sort* que ignore caracteres en blanco al determinar posiciones de campo y carácter. No como otros modificadores de tipo, l modificador b afecta a *START* y *END por separado*.

-b -k 2,3	Sortear por el campo dos hasta el tres. Ignorar los espacios en blanco al principio al determinar el primer carácter del campo dos y el campo tres.

-k 2,3bd	Sortear por el campo dos hasta el tres ambos en orden de diccionario. Espacios en blanco al principio será ignorado al determinar cual es el carácter al principio del campo tres pero no el del campo dos. Probablemente no sea esto que usted desee.
-k 2b,3bd	Sortear por el campo dos entonces por el tres y ambos en orden de diccionario. Ignorar espacios en blanco al principio al determinar el primer carácter del campo dos y el tres.

Más en como especificar *CNum*

Al contar caracteres de campo, el comando *sort* es sensitivo a la presencia de números y tipo de caracteres de separación usados entre los campos. Generalmente, empezara a contar caracteres en un campo después de haber llegado al primer carácter separador especificado por la opción *-t*. Esto tiene sentido si se usa un separador de campo como es la coma. Por ejemplo,

col1fila1,12345678
col2fila2,abcdefgh

-t, -k 2.2,2.4	El sorteo empieza con el carácter "2" en fila número uno y "b" en la fila dos y termina con el carácter "4" en la fila uno y "d" en la fila two.

En el siguiente ejemplo, existe un espacio entre el separador de campo, a coma, y la data útil en el campo dos.

col1fila1, 12345678
col2fila2, abcdefgh

-t, -k 2.3	El sorteo empieza con los caracteres "2" y "b". Los primeros caracteres son el espacio en blanco después de la coma, los segundos caracteres son "1" y "a".
-t, -k 2.2b	Idéntico al ejemplo anterior. El modificador b ignora el espacio en blanco al principio así que los primeros caracteres son "1" y "a".

Es más confuso cuando se usa espacio para separar las columnas. Cuando no se especifica un separador de campo explícitamente con la opción *-t*, el comando *sort* usara cualquier espacio en blanco como separador de campo. También contara este espacio en blanco como un carácter en el próximo campo. Por ejemplo, digamos que tenemos un archivo con dos columnas separadas por un space.

col1fila1 12345678
col1fila2 abcdefgh

-t" " -k 2.1	El sorteo empieza con los caracteres "1" y "a". Porque el espacio fue explícitamente especificado como un separador de campo, *sort* inicia contando caracteres de campo después de este.

-k 2.2	El sorteo empieza con los caracteres "1" y "a". Porque el espacio no fue explícitamente especificado como un separador de campo, *sort* cuenta el espacio que separa los campos uno y dos como el primer carácter del campo dos, aunque el espacio es un separador de campo por defecto y no afectara el orden del sorteo.
-k 2.1b	El sorteo empieza con los caracteres "1" y "a". Como vimos en el último ejemplo, *sort* normalmente cuenta el espacio en blanco separador como un carácter de campo; pero, el modificador de tipo b le dice que no es de incluir el espacio en blanco de alante al contar los caracteres.

Ejemplo de sortear columnas

Los próximos ejemplos usan el archivo ***notas***, el cual contiene la fecha (mes, día, año), nombre del estudiante, apellido y calificación del examen.

```
$ cat notas
Dec 30 2005 Foster roberto 92
Dec 30 2005 Lopez Karen 83
Dec 30 2005 Foster John 78
Dec 30 2005 Rodriguez Sara 85
Feb 4 2006 Foster Roberto 84
Feb 4 2006 Foster John 92
Feb 4 2006 Rodriguez Sara 91
Feb 4 2006 Lopez Karen 72
```

- Sortear *notas* poniendo el apellido del estudiante (4^{to} campo) en orden alfabética.

```
$ sort -k 4 notas
Feb 4 2006 Lopez Karen 72
Dec 30 2005 Lopez Karen 83
Dec 30 2005 Rodriguez Sara 85
Feb 4 2006 Rodriguez Sara 91
Feb 4 2006 Foster Roberto 84
Dec 30 2005 Foster Roberto 92
Dec 30 2005 Foster John 78
Feb 4 2006 Foster John 92
```

Como no se especifico el campo final, el archivo se ordena empezando con la 4^{ta} columna y terminando con la columna final. Así que, el nombre y la calificación son incluidas en el sorteo. Si usamos el siguiente comando

```
$ sort -k 4,4 notas
Dec 30 2005 Lopez Karen 83
Feb 4 2006 Lopez Karen 72
Dec 30 2005 Rodriguez Sara 85
Feb 4 2006 Rodriguez Sara 91

Dec 30 2005 Foster Roberto 92
Dec 30 2005 Foster John 78
Feb 4 2006 Foster Roberto 84
```

Feb 4 2006 Foster John 92

Solamente la 4ta columna es usada en el sorteo. La salida no es sorteada por nombre o calificaciones.

- *Ordena el archivo notas colocando los apellidos de los estudiantes* (4to campo) en orden alfabética. Escribir la salida a un archivo de nombre *notas.final*.

  $ sort -o notas.final -k 4 notas
- Ordene el archivo *notas* en orden descendente de las calificaciones.

  $ sort -nr -k 6,6 notas
  Feb 4 2006 Foster John 92
  Dec 30 2005 Foster Roberto 92
  Feb 4 2006 Rodriguez Sara 91
  Dec 30 2005 Rodriguez Sara 85
  Feb 4 2006 Foster Roberto 84
  Dec 30 2005 Lopez Karen 83
  Dec 30 2005 Foster John 78
  Feb 4 2006 Lopez Karen 72

La opción *-k 6,6* ordena por la 6ta columna. La opción *-n* ordena en orden numérica (ascendente) y la opción *-r* invierte el orden (ascendente). La siguiente sentencia es equivalente.

$ sort -k 6,6nr notas

- Ordene el archivo notas por nombre de estudiante, apellido y nombre, y entonces la fecha del examen, año seguido por el mes y finalmente por el día.

  $ sort -k 4,5 -k 3,3n -k 1,1M -k 2,2n notas
  Dec 30 2005 Lopez Karen 83
  Feb 4 2006 Lopez Karen 72
  Dec 30 2005 Rodriguez Sara 85
  Feb 4 2006 Rodriguez Sara 91
  Dec 30 2005 Foster Roberto 92
  Feb 4 2006 Foster Roberto 84
  Dec 30 2005 Foster John 78
  Feb 4 2006 Foster John 92

Note que los nombres están ordenados en orden alfabética, el año y el día están sorteados en orden numérica y los meses en orden cronológicamente como meses.

Guarde la salida de la sentencia arriba ejecutada a un archivo de nombre *notas2*.

$ sort -o notas2 -k 4,5 -k 3,3n -k 1,1M -k 2,2n notas

Use la opción (*-c*) para determinar si los archivos *notas* o *notas2* están ya ordenados por los nombres de los estudiantes o las fechas de los exámenes.

$ sort -c -k 4,5 -k 3,3n -k 1,1M -k 2,2n notas
sort: disorder on notas
$ sort -c -k 4,5 -k 3,3n -k 1,1M -k 2,2n notas2

Los próximos ejemplos usan *Archivo1*.

```
$ cat Archivo1
.esta línea empieza con un punto
a esta línea la empezamos con minúscula a.
   Esta línea empieza con un espacio.
abracadabra
1234
Donde ordenara esta línea?
A esta línea la empezamos con mayúscula a.
```

- Ordene el *Archivo1* en orden de diccionario ignorando las mayúsculas/minúsculas y los espacios delanteros en blanco.

```
$ sort -dfb -k 1 Archivo1
1234
a esta línea la empezamos con minúscula a.
A esta línea la empezamos con mayúscula a.
abracadabra
.esta línea empieza con un punto
   Esta línea empieza con un espacio.
Donde ordenara esta línea?
```

Esto es un truco para lograr un sorteo simple ignorando los caracteres en blanco delanteros requeridos porque la opción *-b* solamente afecta ordenar por columna. Pero, en algunos sistemas la opción *-b* afectara algunos sorteados simples también. En estos sistemas el siguiente comando es idéntico.

```
$ sort -dfb Archivo1
```

Los próximos ejemplos usaran el archivo *números*, el cual usa un carácter de dos puntos (:) como separador de campo.

```
$ cat números
3:18
12:5
 3:22
8: 5
12:5
```

- Sortee *números* en orden numérica por el campo uno. Si el campo uno ya esta sorteado, sortee por el campo dos. Use un dos punto como separador de campo.

```
$ sort -n -t":" -k 1,1 -k 2,2 números
3:18
 3:22
8: 5
12:5
```

12:5

Repita el mismo sorteo usando la opción única (*-u*). Líneas idénticas de entrada se envían a la salida solo una vez.

```
$ sort -un -t":" -k 1,1 -k 2,2 números
3:18
 3:22
8: 5
12:5
```

- Trate este comando para probar.

```
$ sort -n -t":" -k 1,2 números
 3:22
3:18
8: 5
12:5
12:5
```

No le sorprende que " *3:22*" vienen antes de el "*3:18*"? Esto ocurre porque "*-k 1,2*" combina los campos uno y el dos antes de ordenar creando dos cadenas de texto (strings) " *3:22*" y "*3:18*". Estas no son reconocidas como números así que ellas son sorteadas en orden alfabética aun con la opción *-n*. Porque existe un espacio en blanco en frente de " *3:22*", es ordenado primero en un orden alfabético.

Ejemplos Avanzados: Ordenar Columnas

1. El próximo ejemplo usara el archivo *jnombres*. *Jnombres* contiene nombre, apellido y la inicial de su segundo nombre justificado usando espacios.

```
$ cat Jnombres
Mike    Foster    C
TJ      Diaz      R
Sampson Elliot    T
tj      Meyers    D
Antonio Foster    A
```

2. Sortee *Jnombres* por el segundo hasta el tercer campo, apellido y la inicial del segundo nombre. Ignoremos la distinción de mayúscula/minúscula.

```
$ sort -k 2,3f Jnombres
tj      Meyers    D
TJ      Diaz      R
Mike    Foster    C
Antonio Foster    A
Sampson Elliot    T
```

Porque es que *Diaz* esta ordenado antes que *Foster*? Porque nosotros no le dijimos a a sort que ignore los espacios en blanco delanteros. Entonces sort esta ordenando así "<space><space><space><space><space>Diaz"

Antes "<space><space><space><space>Foster" porque Diaz tiene más espacios en blanco. Para corregir este problema use el modificador de tipo -b.

```
$ sort -k 2b,3bf Jnombres
Sampson     Elliot      T
Tj          Meyers      D
Antonio     Foster      A
Mike        Foster      C
TJ          Diaz        R
```

El siguiente comando también puede ser un poco problemático.

$ sort -b -k 2,3f Jnombres

Porque el modificador de tipo f es agregado a las especificaciones de la llave "*-k 2,3f*", *sort* no aplica la opción *-b* el comando sort ordena las opciones al sortear con esa llave.

$ sort -k 2,3bf Jnombres

Porque el modificador de tipo solo afectara el campo tres.

Los próximo dos ejemplos usan el archivo ***Fechas***, el cual contiene especificaciones del día de la semana en el primer campo seguido por las especificaciones del tiempo en formato de hora:minuto:segundo en el campo dos. Los campos están separados por un **TAB**.

```
$ cat Fechas
Wed    02:43:55
Tue    14:46:32
Wed      11:43:13
```

3. Ordene el archivo *Fechas* por los minutos.

```
$ sort -k 2.4b,2.5bn Fechas
Wed    02:43:55
Wed    11:43:13
Tue    14:46:32
```

Note: Estamos usando el modificador de tipo -b para que no se incluyan los espacios en blanco al contar la posición de los carácter.

4. Ordene el archivo *Fechas* por minutos seguido por los segundos.

```
$ sort -k 2.4b,2.5bn -k 2.7b,2.8bn Fechas
Wed    11:43:13
Wed    02:43:55
Tue    14:46:32
```

Los próximo dos ejemplos usan el archivo *Jovenes*, el cual contiene los nombres, apellido y edad de tres Jovencitos.

```
$ cat Jovenes
nombre apellido Edad
Susana      Perez      6
Elizabeth  Diaz      11
Michael    Reyes      8
```

La primera línea del archivo los nombres de las columnas en vez de data.

5. Ordene la data del archivo *Jovenes* por el campo edad. Ignorando la columna nombres.

```
$ tail +2 Jovenes | sort -k 3n
Susana      Perez      6
Michael    Reyes      8
Elizabeth  Diaz      11
```

El comando “tail *+2 Jovenes*" imprime el contenido del archivo *Jovenes*, empezando por la segunda línea, a la salida estándar. La tubería (pipe **|**) redirecciona la salida del comando *tail* a la entrada del comando *sort*, el cual entonces ordena por el campo número tres en orden numérica.

6. Cree un archivo, *sJovenes*, que contenga la data del archivo *Jovenes* ordenada por edad. Incluya la columna nombres en la parte superior del archivo *sJovenes* pero no la incluya en el sorteado.

   $ (head -1 Jovenes ; tail +2 Jovenes | sort -k 3n) > sJovenes

Como es que esto funciona? El punto y coma (;) usado para poder escribir dos comandos en una misma línea de comandos. El primer comando, “head *-1 Jovenes*", imprime la primera línea del archivo *Jovenes*, la columna nombres, a la salida estándar. El segundo comando "*tail +2 Jovenes | sort -k 3n*" ordena la data en el archivo *Jovenes* por el campo edad y imprime el resultado a la salida estándar. Los paréntesis son usados para ejecutar ambos comandos en un solo subshell así que la salida pueda ser redireccionada simultáneamente al archivo *sJovenes*.

Fusionando/Merging

Los archivos (pre-ordenados y no-ordenados) pueden ser fusionados con el comando sort. Por ejemplo, asumamos que tenemos estos dos archivos, *Archivo1* y *Archivo2*,

$ sort -o sArchivoo1 Archivo2

Fusiona los archivos *Archivo1* y *Archivo2*, los pone en orden y entonces almacena la salida al archivo *sArchivo*. Es el equivalente de ejecutar la siguiente sentencia.

$ cat Archivo1 Archivo2 > Archivo3
$ sort -o sArchivo Archivo3
$ rm Archivo3

Una opción de la línea de comandos que afecta la fusión de los archivos.

Opción	Descripción
-m	Solamente fusionar. Para usar en archivos que han sido sorteados previamente.

Por ejemplo, si dos archivos Archivo1s y Archivo2s ya han sido sorteados

$ sort Archivo1 -o Archivo1s
$ sort Archivo2 -o Archivo2s

Entonces

$ sort -m -o sArchivo Archivo1s Archivo2s

Nos ahorra tiempo no teniendo que reordenar los archivos *Archivo1s* y *Archivo2s*. Solo tenemos que integrarlos.

Ejemplos: Fusionar/Merging

Los siguiente dos ejemplos usan los archivos *Datos1* y *Datos2*.

$ cat Datos1	$ cat Datos2
A - desde el archivo 1 C - desde el archivo 1 E - desde el archivo 1	B - desde el archivo 2 D - desde el archivo 2

1. Fusione los archivos *Datos1* y *Datos2* en orden alfabética.

$ sort -m Datos1 Datos2
A - desde el archivo 1
B - desde el archivo 2
C - desde el archivo 1
D - desde el archivo 2
E - desde el archivo 1

La opción *-m* (solamente fusiona) es apropiada porque los archivos *Datos1* y *Datos2* ya están sorteados y en orden alfabética.

2. Usando la opción *-m* con un archivo que no este ya ordenado en el orden correcto arrojara un resultado desordenado. Por ejemplo, fusione los archivos *Datos1* y *Datos2* en orden inversa alfabética.

$ sort -r -m Datos1 Datos2
B - desde el archivo 2
D - desde el archivo 2
A - desde el archivo 1
C - desde el archivo 1
E - desde el archivo 1

Podemos usar la opción *-c* para revisar si un archivo esta sorteado en orden correcta antes de decidir usar la opción *-m*.

$ sort -c -r Datos1 ; sort -c -r Datos2
sort: disorder on Datos1
sort: disorder on Datos2

Ya que la opción de que revisara los archivos retorno un resultado de desorden, los archivos tendrán que ser ordenados además de fusionados.

$ sort -r Datos1 Datos2
E - desde el archivo 1
D - desde el archivo 2
C - desde el archivo 1
B - desde el archivo 2
A - desde el archivo 1

El siguiente ejemplo usa los archivos *Archivo1* y *Archivo2*.

$ cat Archivo1	**$ cat Archivo2**
.esta línea empieza con un punto a esta línea la empezamos con minúscula a.	Esta es una línea. abracadabra

¿Donde ordenara esta línea? A esta línea la empezamos con mayúscula a.	1234

3. Fusione y sortee los archivos *Archivo1* y *Archivo2* en orden de diccionario, ignorando la distinción entre mayúsculas/minúscula.

   $ sort -fd Archivo1 Archivo2
   1234
   a esta línea la empezamos con minúscula a.
   A esta línea la empezamos con mayúscula a.
   abracadabra
   Esta es una línea.
   .esta línea empieza con un punto
   ¿Donde ordenara esta línea?

Dos maneras equivalentes de fusionar y ordenar estos archivos son:

$ cat Archivo1 Archivo2 > Archivo3
$ sort -fd Archivo3
y
$ sort -fd -o Archivo1s Archivo1
$ sort -fd -o Archivo2s Archivo2
$ sort -fdm Archivo1s Archivo2s

4. En los ejemplos anteriores solo hemos fusionado dos archivos; pero podemos fusionar más de dos archivos con el comando *sort*.

   $ sort Archivo1 Archivo2 Archivo3 Archivo4 Archivo5 ...

Los siguientes dos ejemplos usan los archivos *Calificaciones1* y *Calificaciones2*.

$ cat Calificaciones1	**$ cat Calificaciones2**
Foster Roberto 92	Foster Roberto 84
Lopez Karen 83	Foster John 92
Foster John 78	Rodriguez Sara 91
Rodriguez Sara 85	Lopez Karen 72

Calificaciones1 contiene tres campos, apellido, nombre y la calificación del primer examen del año. *Calificaciones2* contiene la misma data pero del segundo examen del año.

5. Ordene y fusione los archivos *Calificaciones1* y *Calificaciones2* por nombre.

   $ sort -k 1,2 Calificaciones1 Calificaciones2
   Lopez Karen 72
   Lopez Karen 83
   Rodriguez Sara 85
   Rodriguez Sara 91
   Foster Roberto 84

Foster Roberto 92
Foster John 78
Foster John 92

6. Puede ser que desee producir una salida que contenga una línea por estudiante con ambas calificaciones, por ejemplo

   Lopez Karen 83 72
   Rodriguez Sara 85 91
   Foster Roberto 92 84
   Foster John 78 92

Este tipo de fusión no lo provee el comando *sort*. Para este ejemplo, tendrás que utilizar el comando *sort* asistido por los comandos cut y paste.

$ sort -o sCalificaciones1 -k 1,2 Calificaciones1
$ sort -o sCalificaciones2 -k 1,2 Calificaciones2
$ cut -d" " -f 3 sCalificaciones2 | paste sCalificaciones1 -

El comando join provee una manera más avanzada de fusionar por columnas.

El comando uniq

El comando *uniq* remueve o elimina líneas duplicadas de un archivo. Es comúnmente utilizado como parte de un filtro.

Descripción

uniq [options] ***file1 file2***
uniq [opciones] Archivo1 Archivo2

Uniq elimina las líneas duplicadas en Archivo1 y escribe una línea única a Archivo2. Si Archivo2 existe, uniq sobrescribe este archivo sin dar ninguna advertencia. Si Archivo2 no se especifica, uniq escribe a la salida estándar. Si no se especifica Archivo1, uniq lee desde la entrada estándar. Por ejemplo,

$ cat frutas
manzanas
manzanas
naranjas
peras

$ uniq frutas
manzanas
naranjas
peras

Uniq es solamente útil si el archivo ha sido previamente ordenado. En el siguiente ejemplo se deja demostrado

$ cat frutas
manzanas

```
naranjas
manzanas

$ uniq frutas
manzanas
naranjas
manzanas
```

En este caso uniq no removió la segunda línea de manzanas porque no estaba inmediatamente después de la primera línea de manzanas.

Opciones

Opción	Descripción
-c	Escribe el número de veces que una línea ocurre en el archivo de entrada antes de cada línea del archivo de salida.
-d	Escribe cada línea duplicada una sola vez pero no envía a la salida líneas únicas.
-u	Escribe solamente líneas únicas. Todas las duplicadas son eliminadas.
-f *n*	Ignore los primeros *n* campos de una línea. Los campos son delimitados por espacios o tabs.
-s n	Ignora comparar los primeros n caracteres

Nota: Las opciones -c, -d y -u no se pueden usar juntas.

Ejemplos

1. Escriba una copia única de las líneas únicas del *Archivo1* en *Archivo2*.

   $ uniq Archivo1 Archivo2

   Mucho cuidado! Si existe el *Archivo2* este será sobre escrito por el comando *uniq, sin emitir ninguna advertencia*.

Los próximos ejemplos usaran el archivo *log-error*.

```
$ cat log-error
error 11: /tmp directory not found
error 22: out of memory
error 11: /tmp directory not found
error 17: low disk space
error 11: /tmp directory not found
error 22: out of memory
error 04: connection failure
error 11: /tmp directory not found
```

2. El primer paso es ordenar el archivo *log-error*. Esto se puede lograr usando el comando sort y guardando la salida en el archivo *log-error-S*.

   $ sort log-error -o log-error-S

```
$ cat log-error-S
error 04: connection failure
error 11: /tmp directory not found
error 11: /tmp directory not found
error 11: /tmp directory not found
error 11: /tmp directory not found
error 17: low disk space
error 22: out of memory
error 22: out of memory
```

Ahora usamos el comando *uniq* para escribir una línea única por cada tipo de error que ocurre y salvar la salida en el archivo *log-error-U*.

```
$ uniq log-error-S log-error-U
$ cat log-error-U
error 04: connection failure
error 11: /tmp directory not found
error 17: low disk space
error 22: out of memory
```

Como una alternativa podemos escribir las líneas únicas del *log-error-S* a la salida estándar solo con no especificar un archivo de salida.

```
$ uniq log-error-S
error 04: connection failure
error 11: /tmp directory not found
error 17: low disk space
error 22: out of memory
```

Si no se especifica un archivo de de entrada entonces *uniq* lee desde la entrada estándar. Podemos usar esta característica para pasar por tubería la salida del comando sort directamente al comando *uniq* sin guardar la salida ordenada a un archivo.

```
$ sort log-error | uniq
```

3. Use la opción -d para mostrar solamente esos errors que ocurren más de una vez.

```
$ uniq -d log-error-S
error 11: /tmp directory not found
error 22: out of memory
```

4. Use la opción -u para desplegar esos errores que solo ocurren una vez.

```
$ uniq -u log-error-S
error 04: connection failure
error 17: low disk space
```

5. Use la opción -c para contar el número de veces que cada error ocurre en el archivo *log-error*.

```
$ uniq -c log-error-S
 1 error 04: connection failure
```

```
 4 error 11: /tmp directory not found
 1 error 17: low disk space
 2 error 22: out of memory
```

Pase por tubería (pipe) los resultados del comando *uniq* al comando *sort* para listar los errores que ocurren con más frecuencia arriba de la salida.

```
$ uniq -c log-error-S | sort -n -r
 4 error 11: /tmp directory not found
 2 error 22: out of memory
 1 error 17: low disk space
 1 error 04: connection failure
```

Note que la opción -n de *sort* ordena numéricamente y no alfabéticamente y la opción -r pone los ítems en orden inversa (e.j. mayor-a-menor). El ejemplo anterior se puede igualar con el siguiente comando, usando el archivo original, sin ordenar *log-error* y una serie de tuberías.

```
$ sort log-error | uniq -c | sort -n –r
```

Los próximos ejemplos usaran el archivo *Compras*, el cual contiene el nombre de un cliente, la fecha y el artículo vendidos.

```
$ cat Compras
Juan julio Jan 2 Unidad 12
Juana Sanchez Jan 4 Unidad 17
Juan julio Jan 10 Unidad 12
Johnny Perez Jan 15 Unidad 17
Maritza Betances Jan 22 Unidad 05
Juana Sanchez Jan 30 Unidad 12
Liza Mejia Feb 2 Unidad 04
Juan julio Feb 4 Unidad 03
```

6. Genere un listado de cuantos artículos han sido vendidos. El primer paso será ordenar el archivo *Compras* empezando por el quinto campo, "Unidad 03", "Unidad 04", etc.

```
$ sort -k 5 Compras
Juan julio Feb 4 Unidad 03
Liza Mejía Feb 2 Unidad 04
Maritza Betances Jan 22 Unidad 05
Juana Sánchez Jan 30 Unidad 12
Juan julio Jan 10 Unidad 12
Juan julio Jan 2 Unidad 12
Johnny Perez Jan 15 Unidad 17
Juana Sánchez Jan 4 Unidad 17
```

Esta salida puede ser enviada por tubería al comando *uniq* con la opción -4 para que ignore los primeros cuatros campos y la opción -c para que de salida al conteo de cada línea.

$ sort -k 5 Compras | uniq -4 -c
1 Juan julio Feb 4 Unidad 03
1 Liza Mejía Feb 2 Unidad 04
1 Maritza Betances Jan 22 Unidad 05
3 Juana Sánchez Jan 30 Unidad 12
2 Johnny Perez Jan 15 Unidad 17

Los datos de nombre y fecha (los campos del uno al cuatro) en cada línea ya no tienen relevancia. El comando u*niq* ignora los primeros cuatro campos cuando determina líneas duplicadas. Si dos o más líneas son idénticas empezando con el campo número cinco entonces *uniq* usa los primeros cuatros campos de la primera línea que encuentra y elimina los primeros cuatro campos de las próximas líneas.

7. El comando cut puede ser usado para eliminar columnas no deseadas antes de usar el comando *uniq*. El siguiente ejemplo usa *cut* para eliminar los campos de nombre y fecha andes de usar los comandos *sort* y *uniq*.

$ cut -d' ' -f5,6 Compras | sort | uniq -c
1 Unidad 03
1 Unidad 04
1 Unidad 05
3 Unidad 12
2 Unidad 17

8. Use los comandos *cut*, *sort* y *uniq* para generar una lista de clientes y guárdela a un archivo y nómbrelo Clientes.

$ cut -d' ' -f1,2 Compras | sort | uniq > Clientes
$ cat Clientes
Juana Sánchez
Juan julio
Johnny Perez
Liza Mejía
Maritza Betances

El ejemplo anterior usa redireccionamiento de salida para guardar la salida del comando uniq a un archivo de nombre *Clientes* porque el comando un*iq* no permite un archivo de salida al menos que no se ha especificado uno de entrada.

9. Genere una lista de los clientes de compras repetidas.

$ cut -d' ' -f1,2 Compras | sort | uniq -d
Juana Sánchez
Juan julio

Seleccionar Partes de Líneas con cut

Descripción

cut [-b | -c | -f] list [options] [File ...]
cut [-b | -c | -f] lista [opciones] [Archivo ...]

El comando *cut* selecciona columnas desde un archivo y lo imprime a la salida estándar. Si no se especifica un archivo *cut* lee desde la entrada estándar. Las columnas pueden ser especificadas como bytes, caracteres o campos delimitados. Por ejemplo,

$ cut -c 1-10 Archivo1 Archivo2

Imprime los primeros 10 caracteres de cada línea del archivo Archivo*1* a la pantalla entonces imprime los primeros 10 caracteres de cada línea del archivo Archivo*2* a la pantalla.

- Seleccione desde un rango de:
  - Caracteres, con -c
  - Campos, con -f
- Separadores de campos pueden ser especificados con -d (por defecto es tab)
- Los rangos se especifican con posición de comienzo y fin: e.j., 3-5
  - Cualquier puede ser omitido
  - El primer carácter o campo es numerado como 1, y no 0
- Ejemplo: seleccione los nombres de usuarios ingresados en el sistema:

$ who | cut -d" " -f1 | sort –u

Las opciones de la línea de comandos de *cut* se describen a continuación.

Opción	Descripción
-b *list*	Las columnas son especificadas por posiciones de bytes.
-c *list*	Las columnas son especificadas por carácter. Por ejemplo, *-c 1-72* corta los primeros 72 caracteres de cada línea de un archivo.
-f *list*	Las columnas son especificadas por campos. Los campos deben ser separados por un carácter delimitador. El delimitador puede ser establecido con la opción *-d*. El delimitador por defecto es un **TAB**. Por ejemplo, *-f 2,5* selecciona el segundo y quinto campos de cada línea en un archivo con columnas separadas por **TAB**s. Si la línea no contiene ningún delimitador, *cut* imprimirá esa línea a la salida estándar, al menos que no se use la opción *-s*.
-d *c*	Especifica el campo delimitador cuando se usa la opción *-f*.
-s	Use la opción *-f*. Si una línea no contiene delimitadores, la opción *-s* detiene a cut de imprimir esa línea a la pantalla.

Debe especificar exactamente una de las opciones *-b*, *-c* o *-f* seguida por una lista, cual debe ser una lista de números enteros en orden ascendente separados por comas. Un guión puede ser usado como separador para indicar un rango completo. La siguiente tabla muestra algunos ejemplos.

Lista	Significado
n1,*n2*,*n3*	Corta *n1*, *n2* y *n3*.
n1-*n2*	Corta *n1* hasta *n2*.
n1-*n2*,*n3*	Corta *n1* hasta *n2* y *n3*.
-n1,*n2*	Corta desde 1 hasta *n1* y *n2*.
n1,*n2*-	Corta *n1* y desde *n2* hasta el fin de la línea.

Ejemplos

- En el archivo *dataset1*

  Pine 906 26 1.0 211
  Beech 933 26 2.3 160
  Fur 1246 27 2.44 162
  Palm 671 25 3.8 888

  Corte el segundo campo la cual esta almacenada en las columnas 13 al 17.

  $ cut -c 13-17 dataset1

  - Corte columnas del 1 al 72 desde el archivo *prog1.f* y redirecciones la salida desde la pantalla al archivo *code.f*.

  $ cut -c -72 prog1.f > code.f

- Corte todos los caracteres almacenado después de la columna 72 en el archivo *prog1.f* y guarde los resultados en un archivo llamado *comentario*.

  $ cut -c 73- prog1.f > comentario

- En el archivo dataset2 cual tiene ocho campos separado por un espacio.

  Pine 906 26 020079 130.0 80.3 17.1 211
  Beech 933 26 030079 48.0 85.2 22.7 160
  Fur 1246 27 070079 31.0 86.5 6.9 162
  Palm 671 25 100077 41.0 87.3 15.0 888

  Corte el segundo hasta el cuarto y entonces el séptimo campo.

  $ cut -f 2-4,7 -d " " dataset2

- En el archivo *dataset3* corte los campos 1, 3, 4, 5, 6 y el 8.

  Trees of the Forest
  Pine,906,26,020079,130.0,80.3,17.1,211
  Beech,933,26,030079,48.0,85.2,22.7,160
  Fur,1246,27,070079,31.0,86.5,6.9,162
  Palm,671,25,100077,41.0,87.3,15.0,888

$ cut -f 1,3-5,6,8 -d , dataset3

Esto despejara el archivo para hacerlo más legible, ya que no tenia un carácter delimitador. Para cortar los campos deseados sin incluir las líneas,

$ cut -f 1,3-4,6,8 -d , -s dataset3

- Lista los primeros 8 caracteres de cada archivo en directorio actual.
  $ ls -1 | cut -c 1-8

El comando ls -1 lista todos los archivos en el directorio actual en una única columna. La salida del comando *ls* se filtra por una tubería al comando *cut*, el cual selecciona los primeros ocho caracteres de los nombres de los archivos.

Ejemplos Avanzados

- Liste los nombres de usuarios (login names) de todos los usuarios ingresados en el sistema.
  $ who | cut -f 1 -d " "

El comando who lista todos los usuarios ingresados en el sistema. La primera columna contiene el nombre del usuario y las otras columnas contienen conformación adicional. La salida desde el comando *who* es pasada por la tubería al comando *cut*, el cual selecciona solo la primera columna de la salida.

- Despliegue las columnas una y cinco del archivo */etc/passwd*, que son el userid y su nombre real.
  # cut -f 1,5 -d : /etc/passwd

Note el signo de número o pound (#) significando que tenemos que encontrarnos en la cuenta de root para ejecutar este ejemplo.

Expandiendo la Tabulación a Espacios con *expand*

- Usado para reemplazar los tabulados con espacios en los archivos
- El tamaño del Tabulador (número máximo de espacios por tabulador) se puede establecer con -t número
  - Tamaño por defecto del tab es 8
- Para sólo cambiar el **Tab** al principio de las líneas, use -i
- Ejemplo: cambie todos los tabs en ***archivo.txt*** a tres espacios, y desplegarlo a pantalla:
  $ expand -t 3 archivo.txt
  $ expand -3 archivo.txt

Usar *fmt* para darle Formato a Archivos de Texto

- Coloca palabras ordenadamente en filas de longitud consistente
- Use -u para convertir a espacios uniformes
  - Un espacio entre palabras, dos entre oraciones
- Use -w width (ancho) para colocar la máxima anchura de los caracteres
  - Por defecto es 75
- Ejemplo: cambie el largo de las líneas de notas.txt a un máximo de 70 caracteres, y desplegarlo a pantalla:
  $ fmt -w 70 notas.txt | less

Leer las primeras Líneas de un archivo con head

- Imprime a pantalla las primeras líneas del archivo de entrada, obviando las otras.

- La opción -n indica el número de líneas a imprimir.
    - Por defecto imprime las primeras 10 líneas
- Para ver el encabezado de un archivo HTML llamado index.html:
  $ head index.html
- Para imprimir la primera línea de un archivo de texto (tienes dos alternativas):
  $ head -n 1 notas.txt (es un uno no una L)
  $ head -1 notas.txt (es un uno no una L)

Leer las últimas Líneas de un archivo con tail

- Muy similar a head, pero imprime las últimas líneas de un archivo
- La opción -f actualiza por siempre actualiza la salida a pantalla
    - Continuamente actualiza con salida al monitor que a medida que se suman líneas nuevas al archivo se despliega a pantalla
    - Para detener esta supervisión; Se le envía la señal de Kill con **Ctrl+C**
- La opción **-n** es la misma que la de head (el número de líneas a imprimir)
- Ejemplo: Para monitorear requisiciones HTTP de un web Server HTTP:
  $ tail -f /var/log/httpd/access.log

Enumerar Líneas de un archivo con nl o cat

- Despliega el archivo de entrada con sus líneas enumeradas
- Existen opciones par refinar el formato de la salida
- Por defecto, líneas en blanco no son enumeradas
    - La opción **-ba** numera todas las líneas
    - **cat -n** también enumera las líneas, incluyendo aquellas en blanco

Volcar Bytes de Data Binaria con od

- Imprime el valor numérico de los bytes en un archivo
- Útil para estudiar archivos con caracteres que no son del tipo texto
- Por defecto, imprime palabras (two-byte words) de dos bytes en octal
- Para especificar alternativa utilice la opción **-t**
- De una letra para indicar la base: **o** es octal, **x** para hexadecimal, u para decimal sin signo, etc.
    - Puede ser precedido por el número de bytes por palabra **(word)**
    - Agregue le una **z** para mostrar su equivalente en ASCII además de numérico
    - Anotación de opciones útil de **od -t x1z** - hexadecimal, de palabras de UN byte, con ASCII
- Alternativas a **od** incluyen **xxd** y **hexdump,** de echo distros modernas tienen alias creada cuando utilizas **od** en realidad estas trabajando con **hexdump**

Convertir archivos de Texto a archivos compaginados con pr

- Convierte un archivo de texto a un archivo dividido en páginas, con su cabezal y contenido de páginas
- Ya de muy poco utilizado por impresoras modernas, pero muy útil en el pasado
- Opciones:
- -d Salida de espacio doble
- -h header cambiar del cabezal por defecto a header
- -l líneas cambiar el número de líneas por defecto que es 66 a líneas
- -o ancho asignar el 'offset' del margen izquierdo al ancho de ancho

- Ejemplo:

$ pr -h "Mi Tesis" tesis.txt | lpr

El comando split

El comando *split* divide un archivo en varios archivos más pequeños. Usted puede especificar el tamaño de los archivos pequeños en bytes, kilobytes, megabytes o, si es un archivo de texto, por el número de líneas. Dividir nos permite distribuir un archivo en varios floppy disks, CDs, cintas (tapes) o cualquier otro tipo de media transportable. Luego, los archivos ya divididos pueden ser reestablecidos con el comando cat.

Descripción

split [options] [*infile*] [*outfile*]
split [opciones] [archivo-dividir] [archivo-dividido]

Por defecto, el archivo de salida es de 1000 líneas de largo. El comando s*plit* nombra los archivos de salida agregándole un sufijo único (por defecto aa, ab, ac, ...) al archivo de salida. Si no se especifica un archivo de salida, el comando *split* usa una **x** al principio del nombre del archivo de salida (xaa, xab, etc.). Si se usa un guión (-) en lugar de un archivo de entrada, *split* lee desde la entrada estándar.

Por ejemplo, supongamos que tenemos a Archivo-Largo con 4000 líneas.

$ split Archivo-Largo arch-peq

Esta sentencia creara cuatro archivos de nombre: arch-peq*aa*, arch-peq*ab*, arch-peq*ac*, arch-peq*ad*.

*Las o*pciones del comando split se muestra en esta siguiente tabla.

Opción	Descripción
-l *n*	Especifica el número de líneas en cada archivo de salida. Por ejemplo, "-l 80" divide el archivo de entrada en archivos de 80 líneas cada uno. El tamaño por defecto es de 1000 líneas. Note que el último archivo puede que tenga menos líneas de las n líneas. Las opciones -b y -l no se pueden usar juntas. Note: En algunos sistemas anteriores de Unix esta opción se especifica como -n. Por ejemplo, "split -100 Archivo.txt" divide a *Archivo.txt* en archivos de 100 líneas cada uno.
-b *n*[k | m]	Especifica el tamaño de los archivos de salida. Por ejemplo, "-b 1024" divide el archivo entrante en archivos de un tamaño de 1024 bytes. Se le agrega una k para especificar tamaño en kilobytes o una m para especificar el tamaño en megabytes. Por ejemplo, "-b 1m" divide el archivo de entrada en archivos de 1 megabytes. Las opciones -b y -l no deben ser usadas juntas
-a *n*	Usar *n* caracteres como sufijo en el archivo de salida. Por ejemplo, "-a 3" agregaría aaa, aab, ... al nombre del archivo de salida. Por defecto este valor es 2. Note: Esta operación no esta disponible en todos los sistemas *nix.

Ejemplos

Los siguientes ejemplos usan el archivo *archivo-largo.txt* cual es un archivo de texto de 4012 líneas.

$ wc -l archivo-largo.txt
4012 archivo-largo.txt
*****Vea el comando wc para más información sobre este comando usado para contar palabras.**

$ split archivo-largo.txt

El comando split divide a archivo-largo.txt en archivos más pequeños de 1000 líneas cada uno. Como no se le especifico un nombre al archivo de salida, el nombre base será x y los archivos pequeños serán nombrados xaa, xab, xac, xad y xae.

$ ls x??
xaa xab xac xad xae

Note que el quinto archivos, *xae*, solo tiene unas 12 líneas.

$ wc x??
1000 xaa
1000 xab
1000 xac
1000 xad
12 xae
4012 total

Los archivos de salida del comando *split* pueden ser reconstruidos usando el comando cat y la redireccion de la salida de este. Por ejemplo,

$ cat xaa xab xac xad xae > archivo-largo2.txt
$ diff archivo-largo.txt archivo-largo2.txt
diff: no differences

El comando diff compara dos archivos y lista las líneas en las cuales estos archivos difieren.

En la gran mayoría de sistemas *nix, GNU/Linux por supuesto uno de ellos, podemos usar comodines para evadir tener que digitar todos los nombres a los archivos de salida.

$ cat x?? > archivo-largo2.txt

1. Divide *archivo-largo.txt* en archivos de salida de 500 líneas cada uno. Nombre los archivos de salida arch-peq_*suffix* (i.e. *smfl_aa*, *smfl_ab*, ...)

   $ split -l 500 archivo-largo.txt arch-peq_
   $ ls arch-peq_??
   arch-peq_aa arch-peq_ac arch-peq_ae arch-peq_ag arch-peq_ai arch-peq_ab arch-peq_ad arch-peq_af arch-peq_ah

2. Divida *archivo-largo.txt* en archivos de salida con 100 líneas cada uno. Esto creara 41 archivos de la salida.

   $ split -l 100 archivo-largo.txt arch-100_
   $ ls arch-100_??
   arch-100_aa arch-100_aj arch-100_as arch-100_bb arch-100_bk
   arch-100_ab arch-100_ak arch-100_at arch-100_bc arch-100_bl
   arch-100_ac arch-100_al arch-100_au arch-100_bd arch-100_bm
   arch-100_ad arch-100_am arch-100_av arch-100_be arch-100_bn
   arch-100_ae arch-100_an arch-100_aw arch-100_bf arch-100_bo
   arch-100_af arch-100_ao arch-100_ax arch-100_bg

```
arch-100_ag  arch-100_ap  arch-100_ay  arch-100_bh
arch-100_ah  arch-100_aq  arch-100_az  arch-100_bi
arch-100_ai  arch-100_ar  arch-100_ba  arch-100_bj
```

Note que después de la *az* el próximo sufijo es *ba*. Podemos usar la opción "-a 3" para decirle al comando *split* que use 3 letras en el sufijo (e.j. aaa,..., aaz, aba,...)

```
$ split -a 3 -l 100 archivo-largo.txt arch-100_
$ ls arch-100_???
arch-100_aaa  arch-100_aal  arch-100_aaw  arch-100_abh
arch-100_aab  arch-100_aam  arch-100_aax  arch-100_abi
arch-100_aac  arch-100_aan  arch-100_aay  arch-100_abj
arch-100_aad  arch-100_aao  arch-100_aaz  arch-100_abk
arch-100_aae  arch-100_aap  arch-100_aba  arch-100_abl
arch-100_aaf  arch-100_aaq  arch-100_abb  arch-100_abm
arch-100_aag  arch-100_aar  arch-100_abc  arch-100_abn
arch-100_aah  arch-100_aas  arch-100_abd  arch-100_abo
arch-100_aai  arch-100_aat  arch-100_abe
arch-100_aaj  arch-100_aau  arch-100_abf
arch-100_aak  arch-100_aav  arch-100_abg
```

El próximo ejemplo use el archivo *arch-binario*, un archivo binario de 5048 kilobytes (alrededor de 4.9 megabytes, puede se un mp3 por ejemplo).

```
$ ls -s arch-binario
  5048 arch-binario
```

3. Divida el archivo *arch-binario* en archivos pequeños de tamaño de 1 megabyte cada uno, para poder ser copiados a disquete.

```
$ split -b 1m arch-binario arch-bin-peq_
$ ls arch-bin-peq_??
arch-bin-peq_aa arch-bin-peq_ab arch-bin-peq_ac arch-bin-peq_ad arch-bin-peq_ae
```

Archivos binarios pueden ser reconstruidos con el comando cat.

```
$ cat arch-bin-peq_?? > arch-binario2
$ diff arch-binario arch-binario2
diff: no diferencia in binary files
```

Los archivos tar, comprimidos o ejecutables pueden ser divididos con la opción -b y reconstruidos sin sufrir ningún daño.

4. Supongamos que tenemos un archivo de texto largo de nombre *documento1*. El comando pr puede ser usado para darle formato para imprimirlo.

```
$ pr -h "Linux Básico 1" documento1 > documento1-con-formato
```

El archivo *documento1-con-formato* ya tiene formato para poder imprimirlo con 66 líneas por página. Cada línea impresa tendrá su cabecilla incluyendo el titulo de "Linux Básico 1", la fecha y su número de página.

```
$ split -l 66 doc-con-formato-pr doc-con-formato-pg_
```

Creara archivos doc-con-formato-pg_*aa*, doc-con-formato-pg_*ab*, etc. Cada archivo contiene una página preparada para imprimir. Arriba de cada pagina se incluye una cabecilla con el titulo de "Linux Básico 1", fecha y el número de la pagina.

También podemos hacer esto usando una tubería y un guión (-) en lugar del archivo de entrada para decirle al comando *split* que lea desde la entrada estándar.

$ pr -h "Linux Básico 1" documento1 | split -l 66 - doc-con-formato-pg_

El comando diff

El comando *diff* reporta diferencias entre archivos. También puede ser usado para distribuir actualizaciones de archivos sin tener que distribuir archivos completos

Descripción

diff [options] Archivo1 Archivo2
diff [opciones] archivo1 archivo2

Diff reporta la diferencia entre dos archivos. Por ejemplo,

$ diff memo1 memo2
8c8
< 1) Usted no podrá estacionar en el patio A.

> 1) Usted no podrá estacionar en el patio B.

La salida de *diff* se envía a la salida estándar y consiste de lo siguiente.

- Si los archivos son idénticos, *diff* imprime nada.
- Si los archivos no son idénticos, cada diferencia encontrada entre los archivos es reportada con el comando **ed** requerido para convertir el archivo*1* al archivo*2*. ***Ed*** es un editor de línea de comando de texto. El comando ***ed*** empieza con los números de línea relevante del archivo*1* seguido por una letra única (c, d o a) entonces los números de línea del archivo*2*. Las letras pueden ser usadas para convertir archivo*1* a archivo*2* y tienen el siguiente significado.

c	Reemplaza líneas del *Archivo1* con las del *Archivo2*.
d	Elimina líneas del *Archivo1*.
a	Agrega líneas del *Archivo2* al *Archivo1*.

En el ejemplo anterior solo existe una diferencia entre los dos archivos *memo1* y *memo2* y esta diferencia es indicada por el comando *ed* "8c8".

- Cada reporte de diferencia incluye las líneas que difieren. Líneas desde el archivo*1* son precedidas por un símbolo de menos que (<). Líneas desde el *Archivo2* son precedidas por un símbolo de más grande que (>). Una línea discontinua de guiones (---) es usada para separar salida de los dos archivos.

Los argumentos de nombre de archivos, *Archivo1* y *Archivo2*, puede ser cualquier de lo siguiente.

- Nombre de archivos regular.
- Uno de los nombres de los archivos puede ser un guión (-) indicando que *diff* debe leer desde la entrada estándar.
- Si uno de los argumentos es un nombre de un archivo y el otro el de un directorio entonces *diff* compara nombre de archivo contra directorio/nombre de archivo (e.g. *diff Archivo1 Directorio1* es equivalente a *diff*

Archivo1 Directorio1/Archivo1).

- Si ambos nombres son directorios entonces *diff* compara todos los archivos que contienen estos directorios que los nombres de los archivos coinciden (e.g. *diff* compara a *Directorio1/Archivo1* a *Directorio2/Archivo1* y *Directorio1/Archivo2* a *Directorio2/Archivo2*, etc.) También genera un reporte de todos los nombres de archivos y subdirectorios que son únicos a un directorio y una lista de los nombres de los subdirectorios que son iguales en ambos directorios.
-

Algunas opciones útiles del comando *diff* se listan en esta tabla.

Opción	Descripción
-b	Ignora múltiple espacios en blanco (e.j. <espacio><espacio> es igual que <espacio>) y los espacios en blanco al final de las líneas.
-w	Ignora todos los espacios y los tabs (e.j. *1 o 2* es equivalente a *1o 2*).
-i	Ignora la capitalización (e.j. hola, HOLA y HoLa son equivalente).
-c	Usa la salida del formato del contexto. Salida del contexto incluye tres líneas antes y después de esas que normalmente se imprimen para dar "contexto" para la diferencias. Vea el ejemplo 10 para más información en formato de salida de contexto.
-C *n*	Como la opción -c pero incluye *n* líneas de salida de contexto.
-e	Produce un archivo script que puede ser usado por el comando ***ed*** para convertir el *Archivo1* al *Archivo2*. Véase ejemplo 9 para más detalles. Esta opción no es usada tan a menudo como es el comando patch para convertir *Archivo1* al *Archivo2*. Véase ejemplo 8 para aprender como usar *patch* con *diff*.
-h	Efectuar comparaciones más rápidas pero menos precisas. No es muy exacta si los archivos son muy diferentes y no pueden ser usados con la opción -e.

Las siguientes opciones son útiles para comparar dos directorios.

Opción	Descripción
-l	La salida es formateada para que la comparación de cada archivo ocurra en una página nueva. Se listan otras comparaciones en la página final.
-r	Recursivamente compara todos los archivos en subdirectorios comunes.
-s	Incluye un listado de todos los archivos idénticos en la salida.

Ejemplos

Los siguientes ejemplos usan los archivos memo1, memo2, poema1, y poema2.

- Reporte la diferencia entre los archivos *memo1* y *memo2*.

  $ diff memo1 memo2

8c8
< 1) Usted no podrá estacionar en el patio A;

> 1) Usted no podrá estacionar en el patio B;
19a20,21
> 4) Abril 3 es día de festivo.
>
22d23
< CEO y Presidente

Entre los dos archivos hay tres líneas que difieren.

La línea 8 difiere. Línea dice 8 "lot A" en el *memo1* y "lot B" en *memo2*.

Líneas 20 y 21 del *memo2* no existen en el *memo1*. Estas líneas deben ser agregadas después de la línea 19 del *memo1* para que los archivos sean iguales.

La línea 22 del *memo1* no existe en el *memo2*. Debe ser agregada después de la línea 23 del *memo2* para que los archivos sean iguales. Como es costumbre pensar convertir el primer archivo en el segundo archivo es mejor decir que la línea 22 debe ser eliminada del *memo1* para que los archivos sean los mismos.

$ diff Archivo1 Archivo2

Diff no reporta ninguna salida indicando que los archivos Archivo*1* y Archivo*2* son idénticos.

Más Ejemplos

- Reporte las diferencias entre los dos archivos *poema1* y *poema2*.

$ diff poema1 poema2
1a2
>
6c7
< Más bello de América, la más bella sinfonía de colores, el más grandioso derroche de luz...

> Más bello de América, la más bella sinfonía de colores, el más grandioso derroche de luz...
8c9
< Y tú estás conmigo, porque todos me abandonan... Tú conmigo en los postreros latidos de
---
> Y tú estás conmigo, porque todos me abandonan... Tú conmigo en los postreros latidos de
15c16
< Si yo hubiera muerto sobre un campo de batalla, dando frente al enemigo, te daría mi
---
> si yo hubiera muerto sobre un campo de batalla, dando frente al enemigo, te daría mi

Los dos archivos tienen cuatro diferencias.

- Línea 2 (línea en blanco) del *poema2* no existe en el *poema1*. Tendría que ser agregada después de la línea 1 del *poema1* para que los dos archivos sean idénticos.
- Para que el *poema1* sea convertido en el *poema2*, la línea 6 del *poema1* debe ser reemplazada con la línea 7 del *poema2*. El *poema1* tiene la palabra "América" mientras que el *poema2* tiene la palabra "América". Normalmente pensamos de la misma línea de dos archivos necesitan corresponder, pero como el *poema2* tiene una línea extra en blanco cerca del principio del archivo es en realidad la línea 7 del *poema2* que debe corresponder con la línea 6 del *poema1*.
- La líneas 8 del *poema1* y 9 del *poema2* son diferente. Hay dos espacios en vez de uno entre "postreros!" y "latidos" en el *poema2*.

- La línea 17 del *poema1* y la 18 del *poema2* son diferente. En el *poema1* la línea empieza con minúscula erróneamente.

- Reporte la diferencias entre el *poema1* y el *poema2*. Use la opción -i para ignorar las diferencias entre las mayúsculas y las minúsculas.

```
$ diff -i poema1 poema2
1a2
>
8c9
<   Y tú estás conmigo, porque todos me abandonan... Tú conmigo en los postreros latidos de
---
>   Y tú estás conmigo, porque todos me abandonan... Tú conmigo en los postreros  latidos
```

Las diferencias reportadas por "6c7" “15c16” en el ejemplo anterior ya no son reportadas porque son errores de capitalización y le pedimos a diff con la opción -i que ignore las mayúsculas y minúsculas.

- Reporte las diferencias entre poema1 y poema2 usando la opción -i y la opción -b la cual ignora todos los espacios en blanco repetidos y los que se encuentran al final de las líneas.

```
$ diff -i -b poema1 poema2
1a2
>
```

Las diferencias reportada desaparecen primero por lo que explicamos con la -i anterior y la del espacio doble es también ignorada porque usamos la opción -b.

- Reporte las diferencias entre el *poema1* y *poema2* usando la opción -i y la opción -w la cual ignora todos los espacios en blanco.

```
$ diff -i -w poema1 poema2
1a2
>
```

La única diferencia que aun existe entre los dos archivos es la extra línea en blanco en la línea 2 del *poema2*.

Ejemplo de Comparar Directorios

- Use el comando *diff* para comparar dos directorios.

```
$ diff Directorio1 Directorio2
diff Directorio1/Archivo2 Directorio2/Archivo2
2c2
< LINE 2
---
> LINE 2 is different
Only in Directorio1: Archivo4
Only in Directorio1: subdir
Common subdirectorios: Directorio1/subdir1 and Directorio2/subdir1
Only in Directorio2: subdir2
```

En este ejemplo, un archivo que existe en ambos directorios, Archivo*2*, es diferente. *Diff* reporta las diferencias encontradas en los dos archivos. El archivo*4* y el subdirectorio *subdir* solo existen el *Directorio1* mientras que el subdirectorio *subdir2* solamente existe en el *Directorio2*. Ambos directorios contienen un subdirectorio de nombre *subdir1*.

Uso de los Comandos Patch y Diff para Distribuir Cambios de Archivos

- Un uso del comando *diff* es para distribuir un conjunto de cambios que convierten el *Archivo1* al *Archivo2* y así no tener que distribuir el sistema de archivos completo. Esto se puede hacer usando la salida de ***diff*** con el comando **patch**. Primero guarde la salida de ***diff*** en un archivo usando redirección de la salida.

  $ diff memo1 memo2 > diffout

  Entonces use la salida del comando **diff** con el comando ***patch***.

  $ patch memo1 diffout

Ahora memo1 a sido convertido en memo2. El comando diff nos muestra que los dos archivos son idénticos.

$ diff memo1 memo2

Usar ed para Convertir Archivo1 a Archivo2

- La opción -e crea un script que da las directivas para que el editor de texto **ed** convierta el *Archivo1* al *Archivo2*. Por ejemplo,

  $ diff -e memo1 memo28c8
  < 1) Usted no podrá estacionar en el patio B;

  > 1) Usted no podrá estacionar en el patio A;
  19a20
  > 4) Abril 3 es día de festivo.
  23d23
  < CEO y Presidente

Para utilizar el script necesita guardarlo aun archivo usando la redirección de salida.

$ diff -e memo1 memo2 > edscript

Ahora el siguiente comando:

$ (cat edscript && echo w) | ed - memo1

Convierte a *memo1* a *memo2*. (Vea las definiciones de los comandos **cat, echo, subshell** y **pipes (|)** para más información.) Ahora:

$ diff memo1 memo2

Nos muestra que los archivos *memo1* y *memo2* son idénticos.

Ejemplo Diferencia de Contexto

- Reporte las diferencias entre los archivos *memo1* y *memo2* usando la opción -c para producir salida de contexto.

  $ diff -c memo1 memo2

```
*** memo1       2004-01-26 23:07:58.000000000 +0100
--- memo2       2004-01-26 23:07:22.000000000 +0100
***************
*** 5,11 ****
  Favor tome nota de los siguientes cambios en
  La política de la compañía y beneficios:

! 1) Usted no podrá estacionar en el patio B;
  Pero, si podrá el C.
  Transporte gratis se efectuara desde el patio a
```

```
  La puerta de la empres todos los días.
--- 5,11 ----
  Favor tome nota de los siguientes cambios en
  La política de la compañía y beneficios:

! 1) Usted no podrá estacionar en el patio A;
  Pero, si podrá el C.
  Transporte gratis se efectuara desde el patio a
  La puerta de la empres todos los días.
***************
*** 17,23 ****
  La cafetería estará cerrada para remodelar
  Desde Abril 1 a Abril 15.

  Gracias a Todos
  Roberto William
- CEO y Presidente
--- 17,23 ----
  La cafetería estará cerrada para remodelar
  Desde Abril 1 a Abril 15.

+ 4) Abril 3 es día de festivo.

  Gracias a Todos
  Roberto William
```

Como puede ver, la salida de contexto es muy diferente a la salida normal del comando *diff*. Salida de Contexto consiste de lo siguiente.

- Si los archivos son idénticos, *diff* imprime un mensaje indicando que no se encontraron diferencias.
- Si los archivos no son idénticos, *diff* inicia salida con un header indicando cuales archivos han sido comparados y la fecha que fueron modificados por última vez. La salida ferente al *Archivo1* es precedida por estrellas (***) y *Archivo2* por guiones (---).
- Luego cada diferencia es listada. Las diferencias son separadas por una línea larga de estrellas (**********).
- Cada diferencia contiene el contexto del *Archivo1* y del *Archivo2*. Contexto del *Archivo1* empieza listando el rango de líneas que se envían a la salida rodeada de estrellas. Las líneas de contexto mismas incluyen 3 líneas antes y 3 líneas después la línea o líneas que difieren. Recuerde que usted puede usar la opción -C n opción para usar n líneas y no 3. Luego el contexto del Archivo2 es impreso. Por Archivo2 el rango de líneas es rodeado por guiones y no por estrellas. Símbolos especiales usados para resaltar esas líneas de contexto que difieren. Los siguientes símbolos son usados.

!	Indica las líneas correspondientes en los dos archivos que son diferentes.
+	Indica líneas que existen en el *Archivo2* pero el *Archivo1*.
-	Indica las líneas que existen en el *Archivo1* pero no en *Archivo2*.

Avanzado: Ejemplo de Estatus de Exit

El estatus de salida del comando diff puede ser usado con la redirección de salida hacia el archivos /dev/null para determinar si dos archivos son iguales sin preocupación de diferencias en especifico. El comando diff excite con un status de 0 si no se encuentran diferencias, 1 si las diferencias fueran encontrada y asigna un número más grande que 1 si ocurre en un error. El siguiente es un ejemplo que asume que usted esta utilizando el Shell Bash o un de sus derivados. (Use usa $status en lugar de $? si usted usa el c-shell.)

```
$ diff memo1 memo2 > /dev/null
$ echo $?
1
```

El estatus de salida indica que *memo1* y *memo2* son diferentes.

```
$ diff Archivo1 Archivo2 > /dev/null
$ echo $?
0
```

El estatus de salida indica que los archivos Archivo*1* y Archivo*2* son idénticos. El estatus de salida puede ser extremadamente útil cuando se escriben scripts del shell. Por ejemplo, el script *isdiff* mostrado más adelante imprime un simple mensaje indicando un simple mensaje indicando si dos o más archivos son iguales o diferentes. El script usa la opción -h para ejecutar comparaciones más rápido pero con menos preedición. Esto es útil porque no estamos ingresados en el detalle exacto de como los archivos difieren.

```
#!/bin/sh
diff -h $1 $2 > /dev/null
if [ $? -eq 0 ]; then
  echo Los Archivos son Idéntico
elif [ $? -eq 1 ]; then
  echo Los Archivos son Diferentes
else
  echo Ha Ocurrido un ERROR
fi
```

```
$ isdiff memo1 memo2
files are different
$ isdiff Archivo1 Archivo2
files are idéntico
```

Invirtiendo archivos con tac

- Similar a cat, pero en reverso
- Imprime el archivo invirtiendo el orden de las líneas
- Ejemplo: para mostrar una lista de los logins y logouts, con los más recientes de último:

```
$ last | tac
```

Traducir Conjunto de Caracteres con tr

El comando *tr*, traduce caracteres, pude ser usado para substituir, comprimir o eliminar caracteres en un archivo.

Descripción

tr [options] *string1* [*string2*]
tr [opciones] cadena-texto *1* [cadena-texto *2*]

El comando *tr* copias texto desde la entrada estándar, reemplaza caracteres que igualan la cadena de caracteres *string1* con la cadena de caracteres del *string2* o reemplaza múltiple ocurrencias de caracteres en *string1* con un carácter único o elimina un carácter en el *string1* entonces imprime el resultado a la salida estándar. Por ejemplo,

$ tr "abc" "xyz" < archivo-entrada > archivo-salida

Reemplaza los caracteres *a* con *x*, *b* con *y* y *c* con *z* en archivo-*entrada* y almacena el resultado en archivo-salida. No requiere que los caracteres "*abc*" ocurran juntos para que la substitución tome lugar. La cadena "básico" se traduce a "yxsizo". Como en este ejemplo, el comando *tr* se usa a menudo con el uso de redirección de entrada y salida.

Ejemplos tr

- Reemplaza todos los caracteres en mayúsculas del archivo de entrada con minúsculas (dos alternativas):
  $ cat archivo-entrada | tr A-Z a-z
  $ tr A-Z a-z < archivo-entrada

- Borrar todas las ocurrencias de un carácter (z) en carta.txt:
  $ cat carta.txt | tr -d z

- Cambiar todas las ocurrencias de (ll) con una sola (l) en carta.txt
  $ tr -s l < carta.txt

Las opciones de *tr* se muestran en esta tabla.

Opción	Descripción
-s	Comprimir caracteres repetidos en la cadena de caracteres *string1*. Normalmente la opción *-s* no se usa con la opción *-d* o con una segunda especificación de cadena (*string2*). Por ejemplo, tr -s " " < archivo-entrante Reemplazara todas las ocurrencias de múltiple espacios en blanco con un solo en archivo-*entrante*.
-d	Elimina caracteres en la cadena *string1*. Normalmente la opción *-d* no es usada con la opción *-s* o con especificaciones de una segunda cadena (*string2*). Por ejemplo, tr -d "!" < archivo-entrante Eliminara todos los símbolos de admiración *en archivo-entrante*.
-c	Usa el complementario de la cadena de caracteres *string1*. (cada carácter excepto esos en el *string1*).

Especificar las Cadenas (Strings)

- Cuando especificada, la cadena *string2* debe ser de la misma longitud que la cadena *string1*. Cada carácter

en la *string1* será substituido por un carácter correspondiente en el *string2*.

- Las especificaciones de string deben estar entre comillas para que el shell no interprete los caracteres especiales.
- En algunos sistemas *nix, todo las cadenas (strings) deben estar encerradas entre corchetes cuadrados [].

El comando *tr* soporta varias características avanzadas al especificar los caracteres para las cadenas *string1* y *string2*. La siguiente lista las posibles especificaciones de cadenas.

c

Cualquier carácter del teclado (keyboard): alfabético, numérico o símbolo.

c-c

Especifica un rango de caracteres. Por ejemplo, *a-d* incluye los caracteres *a*,*b*,*c* y *d*.

c

Secuencia de escape. Secuencias de escape válido incluyen:

\\	backslash/Barra invertida
\n	newline/nueva línea
\r	carriage return/retorno de carro
\t	tab
\v	tabs verticales
\f	form feed/alimentado de formulario

[:*class*:]

Especificar una clase de caracteres. Clases validas son:

alnum	Caracteres alfabético o numérico
alpha	Caracteres alfabético [A-Za-z]
lower	Caracteres minúsculas [a-z]
upper	Caracteres mayúsculas [A-Z]
digit	Caracteres numéricos [0-9]
blank	tab o un espacio
space	Caracteres en blanco que incluyen el espacio, alimentado de forma, nueva linea, retorno de carro, tabs y tabs verticales.
punct	Caracteres de puntuación [~!@#$%^&*()_+|{}":<>?`-=\[];'/.,]
cntrl	Caracteres de control - tab, nueva línea, alimentado de forma, retorno de carro, etc.
print	Caracteres imprimibles – incluyendo el carácter de espacio pero no los caracteres d control

Las clases de caracteres *upper* y *lower* pueden ser usados para convertir de caracteres minúsculas a mayúsculas y vice-versa. Por ejemplo

```
tr "[:upper:]" "[:lower:]"
```

[*c*n*]**

Representa *n* repeticiones del carácter *c*. Solamente es valido al especificar la *string2*. Por ejemplo,

"[a*3]" es equivalente a "aaa". Si la *n* se omite *c* se repetirá las veces que sean suficiente para que el *string2* sea del mismo largo que el *string1*.

[=*equiv*=]

Todos los caracteres en la clase equivalente como la equiv. Clases Equivalente son establecidas en conjuntos que son agrupados naturalmente. Por ejemplo, todas las letras acentuadas como estas ò ó ô ö que son basadas en la misma letra base en este caso la o. Clases equivalentes solo pueden ser usadas cuando se especifica la cadena *string1*. Ellas no están disponibles para definir los caracteres de reemplazo en una substitución.

Ejemplos

1. El comando *tr* lee su entrada desde la entrada estándar y envía los resultados a la salida estándar. Es comúnmente usado con las redirecciones de entrada/salida. Por ejemplo,

   $ tr "[]" "()" < archivo-entrante > archivo-saliente
   Reemplázame todos los paréntesis cuadrados con paréntesis normales en el archivo-entrante y guárdame los resultados en el archivo-saliente.

2. Para editar un archivo usando el comando tr se requieren dos pasos. Primero traducir los caracteres en el archivo y luego guardar la salida a un archivo temporal.

   $ tr "[]" "()" < archivo-entrante > arch-temporario
   Y el segundo paso es, reemplazar el archivo original con el archivo temporario.

   $ mv arch-temporario archivo-entrante

El los ejemplos a continuación mostraremos ejemplos de tr sin especificar los archivos de entrante o el saliente.

Ejemplos: Comprimir Caracteres

- **$ tr -s "ab"**
  Comprime todas las ocurrencias múltiples de caracteres *a* y *b* en una sola. La cadena "*abaabbaaabbb*" será reemplazada con "*ababab*".
- **$ tr -s "\n"**
  Reemplaza todas las ocurrencias múltiples del carácter nueva línea (\n). Esto convertirá un archivo con doble, triple (o más) espaciado de línea a un archivo de lineado sencillo.
- **$ tr -s " \t"**
  Comprima todas las ocurrencias múltiples de un espacio o el carácter tab (\t).
- **$ tr -s "[:blank:]"**
  Comprima todas las ocurrencias múltiples de caracteres de la clase en *blanco*. Como la clase en *blanco* incluye solamente el espacio y tab (\t), este es igual que el ejemplo anterior.

Ejemplos: Eliminar Caracteres

- **$ tr -d "x"**
  Elimine todas las ocurrencias del carácter *x*.
- **$ tr -d "\t\f"**
  Elimine todos los caracteres de tabs (\t) y alimentado de hoja (\f).
- **$ tr -dc "[:print:]"**
  Elimina todos los caracteres que no están en la clase de caracteres que se imprimen. La opción -c específica el complementario y la clase de todos los caracteres que pueden ser impresos.

- **$ tr -dc "[:alnum:][:space:]"**
  Elimina todos los caracteres que no son alfabéticos, numéricos o caracteres espaciadores.

Ejemplos: Sustituir Caracteres

- **$ tr "abcde" "twxyz"**
  Reemplaza el carácter *a* con la t, *b* con la *w, c* con la *x, d con la y y la e por la z*. No requiere que "*abcde*" ocurran juntas para que la substitución se lleve a cabo. La cadena "básicamente fácil" se traduce a "*wtsixtmzntz ftxil*".
- **$ tr "\t" " "**
  Reemplazar todos los tabs (\t) por espacio.
- **$ tr "[A-Z]" "[a-z]"**
  Traducir todas las mayúsculas a minúsculas. Esto también puede ser llevado a cabo usando la especificación de clase de caracteres *upper* y *lower*.
- **$ tr "[:upper:]" "[:lower:]"**
- **$ tr "0123456789" "dddddddddd"**
  Reemplace todas las ocurrencias de un digito del *0* al *9* con la letra *d*. Podemos ilustrar varias maneras de lograr este objetivo.
- **$ tr "[0-9]" "[d*10]"**
  El rango [0-9] es usado en vez de escribir los diez dígitos. la especificación [d*10] significa repite el carácter d diez veces.
- **$ tr "[:digit:]" "[d*]"**
  La clase de carácter *digit* es usada para especificar los diez dígitos. La especificación [d*] significa repetir el carácter *d* cuanta veces sea necesario para que la cadena *string2* iguale la cadena *string1* en longitud.
- **$ tr -c "[:space:]" "[x*]"**
  Reemplaza cualquier carácter que no este en la clase de carácter de *space* con la letra *x*.
- **tr "[=o=]" o**
  Substituye la letra o por todos los caracteres que no sean de clase equivalente. Esto puede ser usado para remover cualquier marcado diacrítico

1. Substitución y compresión pueden ser llevas a cabo con un comando.

- **$ tr -s "ab" "xy"**
  Reemplaza *a* con una *x* y *b* con una *y*. Entonces comprime todas las ocurrencias múltiples de *x* y *y*. Esto tradujera la cadena "*aaabb*" a la cadena "*xy*". El comando anterior es equivalente a los dos comandos siguientes.
- **$ tr "ab" "xy"**
- **$ tr -s "xy"**
- **$ tr -cs "[:alnum:]" "[\n*]"**
  Reemplaza los caracteres que no son alfabéticos o numéricos con caracteres de nueva línea. Comprime todo los múltiples caracteres de nueva línea a un solo carácter de nueva línea. Esto imprime una palabra por línea.

Ejemplos Avanzados

- **$ echo $PATH | tr ":" "\n"**
  Imprime cada directorio en su ruta o path en una sola línea.

1. Supongamos que tenemos un grupo de archivos en el directorio actual que usted desea ejecutarle este mismo comando *tr*. Por ejemplo, deseamos comprimir todas las ocurrencias múltiples de espacio en blanco dentro de cada archivo que su nombre termine en '*.txt'*. Esto se puede lograr usando un bucle del shell (shell loop). El formato del loop es dependiente del shell que este en uso.

   C-Shell
   Si usted esta usando el c-shell o el tc-shell el siguiente comando trabajara.

   ```
   foreach f ( *.txt )
     cp $f $f.bak
     tr -s "[:space:]" < $f.bak > $f
   end
   ```

   Nota: Puede ser que necesite remover la opción de noclobber para usar este comando.

   % unset noclobber

   Bourne, Korn, Bash y Z-Shell
   Si esta usando el Shell Bourne o un derivado (incluyendo el Korn, bash o z-shell) el siguiente comando trabajara.

   ```
   for f in *.txt; do
     cp $f $f.bak
     tr -s "[:space:]" < $f.bak > $f
   done
   ```

   Nota: Puede ser que necesite remover la opción de noclobber para usar este comando.

   $ set +o noclobber

Además de convertir cada archivo, este comando creara una copia de seguridad del original y la nombrara nombre-original.bak. Agregue la línea "*rm $f*" al bucle para eliminar los archivos de resguardo.

Modificar Archivos con sed

- **sed** usa un simple script para procesar cada línea de un archivo
- Especifique el archivo script con **-f nombre-script**
- También puedes ejecutar comandos individuales con la opción **-e comando**
- Por Ejemplo: Si tienes un script llamado **corregir.*sed*** el cual corrige sus errores comunes, úsalo así:
  $ sed -f corregir.sed < carta.txt > carta-corregida.txt

Sustituir con sed

- Use el comando s/patrón/reemplazo/ para sustituir patrones encontrados con el patrón a reemplazarlo
- Agregue la el modificador **/g** para reemplazar todas las ocurrencias en todas las líneas no sólo la primera
- Por Ejemplo: reemplace ‘abre’ con ‘haber’:
- $ sed -e ’s/abre/haber/g’ carta.txt > carta-corregida.txt
- El **sed** tiene opciones más complicadas que nos permite ejecutar comandos condicionales
- Puede ser usado como lenguaje de programación básico (aunque no es muy amistoso al usuario!)

El comando paste

Descripción

paste [-s] [-d *char*] [Archivos...]

El comando paste fusiona líneas correspondientes de un archivo en columnas verticales e imprime el resultado a pantalla. Por ejemplo,

```
$ cat estatura
5'4"
6'2"
$ cat peso
124lb
180lb
$ paste estatura peso
5'4" 124lb
6'2" 180lb
```

Si uno de los archivos tiene menos líneas que el otro, el comando *paste* concatenara las líneas con el archivo más largo con una línea en blanco. Por ejemplo:

```
$ cat estatura
5'4"
$ cat peso
124lb
180lb
$ paste estatura peso
5'4" 124lb
     180lb
```

Colocar archivos en columnas con paste

El comando **paste** toma líneas desde dos o más archivos y los coloca en columnas y los presenta en la salida estándar.

Use la opción **-d** carácter para colocar el carácter como delimitando entre los campos a la salida

- El delimitador por defecto sin opciones es el **tab**
- Use la opción **-d** con más de un carácter para colocar un carácter diferente entre cada campo

Ejemplo: asigne contraseñas a usuarios desde una lista de usuarios y otro de password, produzca un archivo con los campos separados por (**:**):

$ paste -d: usuarios contraseñas > .htpasswd

Las opciones de la línea de comandos disponible al comando *paste* son las siguientes.

Opción	Descripción
-d *char*	Por defecto, las líneas fusionadas son delimitadas o separadas por el carácter **TAB**. La opción *-d* le dice al comando *paste* que separe las columnas con el carácter especificado por el argumento **char**. *Char* puede ser un carácter regular o uno de las siguientes secuencias de escape. **\n** Newline/Nueva línea **\t** Tab/Tabulador **\0** (Backslash seguido por un cero) Cadena vacía. \\ Backslash Las secuencias de escape deben de estar entre comillas para que el shell no las interprete como caracteres especiales. Puede separar columnas con diferentes caracteres solo con especificar más de un carácter *char*. Por ejemplo, -d '-*' separaría la primera de la segunda columna con un guión (-) y la segunda de la tercera columna con un asterisco (*). Si más de una columna existe, el comando *paste* alternaría usando guiones y asteriscos como delimitador.
-s	Fusiona todas las líneas de cada archivo en una línea. Cada **nuevaLinea** en un archivo, excepto la ultima, es reemplazada con un **TAB** o un delimitador especificado por la opción **-d**. Si múltiple archivos de entrada son especificados entonces habrá uno por línea por archivo impresos en el mismo orden que se listen los archivos en la línea de comandos.
-	Si se especifica un signo de menos (-) como el archivo de entrada entonces se usara la entrada estándar.

Ejemplos

- **$ paste Archivo1 Archivo2 Archivo3 > Archivo.txt**
  Crea un archivo nuevo, *Archivo.txt*, con tres columnas desde los tres archivos *Archivo1*, *Archivo2* y *Archivo3*. El resultado del comando paste son redireccionados desde la pantalla hacia el archivo de texto que nombramos *Archivo.txt*.

- **$ ls | paste -**
  Listaría todos los archivos del directorio actual en una columna. En efecto lo que la sentencia hace es pasar la salida del comando ls a través de una tubería (pipe) al comando *paste* - . El guión (-) especifica que la entrada estándar será usada como el archivo de entrada. Esta sentencia es equivalente a usar el comando *ls con la opción -1*.

- **$ ls | paste - - -**
  Lista todos los archivos en el directorio actual en tres columnas.

Los siguientes ejemplos usaran los archivos *estudiantes* y *notas*.

$ cat estudiantes	**$ cat notas**
Jenny	100
Antonio	92
Susana	88
Leo	97

Fusione las líneas correspondientes de los archivos *estudiantes* y *notas*.

```
$ paste estudiantes notas
Jenny  100
Antonio        92
Susana88
Leo            97
```

En este ejemplo un **TAB**, el delimitador por defecto, separa las columnas.

Fusione las líneas correspondientes de los archivos *estudiantes* y *notas* y guarde los resultados a un archivo de nombre *notas-estudiantes*.

```
$ paste estudiantes notas > notas-estudiantes
```

Fusione las líneas correspondientes de los archivos *estudiantes* y *notas* y separe las columnas con un solo espacio en blanco.

```
$ paste -d ' ' estudiantes notas
Jenny 100
Antonio 92
Susana 88
Leo 97
```

Fusione todas las líneas del archivo *estudiantes* en una sola línea.

```
$ paste -s estudiantes
Jenny  Antonio        SusanaLeo
```

Las líneas se convierten en columnas y son separadas por a **TAB**, the default delimitador.

Fusione todas las líneas del archivo *estudiantes* en una línea usando asterisco (*) como delimitador.

```
$ paste -s -d '*' estudiantes
Jenny*Antonio*Susana*Leo
```

Fusione todas las líneas del archivo *estudiantes* en una sola línea alternando entre el uso de un asterisco (*) y un símbolo de exclamación (!) como delimitador.

```
$ paste -s -d '*!' estudiantes
Jenny*Antonio!Susana*Leo
```

Fusione todas las líneas del archivo *estudiantes* en una línea usando el delimitador de **newline** (retorno de carro).

```
$ paste -s -d '\n' estudiantes
Jenny
Antonio
Susana
Leo
```

Esto no tienen ningún efecto otro que imprimir a pantalla es archivo *estudiantes* porque la opción *-s* le dice al comando *paste* que reemplace cada carácter de **newline** con el carácter especificado por la opción *-d*, que en nuestro caso es el carácter de **newline** (\n).

Fusione cada dos líneas en el archivo *estudiantes* en una sola línea.

```
$ paste -s -d '\t\n' estudiantes
Jenny Antonio
Susana Leo
```

La opción *-s* fusiona todas las líneas del archivo en una sola mientras que la opción *-d '\t\n'* alterna entre usar un **TAB** y un **newline** como el carácter delimitador.

Fusione todas las líneas de los archivos *estudiantes* y *notas* en una sola línea.

```
$ paste -s estudiantes notas
Jenny Antonio Susana Leo
100 92 88 97
```

Cree un archivo, *notas-estudiantes*, que contenga los primeros dos caracteres del nombre del alumno en el archivo es*tudiantes* en la primera columna y los números del archivo *notas* en la columna dos.

```
$ cut -c 1-2 estudiantes | paste - notas > notas-estudiantes
$ cat notas-estudiantes
Je 100
Bo 92
Su 88
Le 97
```

El comando join

El comando *join* hace una fusión de líneas correspondiente de dos archivos ordenados basada en una columna de data común.

Descripción

join [opciones] Archivo1 Archivo2
join [options] Archivo1 Archivo2

El comando join fusiona dos líneas correspondientes de dos archivos, Archivo1 y Archivo2, que contienen columnas de data (común llamarlos campos) que han sido ordenadas usando la mismas reglas de sortear (véase el comando **sort**). Si se usa un guión (-) en lugar de *Archivo1* o *Archivo2*, *join* lee desde la entrada estándar. Los resultados son escritos a la salida estándar. El comando j*oin* fusiona dos archivos a través de comparaciones de los datos en campos comunes. Por defecto, el campo común es el primer campo de cada archivo. Para todas las entradas que igualan, *join* escribe una ocurrencia del campo común, entonces todos los otros campos del *Archivo1* seguido por todos los campos del *Archivo2*. Por ejemplo,

$ cat prueba1	$ cat prueba2	$ join prueba1 prueba2
desiree 92 A	desiree 89 B+	desiree 92 A 89 B+
antonio 87 B+	antonio 94 A	antonio 87 B+ 94 A
marie 90 A-	marie 84 B	marie 90 A- 84 B

Las opciones del comando join se muestran en la siguiente tabla. En estas opciones, f puede ser 1 o 2 indicando Archivo1 o Archivo2.

Opción	Descripción
-t*c*	Especifica el carácter, *c*, que separa los campos. Usado para entrada y salida. Por ejemplo, "-*t,*" indica que comas separan los campos. Cada ocurrencia de c es significante así que *cc* representa un campo vacío. Por ejemplo, si el carácter separador es una coma entonces el campo "a,,d" es "a", el campo dos esta vacío y el campo tres es "d". Cuando no se usa-*t*, cualquier espacio en blanco es considerado un separador. En este caso, múltiple ocurrencias de espacios en blanco no son significativas. Porque en ambos casos de "a<espacio>b" y "a<espacio><espacio>b", campo uno es "a" y campo dos es "b".
-j*f n*	Especifica los campos comunes que son utilizados para fusionar. La fusión ocurre en el campo *n* del archivo *f*. Por ejemplo, "-j1 2 -j2 4" fusiona comparando el segundo campo del *Archivo1* al cuarto campo del *Archivo2*. Si se omite la *f*, se fusionan ambos archivos en el campo *n*. Por ejemplo, "-j 2" fusiona comparando el segundo campo del *Archivo1* al segundo campo del *Archivo2*. Por defecto, *join* fusiona en el primer campo de ambos archivos. Nota: Solo se puede especificar un solo campo por archivo. Por ejemplo, "-j1 2 -j 3" especifica campo dos del *Archivo1* y entonces campo tres del *Archivo1* y *Archivo2*. En este caso, solamente la ultima especificación es tomada en cuenta, "-j 3.
-o *f.n...*	Especifica orden de la salida. Da salida a el campo *n* del archivo *f*. Por ejemplo, "-o 1.2 2.1 1.3" muestra campo dos del A*rchivo1 seguido por campo uno del Archivo2* entonces seguido por campo tres del *Archivo1*. Nota: Cuando se usa la opción -*o*, el campo común no se le da salida automáticamente. Este debe ser especificado como cualquier otro campo. Si -*o* no es usado, *join* da salida a una ocurrencia del campo común, entonces todos los otros campos del *Archivo1* seguido por todos los otros campos del *Archivo2*
-a*f*	Salida a líneas sin aparear del archivo *f*. Por ejemplo, "-a1 -a2" dará salida a líneas sin aparear desde ambos archivos. Por defecto, líneas sin aparear no se le da salida. En algunos sistemas, si se omite la *f*, se le da salida a líneas sin aparear de ambos archivos.
-e *string*	Reemplaza campos vacíos con la cadena de texto ***string***. Debe ser utilizada con la opción -*o*.
-v *f*	En vez de la salida normal, imprime solamente las líneas sin par en el archivo *f*. Por ejemplo, "-v 1 -v 2" da salida a las líneas sin aparear en ambos archivos.

Ejemplos

Los siguientes ejemplos usan los archivos *prueba1* y *prueba2*, archivos que contienen el nombre y las notas del estudiante.

$ cat prueba1	**$ cat prueba2**
marie 79	karen 91
karen 83	antonio 84
antonio 92	marie 95
suzie 85	andy 87

- Fusione los archivos *prueba1* y *prueba2* apareando por nombre de estudiante. El primer paso es ordenar ambos archivos por el campo uno (nombre del estudiante).

  $ sort -k 1 prueba1 > prueba1s
  $ sort -k 1 prueba2 > prueba2s

Los archivos *prueba1s* y *prueba2s* contienen la misma data de los archivos *prueba1* y *prueba2* pero ya ordenada alfabéticamente por el nombre del estudiante. Si usted no esta familiarizado con el uso del carácter (>) para redireccionar la salida a un archivo, debe volver al capitulo que se refiere a control de entrada y salida. El comando sort será discutido más adelante en su propia sección.

$ join prueba1s prueba2s
antonio 92 84
karen 83 91
marie 79 95

Note que *join* no da salida a las líneas que no fueron apareadas. Estudiantes que no aparecen en uno de los dos archivos no aparecen en la data de salida del comando.

- Fusione *prueba1* y *prueba2* por nombre de estudiantes incluyendo las líneas no apareadas de ambos archivos.

  $ join -a1 -a2 prueba1s prueba2s
  andy 87
  antonio 92 84
  karen 83 91
  suzie 85
  marie 79 95

La opción "*-a1*" incluye las líneas no apareadas del Archivo1 (*prueba1s*) y la opción "*-a2*" incluye las líneas no apareadas del Archivo2 (*prueba2s*).

Muestre los estudiantes que estuvieron ausentes del primer y/o segundo examen.

$ join -v 1 prueba1s prueba2s
suzie 85

La opción "*-v 1*"muestra las líneas no apareadas del Archivo1 (*prueba1s*). Estos son los estudiantes que tomaron el primer examen pero no el segundo. De la misma manera, el siguiente comando muestra los estudiantes que tomaron el segundo examen pero no el primero.

$ join -v 2 prueba1s prueba2s
andy 87

Use las opciones "*-v 1*" y "*-v 2*" simultáneamente para dar salida a los estudiantes que no tomaron o el primer o segundo examen.

$ join -v 1 -v 2 prueba1s prueba2s
andy 87
suzie 85

Los siguientes ejemplos usan la identificación del empleado almacenada en un archivo de nombre *empleadoID.txt* y el archivo *Pago.txt*. El archivo *empleadoID.txt* contiene un número de identificación, su nombre y apellido. El archivo *Pago.txt* contiene un número de identificación del empleado, el salario y la bonificación del fin de año. Los campos son separados por dos puntos (:).

$ cat empleadoID.txt	**$ cat Pago.txt**
1001:Juana:Sanchez	1001:40,000:400
1002:Michael:Foster	1002:45,000:450
1003:Monica:Rodriguez	1003:35,000:350
1004:Angel:Gonzalez	1004:22,000:220
1005:Tita:Medina	1005:39,000:390

- Fusione *empleadoID.txt* y *Pago.txt* basado en el campo número de identificación del empleado.

  $ join -t: empleadoID.txt Pago.txt
  1001:Juana:Sanchez:40,000:400
  1002:Michael:Foster:45,000:450
  1003:Monica:Rodriguez:35,000:350
  1004:Angel:Gonzalez:22,000:220
  1005:Tita:Medina:39,000:390

La opción "*-t:*" le dice a al comando *join* que los campos están separados por (:). Note que la salida esta ordenada así el campo común (número de identificación del empleado) seguido por todos los otros campos del Archivo1 (*empleadoID.txt*) luego todos los otros campos del Archivo2 (*Pago.txt*).

- Fusione los archivos *empleadoID.txt* y *Pago.txt* basado en el campo número de identificación del empleado, y de salida solo al nombre y salario del empleado.

  $ join -t: -o 1.3 2.2 empleadoID.txt Pago.txt
  Sanchez:40,000
  Foster:45,000
  Rodriguez:35,000
  Gonzalez:22,000
  Medina:39,000

La opción "-o 1.3 2.2" da salida al tercer campo del Archivo1 (apellido del archivos *empleadoID.txt*) seguido por el segundo campo del Archivo2 (salario anual del archivo *Pago.txt*).

El siguiente ejemplo utiliza los archivos *Notas01* y *Notas02*, los cuales contienen la fecha, nombre del estudiante y sus notas.

$ cat Notas01	**$ cat Notas02**
Dec 30 2005 Jimenez Marie 79	Feb 4 2006 Jimenez Marie 91
Dec 30 2005 Lopez Karen 83	Feb 4 2006 Lopez Karen 72
Dec 30 2005 Foster Roberto 92	Feb 4 2006 Foster Roberto 84

- Fusione los archivos Notas01 y Notas02 para que la salida contenga el apellido, nombre, notas del examen de Dec 30 2003 y notas del examen de Feb 4 2004.

  $ join -j 4 -o 1.4 1.5 1.6 2.6 Notas01 Notas02
  Jimenez Marie 79 91
  Lopez Karen 83 72
  Foster Michael 92 84

La opción "-j 4" fusiona en los campos cuatro de ambos archivos. La opción "-o 1.4 1.5 1.6 2.6" da salida a los campos cuatro, cinco y seis del archivo *Notas01* seguido por el campo seis del archivo *Notas02*.

Ejemplos Avanzados

Ejemplo de Substitución

El siguiente ejemplo usa el archivo *mf*, el cual contiene una lista de nombres y una M o F para Masculino o Femenino.

```
$ cat mf
andy M
Juana F
jim M
michelle F
john M
sue F
sharon F
```

Reemplace la M con un número 1 y F con un número 2. Primero cree un archivo ***trans*** que contenga el siguiente texto

```
F 2
M 1
```

Luego, orden con sort el archivo mf por el contenido del segundo campo.

```
$ sort -k 2 mf > mfs
$ cat mfs
Juana F
michelle F
sharon F
sue F
andy M
jim M
john M
```

Ahora fusione campo dos del archivo ***mfs*** con el campo uno del archivo ***trans*** y de salida solamente al nombre y el número.

```
$ join -j1 2 -j2 1 -o 1.1 2.2 mfs trans
Juana 2
michelle 2
sharon 2
sue 2
andy 1
jim 1
john 1
```

Esto se puede lograr sin crear el archivo mfs.

$ sort -k 2 mf | join -j1 2 -j2 1 -o 1.1 2.2 – trans

La tubería usa la salida estándar del comando sort como la entrada estándar para el comando **join**. El guión – le dice a *join* que use la entrada estándar como Archivo1.

Puede ser que también desee reordenar la salido por nombre.

$ sort -k 2 mf | join -j1 2 -j2 1 -o 1.1 2.2 – trans | sort -k 1

Diferentes Separadores

El siguiente ejemplo usa los archivos *Archivo1* y *Archivo2*.

$ cat Archivo1	**$ cat Archivo2**
aa 1	aa,4,7
bb 2	bb,5,8
cc 3	cc,6,9

Fusione *Archivo1* y *Archivo2* usando el campo uno. *Join* requiere que ambos archivos de entrada usen el mismo separador de campo así que uno de los dos archivos deberá ser editado. El siguiente ejemplo usa el comando **sed** para reemplazar cada ocurrencia de una coma en *Archivo2* con un espacio en blanco.

$ sed 's/,/ /g' Archivo2 > Archivo2-out
$ join Archivo1 Archivo2-out
aa 1 4 7
bb 2 5 8
cc 3 6 9

Dando Formato a la Salida

El siguiente ejemplo usa los archivos *empleadoID.txt* y *Pago.txt*. El archivo *empleadoID.txt* contiene un número de identificación del empleado, nombre y apellido. El archivo *Pago.txt* contiene un número de identificación del empleado, salario y la bonificación.

$ cat empleadoID.txt	**$ cat Pago.txt**
1001 Juana Sanchez	1001 40,000 400
1002 Michael Foster	1002 145,000 1450
1003 Monica Rodriguez	1003 35,000 99

Fusione los archivos *empleadoID.txt* y *Pago.txt* por el campo número de identificación del empleado.

$ join empleadoID.txt Pago.txt
1001 Juana Sanchez 40,000 400
1002 Michael Foster 145,000 1450
1003 Monica Rodriguez 35,000 99

No importa como se usen los espacios en blanco en los archivos de entrada, *join* solo usara o interpretara un solo espacio en blanco para separa los campos de salida. El siguiente ejemplo usa **awk** para darle formato a la salida para que se vea mejor.

**$ join empleadoID.txt Pago.txt | awk '{printf("%-5s %-8s %-10s %8s %7s\n", **
$1, $2, $3, $4, $5)}'

```
1001  Juana   Sanchez       40,000    400
1002  Michael  Foster          145,000   1450
1003  Monica  Rodriguez       35,000    99
```

Dar Formato a Salida no Pareada

El siguiente ejemplo utiliza los archivos *Cabellos1* y *Ojos2*.

$ cat Cabellos1	**$ cat Ojos2**
desiree marron	roberto azul
roberto rojo	cindy verde
Juana negro	Juana marron
marie amarillo	marie azul

- Fusion de los archivos *Cabellos1* y *Ojos2* incluyendo las líneas no apareadas de ambos archivos.

```
$ join -a1 -a2 Cabellos1 Ojos2
roberto azul
cindy verde
desiree marron
roberto rojo
Juana negro marron
marie amarillo azul
```

Como la salida incluye líneas no apareadas, el color de cabellos y ojos no esta en campos distintos. El color de los ojos de Cindy, verde, es colocado en el campo dos porque ella no tiene definido el color del cabello.

Use la opción *-o* con la opción *-e* para colocar un *ND (no disponible)* en los campos que no contienen data para que así el color del cabello y los ojos estén en la columna correcta.

```
$ join -a1 -a2 -o 1.1 1.2 2.2 -e ND Cabellos1 Ojos2
desiree marron ND
roberto rojo azul
ND ND verde
Juana negro marron
marie amarillo azul
```

Esta salida no es ideal. El nombre *cindy* fue reemplazado con *ND* porque el campo nombre es leído desde *Cabellos1* y cindy no tiene una entrada en el archivo *Cabellos1*.

Para corregir la salida del ejemplo anterior, cuando existe una línea no apareada en el archivo *Cabellos1* necesitamos utilizar el nombre del campo uno del archivo *Cabellos1* y el nombre del campo uno del archivo *Ojos2* cuando existe una línea no apareada en el archivo *Ojos2*. Esto es dificultoso pero no imposible. Primero ejecute

```
$ join -a1 -o 1.1 1.2 2.2 -e ND Cabellos1 Ojos2 > temp.txt
$ cat temp.txt
desiree marron ND
Juana negro marron
marie amarillo azul
```

roberto rojo azul

El comando *join* imprime todas las líneas apareadas más las líneas no apareadas del archivo *Cabellos1* en el orden nombre (como es leído desde el archivo *Cabellos1*), color de cabellos, color de los ojos y reemplaza cualquier campo sin data con una entrada de *ND*. La salida es redireccionada (guardada en) al archivo de texto *temp*.txt. Ahora ejecute

```
$ join -v 2 -o 2.1 1.2 2.2 -e ND Cabellos1 Ojos2 >> temp.txt
$ cat temp.txt
desiree marron ND
Juana negro marron
marie amarillo azul
roberto rojo azul
cindy ND verde
```

El comando *join* imprime todas las líneas no apareadas del archivo *Ojos2* en este orden nombre (directamente del archivo *Ojos2*), color del cabello, color de los ojos y reemplaza cualquier data no incluida en los campos con *ND*.

Nota: Como solo le dimo salida a las líneas sin aparear del archivo *Ojos2*, color de cabellos no estará presente. La salida es agregada al archivo *temp.txt*, el cual ahora contiene nombre, color de los cabellos y los ojos en la columna correspondiente; pero, *temp.txt ya no estará ordenada alfabéticamente por nombre*. Para regresar al archivo *temp.txt a su orden alfabética por nombre, ejecute*:

```
$ sort -k 1,1 temp

cindy ND verde
desiree marron ND
Juana negro marron
marie amarillo azul
roberto rojo azul
```

- El ejemplo anterior puede ser ejecutado sin el uso del archivo temporario.

```
$ ( join -a1 -o 1.1 1.2 2.2 -e ND Cabellos1 Ojos2 ; \
join -v 2 -o 2.1 1.2 2.2 -e ND Cabellos1 Ojos2 ) \
| sort -k 1,1
roberto ND azul
roberto rojo ND
cindy ND verde
desiree marron ND
Juana negro marron
marie amarillo azul
```

Como es que esto todo funciona? El punto y coma (;) se usa para enlazar dos comandos juntos. Los paréntesis son usados para ejecutar ambos comandos en solo subshell para que la salida pueda ser redireccionada simultáneamente hacia el comando *sort*. Las barras invertidas o backslashes son solo utilizados para poder distribuir una sola sentencia de comando en más de una sola línea de comando.

- Y ahora este si es de verdad...disfruta este comando...

```
$ ( echo NAME CABELLOS OJOS ; \
( join -a1 -o 1.1 1.2 2.2 -e ND Cabellos1 Ojos2 ; \
join -v 2 -o 2.1 1.2 2.2 -e ND Cabellos1 Ojos2 ) \
| sort -k 1,1 ) | awk \
'{printf("%-10s %-10s %-10s\n", $1, $2, $3)}'
```

NAME	CABELLOS	OJOS
cindy	ND	verde
desiree	marron	ND
Juana	negro	marron
marie	amarillo	azul
roberto	rojo	azul

Logrando Joins tipo Base de Datos con join

- Hace un 'inner join' tipo base de datos de dos tablas, almacenadas en archivos de texto
- La opción **-t** establece el delimitador del campo
    - Por defecto, los campos se separan por un número de espacios o tabs

Ejemplo: muestre una lista de suplidores y sus productos de dos archivos:

$ join suplidores.txt productos.txt | less

- Los archivos deben ser sorteados en orden previamente!
- Este comando es utilizado muy poco, ya que las bases de datos contienen esta utilidad

Práctica4

Ejercicio 1

1) Use **cut** para desplegar una lista de usuarios ingresados en el sistema. (Verifique con **who**)

2) En el ejemplo de arriba imprima los usuarios sin duplicados y en orden alfabética.

3) Pruebe con el comando **last** para desplegar el record de quienes han ingresado al sistema, con el comando **tac** reverse el orden. Para que fuese esto útil? Si la salida es extensa como la direcciona al comando **less**?

4) Use **sed** para corregir el error ortográfico 'sostema' a 'sistema'. Escriba un archivo en **nano**, para probar su comando. ¿Que pasa si el error ocurre más de una ves, y que se puede hacer?

5) Use **nl** para enumerar las líneas que escribió en el ejemplo de arriba para corregir el error.

Ejercicio 2

1) Cree un archivo vacío y utilizando **tail -f** monitoree la actividad de el. Agréguele líneas de texto desde otro terminal, así: **$ echo "sólo es una prueba" >> archivo-vacío**

2) Una ves ha escrito al archivo, use el comando **tr** para desplegarlo con todas las veces que las letras A-F aparezcan se cambien a los números 0-5.

3) Intente leer el comando binario **ls (*/bin/ls*)** con **less**. Si es necesario use la opción **-f** para forzarla a desplegar aunque no es un archivo de texto.

4) Ahora despliéguelo con **od**. Primer en los valores por defecto y luego con las opciones para desplegar la salida en hexadecimal.

Ejercicio 3

1) Use el comando **split** para dividir el comando binario **ls** en pedazos de 1Kb. Haga esto en un directorio nuevo, para después poder borrarlo más tarde.

2) Ahora entre al directorio que despedazo el comando **ls** y vuelva a reponerlo, y entonces ejecútelo asegure que ejecute el suyo y no el del sistema; Ejemplo ./**mi-ls**, y asegúrese de que este como ejecutable antes de correrlo con el comando: $ **chmod a+rx mi-ls**

Ejercicio 4

1) Use el comando cd para ir a su home, y crear un directorio nuevo llamado perros.

2) Cree otro dentro del de arriba llamado gatos, y otro más dentro de este llamado ratones.

3) Remueva los tres directorios. Puedes removerlo uno a la vez o todos juntos.

Capítulo5

Pero la misma mente que esta afuera de la Matriz es la misma mente viva de cada una de las personas que interactúan con ella

Trinity, The Matrix

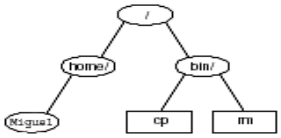
Manejo de Archivos de Texto

Los Objetivos de este Capítulo son:

1) Editores de Textos Básicos, vi
2) Conceptos de Entrada/Salida
3) Redirección y Tuberías
4) Filtros y comandos de manipulación de texto
5) Sortear y ordenar
6) Cortar y pegar desde la línea de comandos
7) Manipulación básica como copiar, mover, de archivos

Objetos de Sistema de Archivos

- Un **archivo e**s un envase para almacenar data: una posible secuencia vacía de bytes
- Un **directorio e**s una colección de archivos y otros directorios
- Los Directorios están organizados en forma jerárquica, con el **directorio root (/)** en la sima del árbol
    - El directorio root es referido como la barra /



Directorios y los Nombres de Archivos

La organización de archivos y directorios es conocida como sistema de archivos (**filesystem**)
Para referirse a archivos dentro de directorio y subdirectorios deberá separarlos con barra /, Ejemplo:

/bin/ls
/usr/share/dict/words
/home/miguel/carta.txt

Rutas a los archivos empiezan con / (si son absoluto) o desde el directorio actual.

Archivos y sus Extensiones

Es práctica común colocarle un punto y una **extensión**, al final de los archivos
La extensión nos indica que tipo de archivo es:

.txt	Archivo de Texto
.gif	Tipo imagen - Graphics Interchange Format
.jpg	Tipo imagen - Joint Photographic Experts Group
.mp3	Tipo audio - MPEG-2 Layer 3
.gz	Archivo Comprimido
.tar	Archivo Tipo Unix de Cintas 'tape archive'
.tar.gz, ***.tgz***	Archivo Comprimido

En GNU/Linux como en Unix, extensiones de archivos son sólo una convención

Para el **kernel** las extensiones son sólo parte del nombre y nada más.
- Algunos programas usan extensiones para determinar el tipo de archivo

Regresando al Directorio Anterior

- El comando **pushd** te lleva a otro directorio al igual que **cd**
  - Pero a la misma vez almacena el directorio actual, y así podrás regresar a este con poco esfuerzo
  - Por Ejemplo, para visitar el directorio home del usuario miguel, y después de una serie de comandos retornar a donde empezamos:

```
$ pushd ~miguel
$ cd /usr/share/pixelmaps
$ ls
...
$ popd
```

El comando **popd** te regresa de donde ejecutaste el comando **pushd**
El comando **dirs** listara los directorios a que el comando **popd** te regresará

Completar Nombre de Archivos

- Los shells modernos te ayudan escribir los comandos y nombres de archivos y directorios largos y a veces repetitivos desde la línea de comandos
- Escriba las primeras letras de un comando (net) y presione la tecla del tabulador (tal ves dos veces) Tab
- Si el nombre es ambiguo (o sea existen varios comandos que empiezan igual), el shell le dará estas opciones:
  - En **Bash**, pulse Tab dos veces consecutivas
  - En el shell C, pulse Ctrl+D

Estas dos shells escaparán automáticamente los espacios y caracteres especiales en los nombres de los archivos.

Patrones de Comodines (Wildcard)

Pasar múltiples archivos a un comando especificando un patrón
Use el símbolo * para igualar cualquier parte del nombre de un archivo:

```
$ ls *.txt
listado.txt   carta.txt    reportes.txt
```

El comodín * produce el nombre de todos los archivos en un directorio
El comodín **?** iguala un carácter exactamente:

```
$ rm -v carta.?
removing carta.1
removing carta.2
removing carta.3
```

Nota: El shell expande los comodines a nombres completos, así pues los programas que les pasamos los nombres con comodines solo ven el nombre completo.

Copiar Archivos con cp

Sintaxis: **cp [*opciones*] archivo-origen *archivo-destino***
Copiar múltiple archivos a un directorio:

$ **cp archivo1 archivo2.... archivoX *directorio/***

Opciones Común:
- -f, forza sobre escritura de los archivos de destino
- -i, interactivo, pregunta antes de sobre escribir un archivo

- -a, archivo, copia el contenido de directorios recursivamente

Ejemplos de cp

Para copia */etc/smb.conf* al directorio actual:

$ cp /etc/smb.conf .

Para crear una copia idéntica del directorio Trabajo y llamarla Trabajo-BAKUP:

$ cp -a Trabajo Trabajo-BACKUP

Para copiar todos los archivos de imágenes **GIF/JPEG** desde el directorio actual al directorio imagen:

$ cp *.gif *.jpeg imagen

Mover Archivos con mv

El comando **mv** puede renombre archivos y directorios, o moverlos a otros directorios
Es el equivalente a copiar y luego borrar
La ventaja es que es más rápido
Opciones:

- -f, forza sobre escribir, aunque el archivo destino ya exista
- -i, pregunta interactivamente antes de sobre escribir los archivos:

$ mv carta.txt reporte.txt

Mover todo en el directorio actual para otro lugar:

$ mv * ~/back-up/

Borrando los Archivos con rm

rm borra ('remueve') un archivo especifico
Tienes que tener permiso de escritura para el directorio que lo contiene, para removerlo
¡Use cuidadosamente si esta en sección como root!
Opciones:

- **-f,** borra archivos protegido de escritura sin pedir confirmación
- **-i,** interactivo - preguntar al usuario antes de borrar archivo
- **-r,** recursivamente borra archivos y directorios

- Por Ejemplo, limpiar todo */tmp*, sin pedir confirmación, borra cada archivo:

$ rm -rf /tmp/*

Borrar archivos con nombres Peculiares

Algunos archivos tienen nombres que los hacen difícil de borrar
Archivos que comiencen signos de menos:

$ rm ./-nombre-archivo
$ rm -- -nombre-archivo

Archivos que contienen caracteres peculiares, quizás caracteres que no puedas escribir con tu teclado:

- Escribe un patrón de comodín que combine solamente con el nombre que quieras borrar:

$ rm -i ./nombre-con-caracteres-peculiares*

- El **./** le obliga a estar en el directorio actual
- Usar la opción **-i** con **rm** garantiza que no borraras nada por accidente

Crear Directorios con mkdir

Sintaxis: mkdir nombre_directorio
Opciones:

- -p, crea los directorios padre si no existen
- -m permisos, ajusta los permisos de acceso al directorio creado

Por Ejemplo, cree un directorio llamado mis_archivos en su directorio home con permisos para que solo usted

pueda escribirle, pero que todos puedan leerlo:

$ mkdir -m 755 /home/miguel/mis_archivos

- Cree un árbol de directorio debajo de /tmp, con tres subdirectorios llamados uno, dos y tres con un comando:

$ mkdir -p /tmp/uno/dos/tres

Remover Directorios con rmdir

El comando **rmdir** borra solo directorio vacío, así es que los archivos deben ser borrados primero:
Por Ejemplo, para borrar el directorio imagen:

$ rm imagen/*
$ rmdir imagen

Para directorios que no están vacíos, use: **rm -r *directorio***
La opción **-p** de **rmdir** borrar la ruta completa, si no contiene otros archivos o directorios dentro
Estos comandos son equivalentes:

$ rmdir -p a/b/c
$ rmdir a/b/c a/b a

Identificar los Tipos de Archivos

La data en los archivos tiene variados formatos (programas ejecutables, archivos de texto, etc.)
El comando **file** trata de identificar los diferentes tipos de archivos:

$ file /bin/bash

/bin/bash: ELF 32-bit LSB executable, Intel 80386, version 1, dynamically linked (uses shared libs), stripped
También nos provee con información adicional sobre los archivos.
Es bien útil para analizar si un archivo es un script:

$ file /usr/bin/zless
/usr/bin/zless: Bourne shell script text

Si el comando file no reconoce el formato específico del archivo; tratara de adivinarlo:

$ file /etc/passwd
/etc/passwd: ASCII text

Cambiar Fecha de Acceso con touch

Cambia el tiempo de **acceso y modificación** de los archivos
Si el archivo no existe lo crea
Opciones:

- -a, cambia solo el tiempo de acceso
- -m, cambia el tiempo de modificación del archivo
- -t [YYYY]MMDDhhmm[.ss], ajusta el atributo de tiempo de los archivos a esta fecha específica
- GNU touch tiene la opción **-d** cual acepta la fecha en formatos más flexibles
- Por Ejemplo, cambie los atributos de tiempo del archivo **tarea.txt** a agosto 16 2003, 5:59p.m.

$ touch -t 200101201759 tarea.txt

El comando date

El comando date imprime la fecha y tiempo actual en una variedad de formatos.

Descripción

date [options] [+*format*]
date [opciones] [+*formato*]

Date imprime la fecha y hora actual a la salida estándar. Por ejemplo,

$ date

Sun Apr 22 19:14:23 CDT 2006

La fecha y hora (date y time) pueden ser especificadas en una gran variedad de formatos. Por ejemplo,

$ date +%D
04/22/04

Las especificaciones de formato son descritas más abajo. El comando es particularmente util cuando se escriben script del shell.

Las opciones del comando date se muestran a continuación.

Opción	Descripción
-u	Muestra el tiempo usando Greenwich Mean Time (GMT).

Note: Un superusuario puede establecer la hora del sistema usando una segunda manera del comando date.

Especificando el formato del comando date

La especificación del formato de date empieza con un símbolo de (+) y debe estar entre comillas dobles para evitar que el shell no interprete como caracteres especiales. El formato puede contener texto y caracteres especiales de formato que le indican valores al comando date. Caracteres especiales de formato empiezan con un símbolo de (%). Por ejemplo,

$ date +"Hoy es %A el %d de %h %Y"
Hoy es Sunday el 22 de Apr 2006

Incluye texto como "Hoy es" interlazado con caracteres especiales de formato como es %A, el cual imprime el día de la semana. Más adelante les presentamos una lista de caracteres especiales de formato usados para especificar fechas y hora.

Formatos de date

General

%D	Date en formato de MM/DD/YY (e.j. 04/23/01).
%x	Formado especifico a la localidad. (En US, es así MM/DD/YY. En Latinoamérica es DD/MM/YY.)

Formatos del Mes

%m	Mes del año (01-12).
%b	Nombre Abreviado del mes (Jan, Feb, ..., Dec).
%h	Lo mismo que %b (Nombre Abreviado del mes).
%B	Nombre completos del mes (January, February, ..., December).

Formatos del Día

%d	Día del mes (01 al 31).
%e	Días del mes (1 al 31). Números del un solo digito espaciados a un carácter.
%j	Día del año (001-366).

Formatos de los días de la semana

%a	Nombres de los días de la semana abreviados (Sun, Mon, ..., Sat).
%A	Nombres completos de los días de la semana (Sunday, Monday, ..., Saturday).
%w	Números de los días de la semana (0-6) empezando por el domingo (Sun=0, Mon=1, ..., Sat=6).
%u	Números de los días de la semana (1-7) empezando con el lunes (Mon=1, Tue=2, ..., Sun=7).

Formatos del Año

%y	Años en 2 dígitos (99,00,01).
%Y	Anos en cuatro dígitos (1999,2005,2006).

Formatos del Tiempo

General

%R	Tiempo en formato HH:MM usando reloj de 24-hora (e.j. 17:31).
%T	Tiempo en formato HH:MM:SS usando reloj de 24-hora (e.j. 17:31:26).
%r	Tiempo en formato HH:MM:SS AM/PM usando reloj de 12-hora (e.j. 05:31:26 PM).
%Z	Nombre de la zona horaria.
%X	Formato de tiempo en localidad especifica. (En US es HH:MM:SS reloj 24-hora).

Formato de Hora

%H	Hora en formato de 24-hora (00-23).
%k	Hora en formato de 24-hora (0 a 24). Un solo digito separado con un espacio.
%I	Hora en formato de 12-hora (01-12).
%l	Hora en formato 12-hora (1-12). Un solo digito separado con un espacio.
%p	AM o PM para indicar a.m. o p.m.

Formato de Minuto

%M	Minuto (00-59).

Formato de Segundos

%S	Segundos (00-61). 60 y 61 son usando por el sistema para rastrear segundos de salto o doble.

Formato Combinado de Fecha y Tiempo

%c	Formato especifico a localidad de fecha y tiempo. (En US, la salida es Sun Apr 22 11:56:37 2006)

Formato Especial

%n	Insertar una nueva línea. Por ejemplo, "%D%n%T" imprime la fecha en formato %D (MM/DD/YY) en una línea seguido por el tiempo en formato %T (HH:MM:SS) en una segunda línea.
%t	Inserta un tab.

Ejemplos

- **$ date**
  Sun Apr 22 20:49:00 CDT 2006

  Muestra la fecha y tiempo actual usando el formato de salida por defecto.

$ date +"%D"
04/22/01

Muestrea la fecha actual usando un formato especial. La especificación %D imprime la fecha en formato MM/DD/YY.

$ date +"%l:%M %p"
11:14 AM

Muestra la hora actuales usando el reloj de 12-hora seguido por dos puntos (:) entonces los minutos actuales seguido por AM o PM.

$ date +"It is %r on %A the %d of %h %Y"
It is 08:54:12 PM on Sunday the 22 of Apr 2006

Muestra la fecha actual usando un formato especificado. El formato de fecha incluye texto “Así como este" combinado con caracteres especiales de formatos como %r cual indica la hora usando el reloj de 12-horas.

$ date -u
Mon Apr 23 01:55:08 UTC 2006

Muestra la fecha usando Greenwich Mean Time.

$ date +"Time:%tHour%t%H%n%tMinute%t%M%n%tSecond%t%S"

Time: Hour 20
Minute 59
Second 48

Muestra las horas, minutos y segundos. Use carácter de tabs (%t) y nueva línea (%n) para dar formato a la salida.

Ejemplos Avanzados

- Use substitución de comandos para agregar la fecha actual al nombre de un archivo. Por ejemplo, el comando **touch** puede ser usado para crear archivos vacíos.

  $ touch Archivo`date +%m-%d-%y`

  Crea un archivo vació de nombre Archivo*01-26-04* si se ejecuta el 26 de Enero del 2004. Si se ejecutase el 1 de Mayo del 2006, crearía un archivo de nombre Archivo*05-01-06*.

- El siguiente es un script del Bourne Shell que le agrega la hora, fecha y una lista de personas ingresadas (logged in) en una computadora a un archivo de nombre */var/log/QuienLog*.

  #!/bin/sh
  Archlog=/var/log/QuienLog
  date +"%T %A %D" >> $Archlog
  who >> $Archlog

Práctica5

Ejercicio 1

1) Copie el archivo ***/etc/passwd*** a su directorio home, y entonces utilice cat para ver su contenido.
2) Renómbrelo a ***usuarios*** utilizando el comando ***mv***.
3) Cree un directorio y nómbrelo ***programas*** y copie todo el contenido de ***/bin*** en el.
4) Borre todos los archivos del directorio programas.
5) Borre el directorio ahora vació ***programas*** y el archivo ***usuarios***.

Ejercicio 2

1) El comando touch puede ser utilizado para crear archivos vacíos. Como ejercicio cree uno de esta forma: **$ touch Linux.txt**
2) Despleguemos a pantalla los atributos de este archivo con el comando **ls**: **$ ls -l Linux.txt**
3) Espere unos minutos, y repita los dos pasos anteriores, y ver que cambie. ¿Que sucede cuando no especificamos el tiempo al comando como opción?
4) Intenta ajustar los atributos de tiempo de un archivo a valores futurísticos.
5) Una vez acabe; borra el archivo.

Capítulo6

GNU es cuestión de libertad y no de precios no se confundan somos libres usando software GNU, podemos hacer muchas cosas además de disfrutar usándolo podemos adaptarlo a nuestro gusto si tenemos los conocimientos necesarios para ello, no tenemos que adaptarnos a él necesariamente sino que lo podemos manejar a nuestro gusto.

Richard M. Stallman

Manejo de Archivos de Texto, de Entrada y Salida y Expresiones Regulares

Los Objetivos de este Capítulo son:

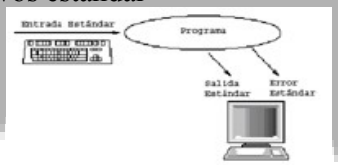
1. **Administración de la E/S**
2. **Los archivos STDIN, STDOUT, STDERR**
3. **Manejo de Tuberías y Redireccionamineto**
4. **Usos avanzados del Shell**
5. **Programación del Shell**

7.

Flujo de Texto (Streams), Tuberías y Redireccionar

Archivos Estándar

Los Procesos están conectados a tres archivos estándar



Muchos programas también acceden otros archivos.

Standard Input (Entrada Estándar)

Programas pueden leer data desde su archivo **standard input**
Abreviado **stdin**
Por defecto, este lee desde el teclado (keyboard)
Caracteres escritos a un programa interactivo (e.j., un editor de texto) van directo al **stdin**

Standard Output (Salida Estándar)

Programas pueden escribir data a su archivo de **standard output**
Abreviado **stdout**
Utilizado por la salida normal del programa
Por defecto esta salida es el terminal

Standard Error

Los programas pueden escribir data a su **standard error**
El Standard error es similar al standard output, pero es utilizado para mensajes de errores y advertencias
Abreviado **stderr**
Útil para separar la salida de un programa de la salida de sus errores

Por defecto se escribe al terminal
  - Así se consigue mezclar con la salida estándar.

Pipes - Tuberías

- Una tubería canaliza la salida de un programa a la entrada de otro
  - Permite que programas sean encadenados
  - Programas encadenados se ejecutan concurrentes
- Use la barra vertical: |
  - Mejor conocido como el carácter de la **tubería** o **'pipe'**
- Los programas no tienen que ejecutar nada especial para usar los **pipes**
  - Leen desde el **stdin** y escriben al **stdout** como es esperado
- Por Ejemplo, envié la salida de echo a la entrada del programa **rev** a través de una tubería (rev reversa cada línea de un archivo):

```
$ echo Viva Linux! | rev
!xuniL aviV
```

Conectando Programas a Archivos

- **Redirección** pasa la salida de un programa a la entrada de otro archivo
- El símbolo"<" indica el archivo que se va leer como entrada

  $ wc < tesis.txt
  - El archivo específico se convierte en la entrada estándar del programa.
- Este > símbolo indica la salida del archivo a escribir:

  $ who > usuarios.txt
  - La salida estándar del programa se dirige al archivo:
  - Si el archivo existe lo sobre escribe:
- Ambos pueden ser usados al mismo tiempo:

  $ grep < Linux-viejo.txt > Linux-nuevo.txt

Agregándole a Archivos

- Use los símbolos doble >> para agregar contenido al archivo:

  $ date >> fecha.txt
  - Agrega la salida estándar del programa al final del archivo existente
  - Si el archivo no existe, lo crea.

Redireccionando Múltiples Archivos

Archivos abiertos se asocian a un número identificador, llamados **file descriptors**
Estos pueden ser utilizados en argumentos de redirección.
Los tres archivos estándar tienen siempre el mismo número asignado:

Name Descriptor	**(Descriptor de Nombre)**
Standard input	0 (Entrada estándar 0)
Standard output	1 (Salida estándar 1)
Standard error	2 (Error estándar 2)

Redireccionar con el Descriptor de Archivos

- Redirección normalmente trabaja con el **stdin** y **stdout**
- Especifique diferentes archivos solo con colocar el número del descriptor de archivos antes del símbolo de redirección:
  - Para redireccionar el error estándar a un archivo:

    $ programa 2> archivo.txt

- Para combinar la salida del error estándar con la salida estándar:
  $ programa > archivo 2>&1
- Para guardar ambas salida de flujo:
  $ programa > stdout.txt 2> stderr.txt

- Descriptores 3-9 pueden ser conectados a archivos normales, utilizados mayormente en scripts shell.

El comando xargs

El utilitario *xargs* construye una lista de argumentos para pasársela a un comando *nix o GNU/Linux usando la entrada estándar. El comando **xargs** lee una cadena de texto y ejecuta otro programa con el texto como su argumentos. Casi siempre el texto de entrada es una lista de nombres de archivos para pasársela a un programa de procesar datos

Descripción

xargs [options] [*command*]
xargs [opciones] [*comando*]

El comando *xargs* crea una lista de argumentos para un comando desde la entrada estándar. Es típicamente usado con una tubería. Por ejemplo,

$ find ~ -name '*.txt' print | xargs cat

Ejemplo: Si hay demasiados archivos en un directorio para eliminarlos uno a la vez, puedes utilizar **xargs** para borrarlos diez a la vez:

$ find /tmp/borrarlos/ | xargs -l10 rm –f

El comando find busca en todo el directorio home por archivos que sus nombres terminen en .txt. El comando *xargs* agrupa todos los nombres de archivos de la salida del comando find que se lo pasa por la tubería a xargs y así le confecciona una lista que se la pasa al comando cat; el cual procede a imprimirlo a pantalla.

En mucho de los shells existe un límite en el número de argumentos permitidos en una línea de comando. Si la lista de argumentos leída por *xargs* es más larga que el número máximo permitido por el shell, el comando *xargs* agrupara los argumentos en grupos más pequeños y ejecuta el *comando* por separado para cada grupo de argumento. Dependiendo de las opciones usadas con *xargs*, los argumentos pueden ser procesados en grupos más pequeños (por ejemplo, uno a la vez).

Si no se especifican comandos, *xargs* funciona similar al comando echo y imprime la lista de argumentos a la salida estándar.

Opciones

Opción	Descripción
-n#	Ejecuta un comando una vez por cada número (*#) de* argumentos. Por ejemplo, -n2 agrupa los argumentos en grupos de dos o menos y ejecuta comandos en cada grupo de argumento.
-l#	Ejecuta un comando una vez por cada número (*#) de* lineas de entrada. Por ejemplo, -l1 crea un grupo de argumentos por cada una de la lineas de entrada y ejecuta los comandos en cada grupo de argumento.

-i	Normalmente *xargs* coloca los argumentos de entrada al final del comando. Usado con la opción -i, *xargs* reemplaza todas las instancias de {} con los argumentos de entrada. En la mayoría de los sistemas deberá colocar una barra invertida o backslash (\) antes de cada llave para evitar que los caracteres especiales sean interpretados.
-t	Hacerle echo a cada comando antes de ejecutarlo.
-p	Pregunta al usuario en el prompt antes de ejecutar cada *comando*.

Nota: No todos los *nix suportan estas opciones de *xargs*. Los de GNU/Linux soporta más opciones que los demás sabores de *nix. Revise sus páginas man.

Ejemplos

Xargs Básico

1. *El comando xargs* puede ser usado para leer la lista de argumentos de un comando desde la entrada estándar. A menudo los argumentos son listas de nombre de archivos pasados a *xargs* vía una tubería. Por ejemplo,

   $ ls f*
   Archivo1 Archivo2 Archivo3

   Tenemos tres archivos en el directorio actual que sus nombres empiezan con la letra *A*. El siguiente ejemplo imprime el contenido de cada archivo a la pantalla.

   $ ls A* | xargs cat
   Contenido de Archivo1...
   Contenido de Archivo2...
   Contenido de Archivo3...

   El comando *xargs* toma la salida del comando ls, "*Archivo1 Archivo2 Archivo3*", y la usa como argumentos para el comando *cat*, creando real y efectivamente el comando ***"cat Archivo1 Archivo2 Archivo3".***

   Note que usar el comando *xargs* es diferente que enviar la salida por una tubería directamente al comando *cat*. Por ejemplo,

   $ ls A* | cat
   Archivo1
   Archivo2
   Archivo3

   Usado sin argumentos, *cat* lee la entrada estándar (en este caso los nombre de los archivos del comando *ls*) y imprime el resultado a la pantalla.

2. Los argumentos leídos desde la entrada estándar pueden seguir opciones o otros argumentos. Por ejemplo:

   $ ls A* | xargs grep -i 'instalar linux' Instrucciones.txt
   Instrucciones.txt: Antes del jueves debes instalar linux y configurarlo.
   Archivo2: Es necesario instalar linux primero.

   El comando *xargs* combina el comando "*grep -i 'instalar linux' Instrucciones.txt*" con la salida del comando *ls*, creando el comando:

grep -i 'instalar linux' Instrucciones.txt Archivo1 Archivo2 Archivo3

La opción de *grep -i* y el argumento *Instrucciones.txt* son escritas en la línea de comandos pero los argumentos *Archivo1*, *Archivo2* y *Archivo3* son leídos desde la entrada estándar.

3. **$ find ~ -name 'prog1*' print | xargs cat > prog1.all**

   El comando *find* busca en el directorio home completo por archivos que sus nombres empiezan con *prog1*. El comando *xargs* agrupa todos los nombres de archivos en una sola lista de argumento para el comando *cat*. La salida del comando *cat se guarda en un archivo nombrado prog1.all* usando la redirección de salida.

Xargs vs. Substitución de Comandos – Procesar Líneas de Comandos Larga

1. La substitución de comandos nos permite usar la salida de un comando como un argumento de otro comando. Cuando parte de un comando se encierra entre comillas simple, el shell evaluara este texto como un comando separado e insertara la salida dentro de la sintaxis del comando original. Por ejemplo, el siguiente comando usa la substitución para buscar todos los archivos regulares en el directorio actual por la cadena "linux".

   $ grep 'linux' `find . -type f -print`
2. *Xargs* efectúa una función muy similar a la de substitución de comandos. A continuación un ejemplo que efectúa lo mismo que el anterior.

   $ find . -type f -print | xargs grep 'linux'
3. En algunos casos la substitución de comandos creara una línea de comandos muy larga para los sistemas *nix. Por ejemplo, intente buscar dentro de cada documento en el directorio root por la cadena de texto '*No Puede*'.

   $ grep 'No Puede' `find / -type f -print`
   grep: too many arguments

   Grep retorna un error y no termina la búsqueda. Note que el total de número de argumentos permitido en la línea de varia entre los shells. Este limitante no es parte de GNU/Linux ya que se usa el bash shell por defecto y no tiene ese problema.
4. *Xargs* pasa los argumentos en batches los cuales son suficientemente pequeños para no exceder este máximo permitido por el sistema. Por ejemplo, a diferencia del ejemplo anterior, el siguiente comando no retornaría el mismo error aunque se este ejecutando en el mismo sistema operativo y el mismo shell.

   $ find / -type f -print | xargs grep 'No Puede'
   El comando *xargs* permitirá a *grep* procesar más argumentos de los que puede normalmente manejar.

***Xargs* Características de Echoing**

1. Usado sin un comando, *xargs* funciona similar al comando echo. El agrupa las líneas de entrada y las imprime a la salida estándar. Por ejemplo,

   $ cat Archivo1
   línea 1 de Archivo1
   línea 2 de Archivo1
   línea 3 de Archivo1

   Ahora trate

$ cat Archivo1 | xargs
línea 1 de Archivo1 línea 2 de Archivo1 línea 3 de Archivo1

Note como *xargs* ha agrupado líneas separadas juntas. Si *Archivo1* fuese un archivo bien largo entonces *xargs* hubiese creado más de un grupo de data para mandar a la salida estándar. Por ejemplo,

$ wc -l archivo-largo.txt
4012

El archivo, *archivo-largo.txt*, tiene 4012 líneas. Ahora trate

$ cat archivo-largo.txt | xargs > xarch-grande
$ wc -l xarch-grande
8
La salida de *xargs* es almacenada en el archivo *xarch-grande* el cual solo tiene ocho líneas. *Xargs* agrupo la salida en grupos lo bastantemente pequeños para que el shell lo pudiese manejar sin producir errores. En este caso en particular los grupos son ocho.

2. La característica de echo del comando xargs es particularmente útil al combinar la salida de múltiples comandos. Por ejemplo, imprime la fecha de hoy:

   $ date +%D
   08/15/01

   Este otro ejemplo imprime el monto total de disco usado en el directorio home del usuario.

   $ du -s ~
   2007 /home/miguel

   Ahora ejecutemos ambos comandos a la vez

   $ date +%D ; du -s ~
   08/15/01
   2007 /home/miguel

   El siguiente comando usa una tubería y el comando *xargs* para agregar la salida de ambos comandos en una línea en el archivo log.

   $ (date +%D ; du -s ~) | xargs >> log
   $ cat log
   ...
   01/15/2004 2007 /home/miguel

Ejecute un Comando cada N Palabras o Líneas de Entrada

1. La opción *-n#* con x*args* ejecuta un comando con hasta n (número) # de argumentos. Por ejemplo,

   $ ls | xargs -n1
   Archivo1
   Archivo2
   Archivo3
   Archivo4

Usando la opción -n1, xargs procesa solamente un argumento a la vez, mientras que usando la opción -n3 , xargs agrupa hasta tres argumentos a la vez.

$ ls | xargs -n3
Archivo1 Archivo2 Archivo3
Archivo4

2. Despliega el contenido de un archivo una palabra por línea.

   $ cat filename | xargs -n1
3. la opción *-l#* con el comando *xargs* ejecuta un comando cada # de líneas de entrada. Por ejemplo,

   $ cat Archivo1
   línea 1
   línea 2
   línea 3
   línea 4

   El archivo *Archivo1* tiene cuatro líneas.

   $ cat Archivo1 | xargs
   línea 1 línea 2 línea 3 línea 4

   Sin opciones *xargs* agrupa las líneas de entrada en el argumento más largo que el shell pueda procesar. En este caso todas las líneas son agrupadas en una sola lista de argumentos. Use *-l2* para agrupar cada dos líneas de entrada juntas.

   $ cat Archivo1 | xargs -l2
   línea 1 línea 2
   línea 3 línea 4

Posicione Argumentos de la Entrada Estándar junto con Otros Argumentos

1. Típicamente *xargs* coloca los argumentos de entrada al final del comando. Usado con la opción *-i*, *xargs* reemplaza todas las instancias de {} con los argumentos de entrada. En la mayoría de los sistemas debe colocar una barra invertida o backslash antes de cada llave para que el shell no lo interprete como caracteres especiales. Por ejemplo, el siguiente comando mueve todos los archivos en *Directorio1* al *Directorio2*.

   $ ls Directorio1 | xargs -i mv Directorio1/\{\} Directorio2/\{\}
2. En el directorio actual existen tres archivos y sus nombres terminan con la extensión *.ascii*.

   $ ls *.ascii
   Archivo1.ascii Archivo2.ascii Archivo3.ascii

   El siguiente ejemplo renombramos todos los archivos que sus nombres terminan con *.ascii* para que sus nombres terminen con *.txt*.

   $ ls *.ascii | xargs -i basename \{\} .ascii | xargs -i mv \{\}.ascii \{\}.txt
   Como es que este comando trabaja? El comando basename imprime el nombre del archivo menos la extensión. Por ejemplo,

   $ basename Archivo1.ascii .ascii
   Archivo1

Así

$ ls *.ascii | xargs -i basename \{\} .ascii
Archivo1
Archivo2
Archivo3

Imprime cada nombre de archivo sin la extensión *.ascii*. Esta salida entonces es enviada al segundo comando *xargs* cual crea los comandos

mv Archivo1.ascii Archivo1.txt
mv Archivo2.ascii Archivo2.txt
mv Archivo3.ascii Archivo3.txt

Imprimir o Cuestionar Antes de Ejecutar los Comandos

1. Usado con la opción *-t*, *xargs* le hace echo a cada comando antes de ejecutarlo. Por ejemplo, el siguiente comando mueve todos los archivos en el *Directorio1* al *Directorio2*.

   $ ls Directorio1 | xargs -i -t mv Directorio1/\{\} Directorio2/\{\}
   mv Directorio1/Archivo1 Directorio2/Archivo1
   mv Directorio1/Archivo2 Directorio2/Archivo2
   mv Directorio1/Archivo3 Directorio2/Archivo3

2. Usado con la opción *-p*, *xargs* le pregunta en el prompt al usuario antes de ejecutar cada comando. Por ejemplo,

   $ ls Directorio1 | xargs -i -p mv Directorio1/\{\} Directorio2/\{\}
   mv Directorio1/Archivo1 Directorio2/Archivo1 ?...y
   mv Directorio1/Archivo2 Directorio2/Archivo2 ?...n
   mv Directorio1/Archivo3 Directorio2/Archivo3 ?...y

   Los archivos *Archivo1* y *Archivo3* fueron movidos pero no el *Archivo2*.

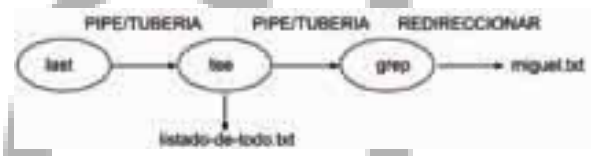
3. Use la opción que cuestiona (-p), para elegir cuales archivos en el directorio actual deben ser comprimidos.

   $ ls | xargs -n1 -p compress
   compress largef1 ?...y
   compress largef2 ?...y
   compress smallf1 ?...n
   `compress smallf2 ?...n`

El comando tee

El programa **tee** hace una tubería en 'T'
Copia data desde el **stdin** al **stdout**, y también a un archivo
Es como combinar > y a la |
Por Ejemplo, para guardar los detalles de los ingresos de todos los usuarios al sistema, y guardar los de Miguel en especial en un archivo separado, ejecute:

$ last | tee listado-de-todos.txt | grep miguel > miguel.txt
tee grep last miguel.*txt*
listado-de-todos.txt



Buscar en Archivos con Expresiones Regulares

El comando grep

El comando grep es una herramienta poderosa y flexible que busca cadenas de texto en los archivos.

Descripción

grep [*options*] '*pattern*' [*file* ...]
grep [*opciones*] '*patrón*' [archivo ...]

El comando *grep* busca en uno o más archivos por patrones de texto y imprime todas las líneas que contienen ese patrón. Si no se especifica, *grep* lee desde la entrada estándar. Si más de un archivo es especificado, el nombre del archivo es impreso antes de las líneas que se igualan al patrón. Por ejemplo,

$ grep 'Hola' *
Memo: Hola Todos
Carta: Hola Miguel,
Carta: Solo te escribo estas líneas para saludarte y decirte Hola.

Imprime tres líneas que contienen la cadena de texto *Hola* desde dos archivos en el directorio actual, *Memo* y Carta.

Buscar Archivos con grep

El comando **grep** imprime líneas desde un archivo que concuerdan con un patrón dado
Por Ejemplo, para buscar una entrada en el archivo de contraseñas ***/etc/passwd*** relacionado con **'miguel'**:

$ grep miguel /etc/passwd

El **grep** tiene muchas opciones útiles:

- **-i** para que el la búsqueda sea caso-insensitivo
- **-r** busca en directorios recursivamente
- **-l** imprime solo el nombre del archivo que contiene el patrón buscado
- **-c** imprime el número de aciertos en cada archivo del patrón
- **-n** enumera las líneas de la salida estándar del patrón buscado
- **-v** iguala el reverso del patrón, imprime las líneas que no concuerdan

Igualar Patrones

- Use grep para buscar patrones, así como otras cadenas de caracteres simples
- Los patrones se expresan como expresiones regulares
- Algunos caracteres de puntuación tienen significados especiales

Por Ejemplo: este ejemplo es una manera mejor de buscar la entrada de Miguel en el archivo contraseñas:

$ grep '^miguel' /etc/passwd

- El carácter **(^)** ancla al patrón a el principio de la línea

De la misma manera que, el símbolo **$** actúa como un **ancla** cuando aparece al final de una cadena, así logrando que el patrón iguale solo al final de las líneas

Igualar Patrones Repetidos

Algunos caracteres especiales de **regexp** también son especiales para el shell, y por esto necesitan estar protegidos con comillas **(")** o barras invertidas **(\)**
Podemos igualar un patrón repetido solo con sumarle un modificador:

$ grep -i 'parte \.*'

El punto **(.)** por si solo iguala cualquier carácter, así es que para igualar un punto debemos escaparlo con la barra invertida \
El comodín ***** iguala los caracteres siguientes sin importar el número de caracteres empezando con cero
Similarmente, el modificador \+ iguala una o más veces

Igualando Patrones Alternativos

Múltiple patrones pueden proveer alternativas, separadas con \|, por Ejemplo:

$ grep 'linux\|debian\|redhat' sistemas.txt

El comando previo busca líneas que igualan por lo menos una palabra de las tres
Use **\(...\)** para esforzar precedencia:

$ grep -i '\(linux\|debian\|redhat\) distros' sistemas.txt

Use corchetes para crear una **clase de carácter**:

$ grep '[Mm]iguel [Bb]loggs' usuarios.txt

Cualquier singular carácter que iguala desde la clase; y rangos de caracteres pueden expresarse así: **'a-z'**

Sintaxis de Expresiones Regulares Extendidas

- El comando egrep ejecuta grep en una modo diferente
  - Lo mismo que **grep -E**
- Caracteres especiales no tienen que ser marcados con \
  - Así pues que \+ se escribe +, **\(...\)** se escribe **(...)**, etc
  - En el extendido **regexps**, \+ es un literal +

Las opciones más comunes de grep desde la línea de comandos son:

Opción	Descripción
-i	Ignora distinción de caso mayúscula/minúscula.
-n	Imprime líneas que igualan y su número de línea.
-c	Imprime solamente el conteo de las líneas que igualan.
-l	Imprime los nombres de los archivos con líneas que igualan pero no la línea misma.

-h	Imprime las líneas que igualan pero no el nombre de los archivos.
-v	Imprime todas las líneas que no igualan el *patrón*.
-s	Suprimir los mensajes de error de archivos no-existente o no-legible.

Los patrones de *grep* están basados en un limitado número de expresiones regulares. Expresiones regulares proveen habilidades de igualar caracteres incluyendo el uso de comodines (wildcards), igualando rango de caracteres y buscando por el inicio o fin de líneas. Por ejemplo, el símbolo caret (^) indica el principio de una línea, así se listan todas las líneas que empiezan con *Hola*.

```
$ grep '^Hola' *
TexasMemo: Hola a Todos
letter:Hola Miguel,
```

Algunas de las características útiles de las expresiones regulares se muestran a continuación:

Símbolo	**Significado**
^	Iguala el principio de una línea.
$	Iguala el fin de una línea.
[...]	Iguala uno desde un conjunto de caracteres.
[^...]	Iguala cualquier carácter no encerrado en llaves.
[*n-m*]	Iguala cualquier carácter en el rango expresado por *n-m*.
.	Iguala cualquier carácter único excepto el de nueva línea.
*c**	Iguala cualquier número de caracteres después del carácter ***c***.
.*	Iguala cero o más ocurrencias de cualquier carácter.
\{*n*\}	Iguala exactamente *n* ocurrencias del carácter anterior o expresión regular.
\{*n*,\}	Iguala por lo menos *n* ocurrencias del carácter anterior o expresión regular.
\{*n*,*m*\}	Iguala cualquier número entre *n* y *m* del carácter anterior o expresión regular. Nota: *n* y *m* deben estar inclusivamente en el rango de 0 y 256.
\	Precediendo cualquier carácter especial com una barra invertida o backslash (\) inhabilita su significado.

Las expresiones regulares deben estar entre comillas sencillas para prevenir que el shell los interprete como caracteres especiales.

Ejemplos

- Busque la cadena de texto Ivellise en Archivo.txt. Cada línea de este archivo que contiene la cadena Ivellise será impresa en pantalla.

$ grep 'Ivellise' Archivo.txt

- Busque *Ivellise* en todos los archivos del directorio actual.

$ grep 'Ivellise' *

- Liste los nombres de los archivos en el directorio actual que contengan la cadena de texto *Ivellise*. Esta sentencia solo listara los nombres de los archivos, no las líneas individuales que contienen el string *Ivellise*.

$ grep -l 'Ivellise' *

- Busque la cadena de texto *"yo estudio Linux"* en todos los archivos en el directorio actual que sus nombres terminan con *.txt*. Ignora la distinción de mayúscula/minúscula de los caracteres.

$ grep -i 'yo estudio linux' *.txt

- Busque la cadena de texto *"final de la oración. Empieza"* in *Archivo.txt*.

$ grep 'final de la oración. Empieza' Archivo.txt

La barra invertida (backslash (\)) antes del punto (**.**) le dice a grep que ignore el significado del carácter especial punto.

- Busque Mozilla en todos los archivos en el directorio actual que tienen nombres que empiezan con Naveg. Ignore la distinción de caracteres mayúscula/minúscula. Imprime las líneas que igualan pero no los nombres de los archivos.

$ grep -i -h 'Mozilla' Naveg*

- Busque la cadena *java* en el archivo Compiladores. Imprime las líneas que igualan y sus números de líneas a la pantalla.

$ grep -n 'java' Compiladores

- Lista todas las líneas en el archivo Compiladores que no contienen la cadena java.

$ grep -v 'java' Compiladores

- Cuenta el número de líneas en el archivo *Compiladores* que contienen la cadena de texto *java*.

$ grep -c 'java' Compiladores

- Cuenta el número de líneas en el archivo *Compiladores* que no contienen la cadena de texto *java*.

$ grep -c -v 'java' Compiladores

- Lista las líneas que contienen cualquiera de las cadenas de texto *Linax*, Linex, Linix, Linux etc.,.

$ grep 'Lin*x' archivo

- Lista las líneas que contienen cualquier de los strings *bid*, *bud*, *bed*, etc., pero no a *bd*, *band* o *lid*.

$ grep 'b.d' archivo

- Lista líneas que contienen cualquier de las cadenas de texto *bd*, *bid*, *bud*, *band*, etc. No *bank*.

$ grep 'b.*d' archivo

- Liste todas las líneas que empiezan con el carácter *#include* en todos los archivos que su nombre terminan con extensión *.c*.

$ grep '^#include' *.c

- Liste todas las líneas que terminan con kernel en el archivo tutorial.

$ grep 'kernel$' tutorial

- Busque todas las líneas que contienen la cadena *Urgente* o *urgente* en el archivo Instalacion.txt. Muestre el número de las líneas que igualan la búsqueda de cualquier de los patrones.

$ grep -n '[uU]rgente' Instalacion.txt

- Liste todas las líneas en el archivo Instalacion.txt *que incluye bad*, *bed*, *bid*, o *bud* pero ni *bod* o *bend*.

$ grep 'b[aeiu]d' Instalacion.txt

- Busque todas las líneas en el archivo LEEME *que incluye un solo digito*.

$ grep '[0-9]' LEEME

- Busque todas las líneas en el archivo Instalacion.txt que incluyen una letra mayúscula.

$ grep '[A-Z]' Instalacion.txt

- Liste todas las líneas que contienen la cadena de caracteres *bed*, *bud*, *b1d*, etc pero no a *bd*, *bid* o *bond*.

$ grep 'b[^i]d' Instalacion.txt

- Liste todas las líneas en el archivo Instalacion.txt que empiezan con una letra mayúscula o minúscula.

$ grep '^[A-Za-z]' Instalacion.txt

- Liste todas las líneas que contengan las cadenas de caracteres Stalman o Stallman pero no encontraría a Staman o a Stalllman.

$ grep 'Stal\{1,2\}man' Leeme.txt

- Liste todas las líneas de que contengan un número de teléfono del formato (nnn) nnn-nnnn.

$ grep '([0-9]\{3\}) [0-9]\{3\}-[0-9]\{4\}' listado.tel.txt

Ejemplos Avanzados

Guarde todas las líneas del archivo *log* que empiezan con error o *dump* en un archivo nuevo y llámelo *problemas.txt*

$ grep '^error' log > problemas.txt
$ grep '^dump' log >> problemas.txt

La primera sentencia del comando grep lista las líneas que empiezan con la palabra *error* y redirecciona la salida a un archivo y lo llama *problemas.txt*. El segundo comando gre*p* lista las líneas que empiezan con *dump* y agrega su salida al mismo archivo *problemas.txt*.

busque todos los archivos en el directorio actual que sus nombres terminan en *.txt* y que contienen la cadena de texto *"Saludo todos"* al principio de la línea. Ni distinga entre mayúsculas y minúsculas e imprima solamente los nombres de los archivos que igualen la búsqueda.

$ grep -i -l '^Saludo todos' *.txt

Use el comando find para efectuar la misma búsqueda en todos los archivos en su árbol de directorio completo y empezando por su directorio home.

$ find ~ -name '*.txt' -exec grep -i -l '^Saludo todos' \{\} \;

Liste todos los archivos en el directorio actual que no contengan la cadena de texto *error*.

$ grep -c 'error' * | grep ':0$'

El primer comando *grep* lista cada archivo en el directorio seguido por dos puntos (**:**) y el número de veces que la cadena *error* aparece en el archivo. La salida es pasada por tubería (piped) al segundo comando *grep* cual lista todas las líneas que terminan en**: 0** (Mejor dicho que no contienen *error*).

Busque todos los archivos en el directorio actual con nombre que terminan con ***.c*** y por tubería pásele la salida del comando ls -1.

$ ls -1 | grep '\.c$'

El comando ls -1 lista los archivos en el directorio actual en una sola columna. El símbolo de **$** en el patrón de *grep* especifica que es al final de la línea mientras que el backslash, (\), evita que grep interprete el punto (**.**) como un carácter especial.

Liste todos los directorios que tienen permisos de ejecución para los usuarios los “otros”.

$ ls -l | grep 'd.....x'

El comando *ls -l* efectúa un lista largo de los archivos incluyendo el bloque de los permisos. El patrón que les pasamos a grep busca por una cadena de caracteres que empieza con **d** y tiene exactamente ocho caracteres no especificados y entonces una **x** al final. Esto encontrara los bloques de permisos que empiezan con **d**, especificando que es un directorio, y que termina con una **x**, especificando permisos de ejecución para los otros.

Cuente el número de usuarios que usan el shell bash en su sistema.

$ grep -c /bin/bash /etc/passwd

El comando sed

- El comando sed lee líneas de entrada, ejecuta comandos sobre ellas, y lo escribe a la salida estándar
- El comando sed usa expresiones regulares como patrones en substituciones
  - El comando sed utiliza el mismo sintaxis de expresiones regulares como el comando grep
  - Por Ejemplo, para hacer que sed coloque un # al principio de cada línea:

$ sed -e ’s/^/#/’ < entrada.txt > salida.txt

El comando sed tiene simple utilidades de substituciones y de traducir, pero puede también ser utilizada como un lenguaje de programación

Uso del Shell Avanzado

Más Acerca de las Comillas

- El shell tiene tres mecanismos diferentes de para usar comillas:
  - Comillas Sencillas
  - Backslashes o Barras Invertidas
  - Doble Comillas

Comillas: Sencillas

Ponerle comillas sencillas a texto, lo protege de interpretación especial del shell:

$ xmms ’Juan Luis - Burbujas de Amor.mp3’
$ rm ’b*bujas de Amor.mp3’

Pero comillas sencillas (obviamente) no protegen a comillas sencillas mismas

- Así es que no puedes proteger algo así: El pregunto, "Donde esta la computadora." Con comillas sencillas.

Citar: Backslashes

Puedes poner un backslash \ en frente de un carácter simple para apagarle su significado especial:

$ echo M\&S
$ xmms Fernando\ Villalona\ -\ Dominicano\ Soy.mp3
$ mail -s C:\\MSDOS.SYS Administrador@Ejemplo.com

Citar: Comillas Doble

Poner comillas doble alrededor de algo, protege el contenido dentro de ellas de la interpretación del shell.

- Un símbolo de **$** retiene la interpretación especial

- Así como las backticks ‘
- El símbolo de **!** no puede ser escapado con comillas doble

Un backslash puede ser usado dentro de comillas doble para selectivamente deshabilitar la interpretación especial de **$**, **‘** y \:

```
$ mail -s "C:\\MSDOS.SYS" Administrador@Ejemplo.com
$ echo "El precio es $precio RD\$"
```

Ponerle un backslash en frente de cualquier cosa te devuelve ambos caracteres:

```
$ echo "\*/"
\*/
```

Citar: Combinar los Mecanismos de usar Comillas

Puedes construir un argumento para un comando de trozos de diferente texto de texto entre comillas.
Solo coloque los trozos de lado a lado sin dejar espacio de por medio:

```
$ echo "Comillas Dobles"'.comillas sencillas.'sin-comillas
Comillas Doble.comillas sencillas.sin-comillas
$ echo 'Duarte dijo, "Dios Patria y Libertad."'
Duarte dijo, "Dios Patria y Libertad."
```

Raramente necesitado - El último ejemplo es escrito mejor así:

```
$ echo "Duarte dijo, \"Dios Patria y Libertad.\""
```

Recapitular: Especificar Archivos con Comodines

El asterisco * en patrones glob puede significar una secuencia de caracteres:

```
$ ls -l *.txt
-rw-rw-r-- 1 miguel admin 108 Nov 16 13:06 reporte.txt
-rw-rw-r-- 1 miguel admin 345 Ene 18 08:56 notas.txt
```

El asterisco * de por si solo expande a todos los archivos y directorios en el directorio actual
Expansiones Globs son ejecutadas por el shell

- Los programas no reconocen cuando los argumentos son expresiones Globs

Expresiones Glob a Archivos dentro de Directorios

- Puedes utilizar expresiones globs para acceder a archivos dentro de los directorios:

```
$ ls Cuentas/199*.txt
Cuentas/1997.txt  Cuentas/1998.txt  Cuentas/1999.txt
$ ls ../fotos/*.gif
../fotos/logo.gif ../fotos/emblema.gif
```

También puedes usar expresiones globs para expandir nombres de directorios:

```
$ cd /usr/man && ls man*/lp*
man1/lpq.1.gz man1/lprm.1.gz man4/lp.4.gz man8/lpd.8.gz
man1/lpr.1.gz man1/lptest.1.gz man8/lpc.8.gz
```

Usar Expresiones Glob para Igualar un Carácter Simple

El asterisco * iguala cualquier secuencia de caracteres
Para igualar un solo carácter, use el **?**:

```
$ ls ?ouse.txt
```

Iguala mouse.txt y house.txt, pero no a grouse.txt
Útil para asegurarse que solo iguale archivos de nombre con cierto número de caracteres:

```
$ rm ???*.txt
```

Iguala los archivos que terminen en ***.txt*** y que tienen por lo menos tres caracteres antes del punto

Usar Expresiones Glob para Igualar Caracteres en Especial

En vez de igualar cualquier carácter singular, podemos gestionar para igualar de un grupo dado de caracteres
.[ch]** iguala cualquier archivo que termine con ***.c o ***.h***
***[0-9].txt** iguala cualquier archivo con un solo dígito antes del punto
Puedes usar un ^ como el primer símbolo en los corchetes para igualar cualquier carácter no listado
[^a-z]*.jpg iguala cualquier archivo JPEG que su nombre no empieza con minúscula
Para igualar cualquier archivo oculto excepto los directorios **.** y **..** : **.[^.]***

Generar Nombres de Archivos: {}

Puedes usar llaves {} para generar nombres de archivos:

```
$ mkdir -p Cuentas/200{1,2}
$ mkdir Cuentas/200{1,2}/{0{1,2,3,4,5,6,7,8,9},1{0,1,2}}
```

Puedes hasta combinar las dos líneas así:

```
$ mkdir -p Cuentas/200{1,2}/{0{1,2,3,4,5,6,7,8,9},1{0,1,2}}
```

O combinar la expansión de las llaves con comillas:

```
$ echo 'Hola '{Mundo,Saludo}\!''
Hola Mundo! Hola Saludo!
```

Llaves pueden ser utilizadas para generar cualquier cadena de caracteres, no solo nombre de archivos
Diferente a la expansión de globs - las palabras generadas no necesitan ser nombres de archivos y directorios existentes

Programación Shell

- El shell esta diseñada para ser ambos:
  - Un ambiente para ingresar comandos
  - Un lenguaje de programación simple
  - Comandos que se pueden ingresar en el prompt pueden ser incluidos en un archivo
- Características de programación incluyen: variables, bucles (incluyendo **for**), y funciones del shell
- El modelo de componentes de Unix hace que sea fácil crear scripts shell que puedan ejecutar tareas complejas
- Campos donde se encuentran aplicaciones en scripts shell incluyen:
  - Procesamiento de Texto
  - Automatización de tareas administrativas

Práctica6

Ejercicio 1

1) Prueba el ejemplo que se dio sobre las tuberías, filtrando por **rev** para invertir el texto
2) Prueba con otro comando que no sea echo, que produce salida (Ej., whoami).
3) Que sucede cuando reemplazas **rev** con **cat**? Prueba ejecutando **cat** sin argumentos y ingresando texto.

Ejercicio 2

1) Ejecute el comando **ls --color** en un directorio con varios archivos y directorios. Algunas distribuciones de GNU/Linux ya vienen para que el comando **ls** siempre use la opción **--color**, pero en este caso pásele la opción de forma explicita.
2) Pruebe ejecutando el mismo comando, pero envíe por tubería la salida a otro programa (Ej., cat o less). Debes notar dos diferencias en la salida. El comando **ls** detecta automáticamente si su salida va a un terminal (para ser observado a pantalla) o se direcciona a una tubería (para ser leído por otro programa).

Ejercicio 3

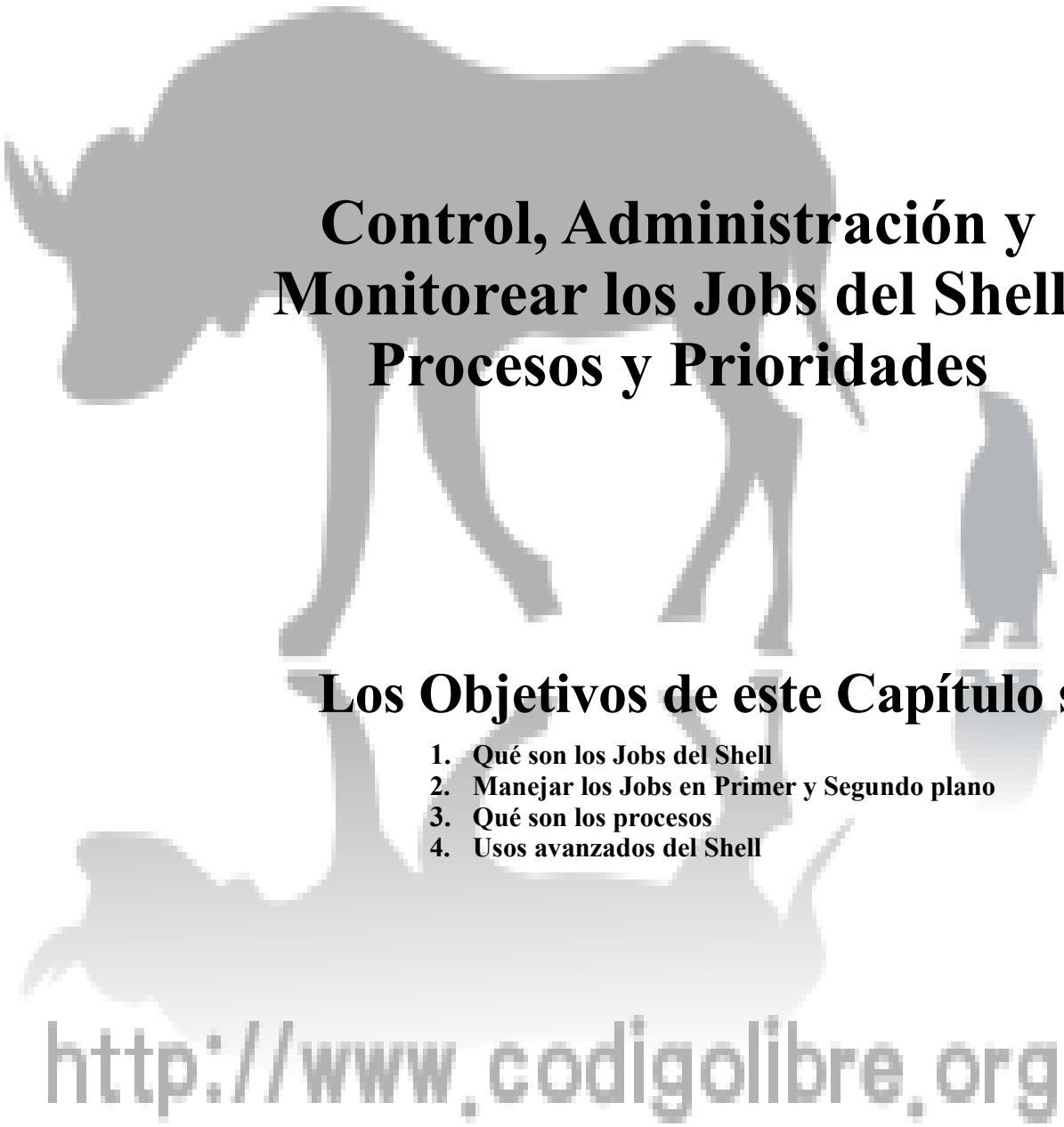
1) Use grep para encontrar información acerca del protocolo HTTP en el archivo ***/etc/services***
2) Este archivo contiene comentarios, que empiezan siempre con el símbolo **'#'**. Use a **grep** con opción **-v** para ignorar las líneas que empiezan con **'#'** y visualice el resto del archivo con **less**.
3) Agregue otro uso de **grep -v** a su tubería para remover líneas en blanco (iguala patrón**^$**).
4) Use sed (en la misma tubería) para remover la información después del símbolo **'/'** en cada línea, así solo dejara los nombres de los protocolos y los números de puertos que usa.

Ejercicio 4

1) Imprima el siguiente mensaje: *** VENDO $$$ ***.
2) Trate maneras diferentes de escapar las variables de entorno con: comillas simples, doble y backslashes.
3) Imprima con echo: 'Citar es Fácil en GNU/Linux', escape los espacios con comillas sencillas.
4) Use el patrón glob .[^.]* para listar los archivos ocultos en su directorio home
5) Para ver los shells disponibles, liste los programas en /bin cullo nombre termine en sh.
6) Use [] corchetes para listar archivos en /usr/bin que sus nombres empiezan con a, b o c.

Capítulo7

Hemos avanzado mucho desde que creamos el DOS, ahora todo es más atractivo visualmente.
—Bill Gates



Control, Administración y Monitorear los Jobs del Shell, Procesos y Prioridades

Los Objetivos de este Capítulo son:

1. **Qué son los Jobs del Shell**
2. **Manejar los Jobs en Primer y Segundo plano**
3. **Qué son los procesos**
4. **Usos avanzados del Shell**

Control de Job

Job Control

- Los shells ofrecen **control job**
  - La habilidad de parar, reiniciar, y enviar al **segundo plano** (background) los procesos en ejecución
- El shell te permite poner un & al final de la línea de comando para iniciarlo en el segundo plano
- También puedes presionar Ctrl+Z para suspender un trabajo ejecutándose en primer plano
- Trabajos suspendidos y enviados a segundo plano son asignados un número por el shell
- Estos números se pueden pasar como argumentos a comandos del shell de control de trabajos
- Estos comandos de Control-de-Trabajo incluyendo: jobs, fg, y bg

Los jobs

- El comando del shell **jobs** imprime los trabajos activos, su estatus y número de job:

```
$ jobs
[1]-   Stopped                 vim index.html
[2]    Running                 netscape &
[3]+   Stopped                 man ls
```

- Los números de los trabajos (**Job)** se dan en corchetes cuadrados
  - Al usarlo en combinación con otros controles de trabajos, necesitas precederles con un símbolo de por ciento, por Ejemplo %1
- Los trabajos marcados con + y - pueden ser accesados con **%+** o **%-** y también por sus números
  - **%+** es como el shell denomina el **trabajo actual** - el trabajo más recientemente activo
  - %- es el trabajo previo al trabajo actual

El Primer Plano fg

- Trae trabajos desde el **(background)** segundo plano al primer **(foreground)**
- Reinicia un trabajo suspendido, ejecutando el el primer plano
- **fg %1** - Trae el trabajo número 1 al primer plano
- **fg** sin argumentos opera sobre el trabajo actual

El Segundo Plano bg

- Reinicia un trabajo suspendido, ejecutándolo en segundo plano
- **bg %1** - Lleva el trabajo número 1 al segundo plano
- **bg** sin argumentos opera sobre el trabajo actual
- Ejemplo, después de ejecutar **gimp** y suspenderlo con **Ctrl+Z**, use **bg** para ejecutarlo en **background**

Crear, Monitorear, y Eliminar (Kill) Procesos

¿Que es un Proceso?

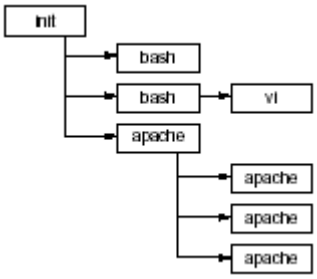
- El kernel considera cada programa ejecutándose en el sistema como un **proceso**
- Un proceso 'vive' durante su ejecución, con un tiempo de vida que puede ser corto o largo
- Se dice que un proceso 'murió' cuando el termina
- El kernel identifica cada proceso con un número conocido como un **id del proceso**, o **pid**
- El kernel mantiene un record de las propiedades de varios procesos

Propiedades de los Procesos

- Un proceso tiene un **id de usuario** (**uid**) y uno de grupo (**gid**) cual juntos especifican permisos que tienen
- Un proceso tiene un **id de proceso padre** (**ppid**) - cual es el **pid** del proceso que lo creo
    - El kernel inicia el proceso **init** con el **pid 1** al arranque o **boot-up**
    - Todos los demás procesos son hijos del proceso con el **pid 1**
- Cada proceso tiene su propio **directorio de trabajo** (**working directory)**, inicialmente heredado del proceso padre
- Existe un ambiente (**environment)** para cada proceso - una colección de variables de ambiente y sus valores asociados
    - El ambiente es normalmente heredado de su proceso padre

Procesos Padres e Hijos

- El proceso **init** es el padre de todos los procesos:



- (Apache inicia muchos procesos hijos para que ellos puedan servir requisiciones HTTP concurrentes)

Monitoreando Procesos: ps

- El comando **ps** nos da un vistazo a los procesos ejecutándose en el sistema en un momento dado
- Muy flexible en lo que muestra, y como:
  - Normalmente en un breve resumen de los procesos
  - Normalmente muestra solo los procesos que son de propiedad del usuario ejecutando
- Desafortunadamente, no utiliza sintaxis estándar de opciones
- En vez utilizar una liga de opciones con tres sintaxis:
  - BSD Tradicional **ps**: solo una letra sin guión
  - Unix98 **ps**: solo una letra pero precedido por un guión
  - GNU: una palabra o frase precedido por dos guiones **(--)**

Opciones de ps

- El comando ps tiene muchas opciones
- Algunas de las más comunes son:

Opciones	Descripción
-a	Muestra los procesos de otros usuarios
-f	Muestra los procesos en un formato de árbol ancestral
-u	Utiliza el formato de salida 'usuario', muestra nombres de usuarios y tiempo de inicio de los procesos
-w	Usa un formato más ancho de salida. Normalmente se recorta cada línea de la salida; cada uso de la opción **w** hace que la ventana sea más ancha
-x	Incluir procesos que no poseen el terminal de control
-e	Muestra información de todos los procesos
-l	Usa el formato 'largo' de salida
-f	Usa el formato 'completo' de salida
-C *comando*	Muestra solo el proceso asociado con el ***comando***
-U *usuario*	Muestra solo los procesos que ***usuario*** es el dueño

Monitorear Procesos: pstree

- Despliega una vista de los procesos en ejecución
- Siempre utiliza un desplegado tipo árbol, como **ps -f**
  - Por defecto solo muestra el nombre de cada comando
- Normalmente muestra todos los procesos
  - Especifique un número **pid** como argumento para mostrar un proceso en particular y sus descendientes
  - Especifique un nombre de usuario como argumento para mostrar el árbol de procesos del usuario

Opciones pstree

Opciones	Descripción
-a	Muestra los argumentos de los comandos
-c	No compactar sub-árboles idénticos en contenido
-G	Intenta utilizar caracteres de pantallas específicos al terminal

-h	Resalta los procesos ancestrales del proceso actual
-n	Ordena numéricamente por **pid**, y no alfabéticamente por nombre
-p	Incluir **pids** en la salida

Monitorear Procesos: top

- Muestra continuamente en pantalla completa, vista de actividad de los procesos en ejecución
  - Espera un lapso de tiempo para refrescar la pantalla y así crear la ilusión de ejecución en tiempo real
- Los procesos se muestran en orden descendiente de su uso de recursos del procesador
- También muestra tiempo que el sistema esta en uso, average de carga, estatus del CPU, y información de memoria

Opciones del comando top

Opciones	**Descripción**
-b Modo Batch -	envía una vista (snapshots) a la salida estándar
-n número	Sale después de mostrar número de vistas
-d número	Espera número de segundos entre las vistas
-i	Ignora los procesos inactivos (idle)
-s	Deshabilita comandos interactivos, puede ser peligroso si ejecuta como el root

Interactuando con el comando top

Key	**Comportamientos**
q	Sale del Programa
Ctrl+L	Redibuja la pantalla
h	Muestra pantalla de ayuda
k	Pide interactivamente un **pid** y una señal, y le envía la señal al proceso
n	Pide interactivamente por el número del proceso para mostrar la información; 0 (el por defecto) significa mostrar los que caben
r	Cambiar la prioridad (niceness) de un proceso
s	Cambia el número de segundos ha pausar entre actualizaciones. Se puede incluir fracciones de tiempo segundo (0.5, por Ejemplo)

Enviar Señales a los Procesos

- A un proceso se le puede enviar una señal por el kernel o otro proceso
- Cada señal es un mensaje muy simple:
  - Un número entero pequeño
  - Con un nombre de mnemónico
- Los nombres de las señales se escriben todos capitalizados, ejemplo INT
  - A menudo se escriben con la parte SIG como parte del nombre: SIGINT
- Algunas señales son tratadas especialmente por el kernel; otras tienen un significado convencional
- Existen unas 30 señales disponibles, no todos son muy útiles

Señales Comunes Para Uso Interactivo

- El comando **kill -l** lista todas las señales
- La siguiente son las más usadas:

Nombre	**Número**	**Significado**

INT	2	**Interrupt**-Para la ejecución. Enviada por el kernel cuando presionas Ctrl+C en un terminal.
TERM	15	"**Por favor termina**." Usada para pedirle a un proceso que cierre correctamente.
KILL	9	"**Matar!**" Forza al proceso que pare de ejecutar; sin dar oportunidad que el cierre sea apropiado.
TSTP	18	**Detenerse Temporalmente**. Enviada por el kernel al presionar **Ctrl+Z** en un terminal.
HUP	1	**Hang up**. Enviado por el kernel cuando ejecutas un logout, o se desconecta un modem. Convencionalmente utilizada por muchos daemons como instrucción para volver a leer el archivo de configuración.

Enviar Señales: kill

- El comando **kill** se usa para enviar una señal a un proceso
  - No es solo para terminar procesos en ejecución!
- Es un comando ejecutable normal, pero muchas shells también lo proveen como un comando interno
- Use **kill -HUP *pid*** o **kill -s HUP *pid*** para enviar un **SIGHUP** al proceso con este ***pid***
- Si no incluyes el nombre de la señal, el comando **kill** enviara un **SIGTERM**
- Puedes especificar más de un **pid** para enviarle señal a más de un proceso

Enviar Señales a los Daemons: pidof

- En sistemas Unix, procesos que proveen servicios de larga duración son referidos como **daemons**
- Típicamente Daemons tienen archivos de configuración (normalmente en ***/etc***) que controla sus comportamientos.
- La mayoría de los daemons leen su archivo de configuración solo al inicio del servicio
- Si el archivo de configuración cambia, tendrás que explícitamente decirle al daemon con el envío de una señal **SIGHUP**
- Puedes usar el comando **pidof** para investigar el **pid** de los **daemons**; por ejemplo, para pedirle a **inetd daemon** que recargue su archivo de configuración, ejecute:
  $ kill -HUP $(pidof /usr/sbin/inetd) - as root

El comando at

El comando at programa uno o más comandos de GNU/Linux para ser ejecutados más tarde en fecha y tiempo.

Descripción

at opciones1 tiempo [fecha] [+incremento] (forma 1)
at options1 time [date] [+increment]
at opciones2 [IDsTrabajo] (forma 2)
at options2 [JobsIDs]
Forma 1 del comando *at* se usa para especificar un tiempo y una fecha para ejecutar uno o más comandos Unix ejecutados hacia la entrada estándar. Por ejemplo,

```
$ at 1 am December 8
at> tar cvf ~/misdocumentos backup.tar
at> compress backup.tar
at> CTRL+d
warning: cmds will be executed with /bin/sh
job 976385752.a at Sat Dec  8 01:00:00 2005[1]
```

Ejecuta el comando tar seguido por el comando compress a las 1am Diciembre 8. No es necesario que este ingresado en el sistema a esta hora de ejecución. El utilitario *at* lee un comando por línea hasta llegar a la secuencia de teclas EOF (E n la mayoría de sistemas es **CTRL+d**).

Un grupo de comandos programados por *at* es colectivamente referido como un at-job. Cada at-job es asignado un número único llamado un jobID.

La forma 2 del comando *at* controla at-jobs que has sido previamente programados. Por ejemplo,

```
$ at -l
976385710.a     Sat Dec  8 01:00:00 2005
976385403.a     Sun Dec  9 17:54:00 2005
```

Lista los JobIDs y la programación del tiempo de ejecución de todos los at-jobs.

Opciones (Forma 1: Programar At-Jobs)

Opción	Descripción
-f file	Ejecute comandos listados en un archivo (file*)* y no desde la entrada estándar.
-m	Envía correo cuando el at-job se complete.
-q *queuename*	Programe trabajos en la cola (queue). *Queuename* (nombre de la cola) es una letra minúscula (*a* hasta *z*). Por defecto, los at-jobs serán programados en la cola (queue) *a*. Los Batch Jobs son típicamente programados en la cola *b*. Otros queuenames varían dependiendo de la implementación de Unix.
-t *time*	Especifica un tiempo de ejecución utilizando un formato igual al del comando touch. *Note: La opción -t no es soportada en todos los Sabores de Unix.*

Opciones (Forma 2: Administrar At-Jobs Previamente Programados)

Opción	Descripción
-l	Lista los jobID, queuename y tiempo de ejecución programado de todos los at-jobs en espera.
-q *queuename*	Cuando se úsala opción *-l*, *-q* limita el listado de solo esos at-jobs programados en la cola *queuename*.
-r *jobIDs*	Remueve at-jobs programado para que no ejecuten.

Especificar Time

hh[:mm] [am | pm]
: Las horas se le pueden dar con uno o dos dígitos. Los minutos son opcionales. La hora es especificada con un reloj de 24-hora al menos que se le agregue *am* o *pm*. Algunos ejemplos de times valido son **1**, **1:15**, **1:15 pm**, **1 am** y **16:55**.

now | noon | midnight
: Estas palabras claves pueden ser usadas en lugar de especificar horas y minutos. La palabra *Now* es a menudo seguido por una especificación de incremento.

Especificación de Date

Month Day[, *Year*]
Month puede ser el nombre del mes completo o abreviado con las primeras tres letras. Capitalizar no se toma en cuenta. Day es un valor numérico del día del mes y año y debe ser especificado con cuatro dígitos. *Note: La mayoría de los sistemas requiere una coma entre el día y el año; otros no permiten la coma.* Algunos ejemplos validos de dates son **january 8, 2007** y **December 8** y **jul 14**.

Weekday
Uno de los siete días deletreados o abreviado con las tres primeras letras del nombre del día. Capitalizar no se toma en cuenta. Los weekdays validos son **mon**, **tue**, **wed**, **thu**, **fri**, **sat** y **sun**.

today | tomorrow
Estas keywords son hoy | mañana y pueden ser usadas con date.

Especificar el Incremento

+ *n* [minute(s) | hour(s) | day(s) | week(s) | month(s) | year(s)]
Incrementa por *n* el tiempo especificado. Palabras clave indican que unidad de tiempo es agregada. Pueden ser singulares o plurales. Algunos ejemplos de incrementos validos son **+ 1 hour** y **+ 2 months**. Estos incrementos son utilizados más comúnmente cuando el valor de date is *now*. Por ejemplo, **now + 30 minutes** programa la ejecución en 30 minutos. La palabra clave ***next*** puede ser usada en lugar del incremento **+ 1**. Por ejemplo, **2pm next week** programa ejecución a las 2pm una semana desde hoy.

...un poco más sobre especificaciones de Time y Date

Si el día y año no son especificado, ***at*** ejecuta el comando el primer día que iguala la especificación de time/date. Por ejemplo, si escribe

$ at noon dec 8
Comandos...

El 7 Diciembre **at** ejecutara los comandos en la tarde del próximo día. El mismo comando ***at*** usado el 9 de Diciembre, no ejecutaría los comandos hasta el próximo año. El comando ***at*** retorna un error si la especificación de time es en el pasado

$ at noon mar 22, 1969
at: too late.

o si la especificación no tiene sentido. Por ejemplo,

$ at midnight sat dec 26
at: bad time specification

Retorna un error ya que ambos el **sat** y **dec 26** son especificadores del día.

¿Que sucede con la Salida de los Comandos?
Las salidas de Standard Output y Standard Error que producen los at-job son enviadas por email al usuario al menos que sean redireccionadas. Por ejemplo,

$ cat afile

Imprime el contenido del archivo afile a la salida estándar.

$ at now + 1 minute
at> **cat afile**

Le enviara por email el contenido del archivo *afile* al usuario en 1 minuto. Mientras que

$ at now + 1 minute
at> **cat afile > bfile**

Redireccionar la salida estándar del comando cat, guardándolo en el archivo *bfile* en vez de enviarlo por email. El archivo *bfile* se guardara en el directorio que usted estaba trabajando en el momento que ejecuto el comando **at**.

¿Que Shell Usa At?

Cuando se ejecuta un at-job, este invoca un shell nuevo para interpretar los comandos de la tarea. Dependiendo de sistema Unix que esta utilizando, ***at*** hará una de tres cosas para elegir un shell.

- Algunos sistemas revisan a ver si la variable de entorno SHELL esta definido y, si es así, utiliza ese shell.
- La mayoría de los sistemas GNU/Linux usan el Bourne Again Shell (bash).
- Algunos sistemas utilizan su shell por defecto de login.

En la mayoría de sistemas Unix, el comando *at* imprime un mensaje indicándole el shell que será utilizado al momento de ejecutar el at-job. Por ejemplo,

$ at 1 am December 8
los *comandos...*
warning: cmds will be executed with **/bin/sh**
job 976385752.a at Sat Dec 8 01:00:00 2005

Este mensaje nos indica que los comandos se ejecutaran usando el shell *sh* o mejor conocido como el Bourne shell. Note que este mensaje se escribe al error estándar y no a la salida estándar.

El shell retiene el directorio actual de trabajo (pwd), las variables de ambiente (con excepción de valores de terminales y consola) y valores de umask en efecto al momento de invocarlo.

¿Quien puede usar el comando at?

Acceso al comando ***at*** pueden ser restringido. Por ejemplo,

$ at now + 1 minute
at: you do not have permission to use at

Indica que el usuario no tiene permiso para usar el ***at***. Dos archivos, at.allow y at.deny determina cuales usuarios son permitidos ejecutar el ***at***.

Si *at.allow* existe, solo usuarios listados en el tienen permisos de usar *at*.
Si *at.allow* no existe y *at.deny* si, todos los usuarios excepto esos listados en *at.deny* tienen permiso para usar ***at***.
Si *at.deny* existe y esta vacío, todos los usuarios tienen permiso para usar ***at***.
Si no existe ni *at.allow* o *at.deny* entonces solamente el superusuario tiene permiso para usar **at**.

Los archivos *at.allow* y *at.deny* deben tener un userid por línea. En la mayoría de los sistemas Unix, ***at*** busca estos archivos en el directorio */usr/lib/cron*. Pero, algunos sistemas buscan el */var/at, o otros directorios, GNU/Linux coloca estos archivos /etc/at.allow y /etc/at.deny*. Revise su documentación local **man at** para más información.

Ejemplos

- Aquí presentamos algunos ejemplos de especificar date y time con ***at***.

$ at 17:30 feb 1, 2007
Ejecute a las 17:30 Febrero 1, 2007. Por defecto, horas y minutos usan un formato de 24-horas. Agréguele *am* o *pm* para usar el formato de reloj de 12-horas.

$ at 5:30 pm feb 1, 2007
Equivalente al ejemplo de arriba con el tiempo especificado con el tiempo en formato de reloj de 12-horas.

$ at -t 0202011730
Equivalente al ejemplo de arriba pero con la opción -t para pasarle los valores de date y time en formato del comando **touch**. El formato de touch de time es *[YY]MMDDhhmm* (dos dígitos año [opcional], mes, día, hora, minuto).

$ at 5 am feb 1, 2007
Los minutos pueden ser omitidos. El ejemplo de arriba ejecutara a las 5:00am en Febrero 1, 2007.

$ at 10 pm
Si el día y el año no están definido exactamente, **at** ejecuta los comandos en el primer día que iguala la especificación de time/date. En este ejemplo, el at-job se se ejecutara a las 10pm cualquier día. Así que, si este comando se usa a las 9pm, el at-job se ejecutara a las 10pm el mismo día (en una hora). Si se usa a las 11pm, por ejemplo, el at-job se ejecutara el próximo día a las 10pm (en 23 horas).

$ at 2:00 pm sat
Ejecuta at-job el próximo día que sea Saturday (Sábado) y el time sea 2pm. Si este comando ***at*** se usa un viernes, el at-job se ejecutaría a las 2pm el día siguiente. Si se usa a las 3pm el sábado, el at-job no se ejecutara hasta las 2pm el sábado de la próxima semana.

$ at midnight mar 1
Ejecuta el comando **at** a las 12am en Marzo 1^{ero} (1 de Marzo) de este mismo año si se usa antes de Enero-Febrero o el año que viene si se ejecuta después de Marzo hasta Diciembre.

$ at 9 am tomorrow
Ejecuta el at-job a las 9am mañana.

$ at midnight
Ejecuta el at-job a las 12am.

$ at now + 1 day
Ejecuta el at-job en exactamente 1 día.

$ at now next day
Equivalente al ejemplo anterior. La palabra clave ***next*** se usa en lugar de **+ 1**.(next --> próximo)

$ at noon + 5 minutes
Ejecuta el at-job 5 minutos después noon.

$ at 12:05 pm
Equivalente al ejemplo anterior.

- Imaginémosno dos comandos de prolongada ejecución, *comando1* y *comando2*, y además alto consumo de recursos computacionales. En vez de ejecutar estos comandos durante el día y consumir recursos necesarios para el funcionamiento de la empresa, preferimos ejecutarlos en la noche. El siguiente utiliza al comando **at** para programar que los dos comandos se ejecuten a las 10pm.

$ at 10 pm
at> **comando1**
at> **comando2**
at> **CTRL+d**
warning: cmds will be executed with /bin/sh
job 976385752.a at Sat Dec 8 01:00:00 2005

Algunas anotaciones de este ejemplo:

- Después de usar el comando **at** para especificar la fecha y hora de ejecución, digite los comandos el orden que desea que ellos se ejecuten, uno por línea. En este ejemplo programamos para que los comandos *comando1* sea ejecutado y seguido por el comando *comando2*. Se incluye el prompt (at>) antes de la entrada leída por el comando **at**. En su implementación de Linux/Unix puede ser que el prompt sea diferente o quizás no se presente ninguno.
- Cuando ha terminado de ingresar los comandos, escriba la secuencia de teclas de su sistema que indican EOF. Para la gran mayoría de sistemas *nix esta es la tecla **Control** y la tecla **d** simultáneamente (**CTRL+d**).
- Después de terminar dándole entrada a los comandos, ***at*** imprime dos mensajes. El primero es indicándole que shell utilizara para ejecutar sus comandos, en este caso es Bourne Shell (*sh*). El segundo nos da el JobID, queuename y horario programado de ejecución de at-job. En este caso de JobID es el 976385752. El nombre del ***queue queuename*** es ***a*** y el tiempo y fecha de ejecución es Sat Dec 8 01:00:00 2005 (o sea 1am el Sábado Diciembre 8, del 2005). El formato de esta salida varía dependiendo de la implementación de Unix que este usando; pero como enfatizamos una y otra vez debe ser muy similar y entendible por toda la información que arroje.
- Cualquier salida o output de los comandos comando1 o *comando2* le serán enviadas por email después de completada su ejecución.

- *El comando **at*** puede leer comandos almacenados en un archivo en vez de leerlos desde la entrada estándar. Por ejemplo, cree un archivo y nómbrelo *mi-at* en un editor de texto como el vi y agréguele las siguientes líneas.

  comando1
  comando2

  La sentencia de comando:

  $ at -f mi.at 10 pm

  Ejecutara los comandos listados en el archivo ***mi.at*** a las 10pm. Esto es equivalente al ejemplo anterior.

- Use la opción *-m* para recibir una notificación por email cuando el at-job haya concluido de ejecutar.

  $ at -m -f mi.at 10 pm

  Note que cualquier salida de los comandos *comando1* y *comando2 hubiese sido enviada por correo de todas formas*. La opción ***-m*** solo nos asegura de que en caso que los comandos no produzcan ninguna salida entonces recibamos un correo como quiera.

- Listar los JobIDs, queuename y programas de tiempo de ejecución de todos los at-jobs que tenemos programados.

  $ at -l
  976385710.a Sat Dec 8 01:00:00 2005
  976385403.a Sun Dec 9 17:54:00 2005

- Eliminar un at-job. Los comandos **at** nombrados por JobIDs no serán ejecutados y removidos del queue.

  $ at -r 976385710.a

Ejemplos Avanzados

Las salidas estándar y de error generadas por un at-job son enviadas por email al usuario al menos que sean redireccionadas. Por ejemplo, El comando Unix **who** imprime una lista de los usuarios actualmente ingresados al sistema (logged in).

$ at midnight
at> who
at> **Ctrl+d**

Le enviara un email con una lista de todos los usuarios que se encontraban ingresados en el sistema a medianoche (logged a la midnight). Mientras que aquí se redirecciona la salida estándar del comando **who,** almacenándola a un archivo de texto plano, llamado lista.txt, en vez de enviársela por correo electrónico o email.

$ at midnight
at> **who > lista.txt**
at> Ctrl+d

El archivo lista.txt se guardara en el directorio que se encontraba cuando programo el comando ***at***.

Tenga mucho cuidado al usar sintaxis dependiente del shell. El shell utilizado por los at-jobs puede que no sea su shell por defecto. Si usted no esta seguro, use ***at*** para programar un comando que no hace nada.

$ at now + 1 minute
at> **echo nada > /dev/null**
at> **CTRL+d**
warning: cmds will be executed with /bin/sh
job 976378752.a at Thu Dec 22 11:05:13 2005

Esto le listara el shell utilizado para interpretar los comandos. En este ejemplo, es el Bourne shell (sh).

En este ejemplo, redirecionamos la salida estándar y el error al mismo archivo. Aquí el comando *backup* se usa para efectuar un backup de un directorio. Por ejemplo,

$ at midnight
at> **backup ~**

Este comando **at** creara un backup de su directorio home a medianoche (midnight) esta misma noche. La sintaxis correcta para redireccionar las salidas estándar y de error a un mismo archivo es dependiente del shell. Si se encuentra en un c-shell o uno de sus derivados use:

$ at midnight
at> **backup /home >& blog.txt**

Si esta usando el Bourne shell o uno de sus dedicados use:

$ at midnight
at> **backup > blog.txt 2>&1**

Para escribir ambas salidas de salida estándar error estándar a un solo archivo de texto de nombre ***blog.txt***.

El shell usado por el comando ***at*** retiene el directorio actual de trabajo, y las variables de ambiente (excepto las variables del terminal y los ajustes de monitor) y el valor del **umask** en efecto en el momento de invocarlo. Experimentemos con el siguiente ejemplo para confirmar este echo.

```
$ at now + 1 minute
at> echo "Mi nombre o username es:"
at> whoami
at> echo "Mi Directorio de Trabajo Actual es:"
at> pwd
at> echo "Mi umask por defecto es:"
at> umask
at> echo "La ruta o path es:"
at> echo $PATH
```

Modificar Prioridades de Procesos

Conceptos

- No todas las tareas requieren el mismo monto de tiempo de ejecución
- GNU/Linux tiene el concepto de **prioridad de ejecución** para esta situación
- La prioridad de los Procesos son dinámicamente alterado por el kernel
- Puedes ver la prioridad actual de un proceso con **top** o **ps -l** y observar la columna **PRI**
- La prioridad pueden ser alteradas usando el comando **nice**
    - La asignación alterada por **nice** se ve en la columna **NI** del comando **top**

El comando nice

- Inicia un programa con una prioridad alterada
- Nombre extraño: procesos con 'nice' aplicado requieren menos recursos
- El rango de **nice** es desde **+19** (muy **nice**) a -20 (no muy **nice**)
- Usuarios que no son **root** solo pueden especificar rangos desde 1 al 19; el root tiene el rango completo
- El valor por defecto de **nice** es 10
- Para ejecutar un comando con su **nice** incrementada (prioridad reducida):
  $ nice -10 comando-de-ejecución-prolongada &
  $ nice -n 10 comando-de-ejecución-prolongada &

• Para ejecutar un comando con su **nice** reducida (prioridad más alta):
  $ nice --15 comando-importante &
  $ nice -n -15 comando-importante &

El Comando renice

- El comando **renice** cambia el nivel de nice de un proceso existente
- Los usuarios no son permitidos incrementar el nivel de **nice** de un proceso
- Para ajustar el proceso con **pid 2984** al máximo de **nice** de el (reducir su prioridad):
  $ renice 20 2984
- El **nice** es solo un número: así es que no extra símbolo **(-)**
- Para ajustar el proceso con **pid 3598** a un **nice** más bajo (con prioridad más alta):
  $ renice -15 3598
- También puedes cambiar el nivel de **nice** de todos los procesos de un usuario:
  $ renice 15 -u miguel

Práctica7

Ejercicio 1

1) Inicie un proceso ejecute **man find** y suspéndalo con **Ctrl+Z**.
2) Ejecute **xclock** en background, use **&**
3) Use **jobs** para listar los trabajos en segundo plano y los procesos detenidos.
4) Use el comando **fg** para traer a **man find** al primer plano; salga de el normalmente con **"q"**
5) Use **fg** para traer a **xclock** al foreground, y termínelo con **Ctrl+C**
6) Ejecute **xclock** nuevamente, pero esta vez sin el &. Debe estar ejecutándose en el foreground (no podrás utilizar el shell). Suspéndalo con Ctrl+Z y observe que sucede. Para apropiadamente ejecutarlo en el background, use bg.

Ejercicio 2

1) Use **top** para mostrar los procesos en ejecución en su computador.
2) Ejecute a **top** para que ordene por el uso de la memoria, para que los procesos que usan más memoria estén arriba.
3) Restrinja para que solo se muestren los procesos que usted es el dueño.
4) Intente matar (kill) uno de sus procesos (asegúrese de no ser nada importante).
5) Muestre una lista completa de los procesos ejecutándose en su maquina utilizando **ps**.
6) Muestre la misma lista pero en forma de árbol, usando ambos **ps** y **pstree**.
7) Logre que **ps** ordene la salida por el tiempo de uso del sistema.

Ejercicio 3

1) Cree el siguiente script shell, llámelo **siempre**, en su directorio home:

```
#!/bin/sh
while [ 1 ]; do
echo hola todos... >/dev/null;
done
```

Hazlo ejecutable y ejecútelo en el background así:

```
$ chmod a+rx siempre
$ ./siempre &
```

2) Use el comando **ps -l** para revisar el nivel de **nice** del archivo
3) Ejecuta el script con **nice** y asignarle un nivel de 15. Intenta ejecutarlo al lado de uno con un nivel de **nice** menor, y ver la diferencia en **top**
4) Usando a **nice** o **renice** trate que el nivel de nice sea menos de 0, o sea negativo-

Capítulo8

Aquellos que no conocen Unix, no son informaticos, y por ende están condenados a reinventarlo, y mal. *Antonio Perpiñan*

Conceptos de Sistemas de Archivos y el Manejo de los Permisos

Los Objetivos de este Capítulo son:

1. **Los diferentes tipos de archivos**
2. **Qué son los inodes**
3. **Qué son los vínculos Hard y Soft**
4. **Usuarios y Grupos**
5. **La cuenta del superusuario ROOT**
6. **Administrar Permisos**
7. **Gestionar permisos Especiales**

Conceptos de Sistemas de Archivos (FileSystem)

Sistemas de Archivos

- Existe mucha confusión en el uso del termino 'filesystem'
- Es comúnmente utilizado para expresar dos conceptos distintos

La jerarquía de archivos y directorios que humanos crean para organizar data en un sistema ('Sistema de archivos unificados')
El formato que el kernel usa para almacenar data en medios físicos, como son los discos ('tipos de sistemas de archivos')

Sistema de Archivos Unificado

- Sistemas **Unix** y **GNU/Linux** contienen un sistema de archivos unificado (**unified filesystem)**
  - Cualquier archivo, en cualquier disco o recurso de red compartido, puede ser accesado con un nombre que empiece con /
- El sistema de archivos unificado consiste de uno o más **sistemas de archivos individuales** ('ramificación' de la jerarquía unificada)
  - Cada sistema de archivos tiene su propia raíz (root)
  - La raíz puede ser anclada a cualquier directorio en el sistema unificado
  - El directorio en el cual un sistema de archivos individual es anclado en el sistema de archivos unificado es denominado en punto de montaje (**mount point)**
- Un sistema de archivos individual vive o existe en un dispositivo físico (por ejemplo un disco duro o disquete), aunque no este físicamente conectado a la misma computadora donde se monte

Tipos de Archivos

- Archivos contienen data directamente
- Los directorios proveen la jerarquía de los archivos: ellos pueden contener ambos archivos y directorios
- Archivos y directorios son ambos **tipos de archivos**
- Existen otros tipos de archivos, incluyendo **archivos especiales de dispositivos**:
    - **Device files** proveen un manera de poderle pedir al kernel que accese un dispositivo físico
    - La data que los **device file** aparentan contener es en realidad la secuencia de bytes o sectores en el dispositivo mismo.
    - **Device files** son por convención almacenados en el directorio ***/dev***

Inodes (Inodos) y Directorios

- Un **inode** es la estructura de data que describe un archivo en un sistema de archivos individual
- El contiene información del archivo, incluyendo su tipo (ya sea: archivo/directorio/dispositivo), tamaño, fecha de modificación, permisos, etc.
- Puedes visualizar que el inode es el archivo mismo
- Los inodes dentro del sistema de archivos están enumerados
    - El número del inode es denominado su '**inum**'
- Note que el nombre del archivo no se almacena en el inode si no en el directorio
    - Un directorio se almacena en disco como una lista de archivos y nombres de directorios
    - Cada nombre tiene un número de inode asociado a el
    - Separar los nombres de los inodes proporciona que puedes tener múltiples entradas de directorios refiriéndose a un mismo archivo.

Crear y Cambiar Vínculos Hard y Simbólicos

Vínculos Simbólicos (Links)

- Un vínculo simbólico (o **symlink**) es un quasi archivo que se comporta como nombre alternativo a otro archivo o directorio
- El 'contenido' del symlink es el archivo real al cual el apunta
- Cuando tratas de usar un nombre de archivo que incluye un symlink, el kernel reemplaza el symlink con el 'contenido' del original
- Symlinks te permiten mantener un archivo (o directorio) en un lugar, pero pretender que esta en otro
    - Por Ejemplo, para asegurarse que un nombre obsoleto funcione en software anteriores
    - O para distribuir data desde un sistema de archivos individual a particiones de discos

Examinando y Creando Enlaces Simbólicos

- ls -l te muestra donde se encuentran los enlaces simbólicos:
  $ ls -l /usr/tmp
  lrwxrwxrwx 1 root root 30 Sep 26 2000 /usr/tmp -> /var/tmp
- ls puede también mostrarte una lista de enlaces en diferentes colores dependiendo del archivo, o con el sufijo '@'
- Un enlace simbólico se crea con el comando ln -s
- Su sintaxis es similar a cp - el nombre original que viene primero seguido de el nombre que tu quieres crear:
  $ ln -s real-archivo archivo-link
  $ ln -s real-directorio directorio-link
  $ ls -l file-link directorio-link

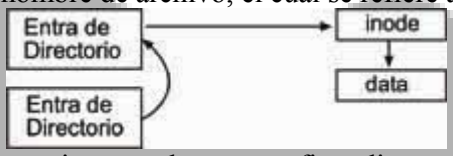
lrwxrwxrwx 1 miguel miguel 9 Jan 11 15:22 archivo-link -> real-archivo
lrwxrwxrwx 1 miguel miguel 8 Jan 11 15:22 directorio-link -> real-directorio

Enlaces Duros o Hard Links

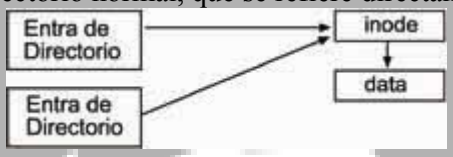
- ¿Donde esta referido un symlinks por el nombre de otro archivo?, un **hard link** esta referido a otro archivo por un número de inodo.
  - Un inodo es una estructura de datos que describe un archivo en el disco.
  - Este contiene la información acerca del archivo, su tipo (archivo/directorio/dispositivo), tiempo en el que fue modificado, permisos, etc.
- Un directorio contiene nombre y números de inodos.
  - Entonces los nombres de archivos no son considerados como parte interna del archivo.
- Tiene un hard link cuando diferentes entradas de un directorio entran en un sistema de archivos referidos a un mismo número de inodos.

Ilustrando un Symlinks y un Hard Links

- Un Symbolic Link se refiere a un nombre de archivo, el cual se refiere a un inodo:



- Un hard link es una entrada de directorio normal, que se refiere directamente a un inodo:



Comparando salidas de los hardlinks

Symlinks Hard links

- Los Symlinks son de manera distinta de los archivos normales, entonces nosotros podemos distinguir un symlink del punto hacia donde apunta.
- Los Symlinks pueden apuntar a cualquier tipo de archivo (dispositivos de directorio de archivos normales, symlink, etc.)
- Symlinks refiere al nombre, porque puede apuntar a otros sistemas de archivo.
- Los Symlinks pueden ocupar espacio adicional en el disco (para almacenar el nombre que ellos apuntan)
- Múltiples nombres tipo hard-link para el mismo archivo no tienen diferencia; el termino "hardlink" es puramente convencional. Hard links no puede apuntar a un directorio (o, un sistema NO GNU/Linux, a un symlink)
- Hard links trabaja por nombre de inode, pues ellos pueden trabajar un simple sistema de archivo.
- Al inverso, si renombras o borras el archivo original al cual el es apuntado por un symlink, el vinculo se rompe
- Renombrar o borrar el archivo 'original' apuntado por un hard link no tiene efecto sobre el hard link

- Los Hard links solo necesitan el espacio suficiente para la entrada en el directorio

Examinar y Crear Hard Links

- Use el comando ln para crear un hard link
- No use la opción -s al crear un hard links
- Al igual que cuando se crean symlinks, el orden de los argumentos de **ln** es parecido al de cp:

```
$ ls -l *.dtd
-rw-r--r-- 1 miguel miguel 11170 Dec 9 14:11 module.dtd
$ ln module.dtd capitulo.dtd
$ ls -l *.dtd
-rw-r--r-- 2 miguel miguel 11170 Dec 9 14:11 capitulo.dtd
-rw-r--r-- 2 miguel miguel 11170 Dec 9 14:11 module.dtd
```

- Note que el total de link en el listado se incremento a 2
- Los dos nombres ahora no se pueden distinguir
    - Borrar o renombrar uno de ellos no afecta el otro

Preservar Links

- Comandos que operan sobre archivos a menudo contienen opciones para especificar si se deben seguir los vínculos
- El comando tar se percata cuando dos archivos son hard links uno del otro, y los almacena correctamente.
- Por defecto tar también almacena los symlinks en los comprimidos
    - Use la opción -h (--dereference) para almacenar los archivos a que se le apunta
- El comando **cp p**or defecto ignora ambos hard links y symlinks
    - Use la opción -d (--no-dereference) para preservar todos los links
    - Use la opción -R (--recursive) para copiar recursivamente para asegurarse que los symlinks son preservados.
    - La opción **-a** (--archive) implica ambas **-d** y **-R**

Encontrar Symbolic Links a un archivo

- El comando **find** tiene una opción **-lname** la cual busca por symbolic links que contiene cierto texto:

```
$ find /etc -lname  "*kdm" -printf " %p -> %l\n"
```

- Este comando imprime los nombres y destinaciones de los symbolic links de los cuales los nombres de los archivos de destinos terminan en kdm
- Recuerde que estos comandos son intenso en el uso de recursos del disco duro!

Encontrar Hard Links a un Archivo

- Se pueden encontrar Hard links buscando en una entrada de directorio con el número del inode
- Primera, identidad del sistema de archivos y número de inode del archivo que estamos interesado:

```
# df Linux.pdf
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/hda2             2887140   2169880    570596  80%  /
# ls -i   Linux.pdf
341069 Linux.pdf
```

- Entonces use la opción **-inum** del comando para buscar por entradas en el directorio que los inodes sean igual
  $ find /home -xdev -inum 341069
- La opción **-xdev** previene que el comando **find** recursivamente a través del sistema de archivos

Administrar los Permisos

Usuarios y Grupos

- Cualquiera que usa GNU/Linux es un **usuario**
- El sistema mantiene un registro de todos los usuarios, por su nombre de usuario
  - Características de seguridad permite a diferentes usuarios tener diferente privilegios
- Los usuarios pueden pertenecer a **grupos**, así permitiendo que la seguridad sea administrada para un grupo de personas con diferentes requerimientos
- Use el comando **su** para cambiarse de usuario a usuario
  - Es más rápido que salir del sistema y entrar como el nuevo usuario de nuevo
- El comando **su** de pide el password del usuario:
  $ su - root
  Password:

La opción **- nombre_usuario** hace que **su** se comporte como si el usuario hubiese ingresado al sistema

El Superusuario: root

- Todo sistema GNU/Linux tiene un usuario llamado '**root**'
- El root es un usuario con todos los permisos del sistema
  - Puede accesar cualquier archivo
- La cuenta de root solo debe usarse para tareas administrativas, como instalar programas
- Cuando en la cuenta de root, el prompt del shell a menudo presenta un # al final
- Es mejor práctica utilizar su que trabajar como root:
  $ whoami
  miguel
  $ su -
  Password:
  # whoami
  root

Cambiar los Permisos de Propiedad con chown

- El comando **chown** cambia los derechos de propiedad de archivos y directorios
- Uso Simple:
  # chown miguel carta.txt
- Hace que ***carta.txt*** ahora sea adueñado por miguel
- Especifica cualquier número de archivos o directorios
- Solo el superusuario puede cambiar los permisos de dueños de un archivo
  - Esta características de seguridad - quotas, set-uid

Cambiar Grupos de Archivos con chgrp

- El comando **chgrp** cambia el apoderamiento de los **grupos** a los archivos o directorios

- Uso Simple:
  # chgrp estudiante reportes.txt
- Hace que estudiantes sea el grupo apoderado del archivo ***reportes.txt***
- Aunque con chown, puedes especificarle cualquier número de archivos o directorios
- El superusuario puede cambiar el apoderamiento de grupos a cualquier archivo a cualquier grupo
- El dueño de un archivo puede cambiar el apoderamiento de los grupos
  - Pero sólo a otro grupo del cual el es un miembro

Cambiar el Apoderamiento de un Directorio y su Contenido

- Una tarea común es cambiar el apoderamiento de un directorio y su contenido
- Ambos **chown** y **chgrp** aceptan la opción **-R**:
  # chgrp -R contabilidad directorio-compartido
- Mnemónicas: 'recursiva'
- Cambiar el apoderamiento del directorio-compartido a contabilidad
  - Y su contenido, y su subdirectorio, recursivamente
- Cambiar apoderamiento de los usuarios (solo el superusuario):
  # chown -R root /usr/local/share/misc/

Cambiar Apoderamiento de Usuarios y Grupos Simultáneamente

- El comando **chown** puede cambiar el apoderamiento del usuario y del grupo de archivo simultáneamente:
  # chown miguel:www /var/www/index.html
- Cambia el usuario dueño a **miguel** y el grupo dueño a **www**
- Se puede usar la opción **-R** normalmente
- Se puede usar un punto **(.)** en vez de dos puntos **(:)**
  # chown -R miguel.www /var/html/intranet/empleados/miguel/

Permisos y Control del Acceso a Archivos

Conceptos Básicos: Permisos en Archivos

- Tres tipos de permisos en archivos, cada uno denotado por una letra
- Un permiso representa una acción que se le puede hacer sobre el archivo:

Permisos	**Letra**	**Descripción**
Read	r	Permisos para leer la data almacenada en el archivo
Write	w	Permisos para escribir data al archivo, para truncar, o sobre escribir data
Execute	x	Permiso para intentar ejecutar el contenido del archivo como programa

- A menudo referido como los bits de los 'permisos'
- Note que para los scripts, usted necesita ambos permisos de **ejecutar** y **leer**
  - El intérprete del script (el cual ejecuta con sus permisos) necesita poder leer el script del archivo

Conceptos Básicos: Permisos en Directorios

- Los permisos r, w, x también tienen significado sobre los directorios

- El significado para los directorios es un poco diferente:
-

Permisión	Letra	Descripción
Read	r	Permisos para echar un vistazo en el directorio
Write	w	Permisos para crear, borrar, o renombrar archivos (o subdir) dentro del directorio
Execute	x	Permisos para cambiar o usar el directorio como parte intermediaria a un archivo

- La diferencia entre read y execute en un directorio es grande

- tener un permiso pero no otro casi siempre no es lo más deseado

Conceptos Básicos: Permisos para Diferente Grupos de Gente

- Además de tener diferente tipo de permisos, podemos aplicar diferente conjunto de permisos a diferente grupo de gente
- Un archivo (o directorio) tiene un **usuario dueño** y **grupo dueño**
- Los permisos r, w, x son especificados por separado por el, dueño, para el grupo dueño, y para todo los otros (el 'mundo')

Examinar Permisos: ls -l

- El comando **ls -l** te permite ver los permisos de un archivo:

```
$ ls -l
drwxr-x---  9  miguel  contabilidad  4096  Oct 12 12:57  cuentas
-rw-rw-r--  1  miguel  contabilidad  11170 Dec 9 14:11   reportes.txt
```

- La tercera y cuarta columnas son el dueño y el grupo dueño
- La primera columna son los permisos:
  - Un carácter para el tipo de archivo: d para directorios, - para archivos
  - Tres caracteres de permisos rwx del dueño (o un dash si los permisos no están disponible)
  - Tres caracteres de permisos rwx para los dueños del grupo
  - Tres caracteres de permisos rwx para todos los demás

Preservar Permisos para Copiar Archivos

- Por defecto, el comando **cp** hace intentos de preservar los permisos (y otros atributos, como timestamps)
- Puedes usar la opción **-p** para preservar los permisos y timestamps:

```
$ cp -p importante.txt importante.txt.orig
```

- Alternativamente, la opción **-a** preserva toda la información posible, incluyendo permisos y timestamps

Como se Aplican los Permisos

- Si eres el dueño de un archivo, los permisos de dueño le aplican
- De otra manera, si perteneces al grupo que es dueño, los permisos de grupo le aplican
- Si no es ninguno de estos dos casos, los permisos para los otros le aplican

Cambiar Permisos de Archivos y Directorios: chmod

- El comando **chmod** cambia los permisos de archivos o directorio
  - Los permisos de un archivo solo pueden ser modificado por el dueño y el superusuario
- El comando **chmod** toma argumentos que describen los nuevos permisos
  - Pueden ser especificado en varias maneras flexibles
- Ejemplo Simple:

```
$ chmod a+x programa
```

Suma **(+)** permisos de ejecución **(x)** para todos los usuarios (**a**) sobre el archivo ***programa***

Especificar Permisos con chmod

- Se puede colocar permisos utilizando letras del siguiente formato: **[ugoa][+=-][rwxX]**
- La primera letra indica a quien adjudicar los permisos:
  - La **u** es el dueño del archivo, **g** grupo dueño, la **o** es para todos los otros usuarios
- El símbolo **=** establece permisos para un archivo, el **+** suma permisos, el **-** remueve permisos
- Las ultimas letras indican cuales de los permisos se van a colocar **r, w, x**
  - O use la mayúscula X para colocar permisos de **x**, pero solo a directorio y archivos ya ejecutables

Cambiar los Permisos de un Directorio y su Contenido

- Un requerimiento común es cambiar los permisos de un directorio y su contenido
- El comando **chmod** acepta la opción **-R**:

  $ chmod -R g+rwX,o+rX directorio
- Mnemónica: 'recursive'
- Suma los permisos **rwx** sobre ***directorio*** para el grupo dueño, y le suma permisos **rx** a todos los otros
- Y a cualquier subdirectorio, recursivamente
- Cualquier archivo ejecutable contenido
- Archivos contenidos no ejecutable tienen permisos de **rw** agregados a ellos para el grupo dueño, y permisos de **r** leer para todo los otros

Permisos Especiales de Directorios: 'Sticky'

- El directorio ***/tmp*** debe ser escribible por todo el mundo, para que cualquiera pueda crear archivos temporales dentro de el
- Pero esto significaría que cualquiera pudiese borrar un archivo dentro de el - obviamente un tremendo agujero de seguridad
- Un directorio puede tener permisos 'sticky':
  - Solo el dueño del archivo lo puede borrar de directorio sticky
- Expresada con una **t** (mnemónicas: directorio temporario) en la lista:

  $ ls -l -d /tmp

  drwxrwxrwt 30 root root 11264 Dec 21 09:35 /tmp
- Habilite los permisos 'sticky' así:

  # chmod +t /data/tmp

Permisos Especiales de Directorios: Setgid

- Si un directorio esta **setgid** ('set group-id'), archivos creados dentro de el adquieren la apropiación del grupo dueño del directorio
  - Y los directorios creados dentro de el adquieren ambas la apropiación del grupo y el permiso setgid
- Muy útil para directorios compartidos donde todos los usuarios que le trabajan a los archivos son de un mismo grupo
- Expresada con una **s** en la posición del 'grupo' en el listado:

  $ ls -l -d /data/proyectos

  drwxrwxrwt-x 16 root admins 4096 Oct 19 13:14 /data/proyectos
- Habilite **setgid** así:

  # chmod g+s /data/proyectos

Permisos Especiales de Archivos: Setgid

- Permisos Setgid también pueden ser aplicados a archivos ejecutables
- Un proceso ejecutado desde un archivo con permisos **setgid** adquiere la identificación del grupo de ese archivo
- Nota: GNU/Linux no permite directamente que los scripts sean **setgid** - solo programas compilados
- Útil si deseas que un programa sea capaz de editar archivos que son propiedad de un grupo
    - Sin permitir que usuarios individuales accedan los archivos directamente.

Permisos Especiales de Archivos: Setuid

- Archivos pueden también tener permisos **setuid** ('set user-id')
- Equivalente a **setgid**: un proceso que se ejecuta desde un archivo con **setuid** adquiere la **id** del usuario del archivo
- Así como el **setgid**, GNU/Linux no permite que scripts sean **setuid**
- Expresada con una **s** en la posición de 'user' en el listado:

    $ ls -l /usr/bin/passwd
    -r-s--x--x 1 root root 12244 Feb 7 2000 /usr/bin/passwd
- Habilitar los permisos **setuid**:
    # chmod u+s /usr/local/bin/programa

Desplegar Permisos no Usual

- Use el comando **ls -l** para mostrar los permisos de los archivos
    - Permisos de **Setuid** y **Setgid** se muestran con una **s** en la posición de ejecutar del grupo y el usuario
    - El sticky bit se muestra con una **t** en la posición de ejecutar de los 'other'
- La letras **s** y **t** cubren el bit de ejecución
    - Pero puedes ver si el bit de ejecución esta encendido
    - La **s** o **t** minúsculas indican que el bit de ejecución esta disponible (si hay una **x** detrás de las letras)
    - La **S** o **T** mayúsculas indican que el bit de ejecución no esta disponible (existe un **-** detrás de la letra)

Permisos como Números

- A veces encontrarás números que se refieren a un conjunto de permisos
- Calcular los números sumando uno o más de los siguientes juntos:

4000	**Setuid**	**40**	Legible por el dueño del grupo
2000	**Setgid**	**20**	Escribible por el dueño del grupo
1000	**'Sticky'**	**10**	Ejecutable por el dueño del grupo
400	**Readable by owner**	**4**	Legible por cualquiera
200	**Writable by owner**	**2**	Escribible por cualquiera
100	**Executable by owner**	**1**	Ejecutable por cualquiera

• Puedes utilizar permisos con valor numérico con el comando chmod:
    $ chmod 664 *.txt
es equivalente a:
    $ chmod ug=rw,o=r *.txt

Permisos por Defecto: umask

- El comando **umask** permite que afectemos los permisos por defecto en los archivos y directorio que podremos crear:
  $ umask 002
- El argumento se calcula sumando junto el valor de los permisos **rwx** que no deseas aplicados a los nuevos archivos y directorios cuando se creen
  - Este Ejemplo tiene solo 2 - evitar escribible por todos, pero enciendo todo lo otro
- Otros valores de umask comunes son:
- 022 - evitar que el mundo - y grupo-escribible, permite todo los otros
- 027 - evitar grupo-escribible, y permitir no permisos para los otros
- Normalmente deseas ponerle una llamada al umask en sus archivos de configuración

Práctica8

Ejercicio 1

1) Cree un directorio temporal y cámbiese a el.
2) Cree varios archivos de la siguiente manera:

   $ echo "naranjas y limones" > frutas

   $ echo aguacate > vegetal

3) Cree un symbolic link llamado banana al archivo vegetal.
4) Cree un hard link llamado cítricos al archivo apropiado, y revise que tenga el mismo número de inode.
5) Borre el archivo original frutas y revise que citrus aun contenga el texto.
6) Borre el archivo original vegetal y trate de ver el contenido de almidón. Use **ls** para revisar el symlink.

Ejercicio 2

1) Trate de ver que sucede en el siguiente loop, y entonces cree algunos archivos .htm y pruébelo:

   ```
   $ for htm in *.htm; do
   ln -s $htm ${htm}l;
   done
   ```

2) Cree un symlink a un directorio llamado dir (por ejemplo a /etc).
3) Pruebe el siguiente comando para desplegar el vínculo y compare con el resultado:

   ```
   $ ls -l dir
   $ ls -l dir/
   ```

Ejercicio 3

1) Investigue quien es el dueño del archivo */bin/ls* y quien es el dueño de su directorio home (en */home*).
2) Ingrese como root, y cree un archivo vació con touch. El usuario y grupo apropiado debe ser 'root' - revise con ls.

3) Cambie el dueño del archivo que sea ‘users’.

4) Cambie el grupo dueño que sea cualquier usuario que no sea el root.

5) Cambie a ambos dueños de nuevo a que sea ‘root’ con un solo comando.

Ejercicio 4

1) Busca los permisos en su directorio home (como un usuario normal). Pueden otros usuarios accesar archivos ahí dentro?

2) Si su directorio home no esta accesible para usted, entonces cambiemos los permisos para permitir que otros que puedan leer archivos dentro de el, si no cámbielo para que puedan.

3) Revise los permisos en */bin* y */bin/ls* y asegúrese que sean razonable.

4) Revise los permisos disponibles en */etc/passwd* y */etc/shadow*.

5) Escriba un comando el cual le permitiera a los usuarios poder navegar en sus directorios home y cualquier subdirectorio dentro de el y que puedan leer los archivos.

Capítulo9

La diferencia básica entre un hacker y un cracker: ``el primero crea, el segundo destruye.''
Eric S. Raymond

Conceptos de Crear, Montar, Mantener y Administrar Particiones y Sistemas de Archivos

Los Objetivos de este Capítulo son:

1. **Discos y Particiones**
2. **Crear Sistemas de Archivos**
3. **Montar y Desmontar Sistemas de Archivos**
4. **Monitorear los Discos y Particiones**
5. **Encontrar Archivos del Sistema**
6. **El FHS**
7. **Gestionar permisos Especiales**

Crear Particiones y Sistemas de Archivos

Conceptos: Discos y Particiones

- Un disco duro provee un espacio amplio de almacenaje
- Usualmente divido en particiones
- Información de las particiones se almacenan en la tabla de partición
- GNU/Linux por defecto usa tablas de particiones compatibles con Microsoft Windows
- Para ser compatible con Windows, no más de cuatros particiones primarias pueden ser creadas
- Pero si pueden ser Particiones Extendidas, las cuales pueden ser subdivididas en particiones lógicas
- Particiones Extendidas tienen sus propias tablas de partición donde almacenan su información de sus particiones lógicas

Nombre de los Discos

- Los archivos para los discos IDE son desde el /dev/hda al /dev/hdd
- Los hda y hdb son los dispositivos en el primer canal, hdc y hdd son los del segundo canal
- El primer dispositivo en cada canal es el IDE 'master', y el segundo es el IDE 'slave'
- Las particiones primarias están enumeradas desde el 1-4
- Particiones lógicas son enumeradas desde el 5 en adelante
- Los dispositivos /dev/hda, etc., se refieren a discos duro enteros, no a particiones
- Súmele el número de la partición para referirse a una en particular
- Por Ejemplo, /dev/hda1 es la primera partición en el primer disco IDE en el primer canal
- Discos SCSI se nombran /dev/sda, /dev/sdb, etc

Usar el fdisk

- El comando fdisk es utilizado para crear, borrar y cambiar particiones en un disco
- Pásale a fdisk el nombre del disco a editar como argumento, por Ejemplo:
- **# fdisk /dev/hda**
- El **fdisk** lee comandos de una letra desde el usuario
- Digite **m** para un listado de los comandos
- Use **p** para mostrar las particiones que existen actualmente
- Use **q** para salir sin alterar nada
- Use **w** para escribir sus cambios
- Trabaje con mucho cuidado, los cambios de fdisk no son reversibles!

Usar el cfdisk

- El cfdisk es un programa basado en **curses**, para particionar un disco duro. El dispositivo puede ser cualquiera de los siguientes: **/dev/hda** [por defecto] **/dev/hdb /dev/sda /dev/sdb /dev/sdc /dev/sdd**
- Si sabes usar el fdisk, es más fácil ya que tiene un druid muy intuitivo basado en curses

Crear nueva Particiones

- Cree una partición nueva con el comando **n**
- Elija si va a ser una partición primaria, extendida o lógica
- Elija el número que se le va a asignar
- El **fdisk** le preguntara donde poner el principio y el final de la partición
- El tamaño por defecto crea una partición con todo el espacio disponible
- El tamaño deseado puede ser especificado en mega-bytes, Ej., +250M
- Cambios a la tabla de particiones solo se escriben cuando ejecutas el comando **w**

Cambiar Tipos de Particiones

- Cada partición tiene un tipo asociado con ella, cual es representado por un número
- El comando l de fdisk lista todos los tipos conocidos
- El comando t cambia el tipo de una partición ya existente
- Digite el tipo en el prompt
- Las particiones GNU/Linux son usualmente de tipo ‘Linux native’ (tipo 83)
- Otros sistemas operativos puede ser que usen otro tipo de particiones, muchas de ellas son reconocidas por GNU/Linux

Crear Sistema de Archivos con mkfs

- El comando **mkfs** inicializa un sistema de archivos en una partición nueva
- Advertencia: toda data en la partición se perderá
- Por Ejemplo, para crear un sistema de archivos **ext2** en la partición /dev/hda2:
- **# mkfs -t ext2 -c /dev/hda2**
- La **-t** fija el tipo de sistema de archivos a crear, y la **-c** revisa si el disco tiene bloques defectuosos
- El comando **mkfs** utiliza otros programas para hacer tipos específicos de sistemas de archivos, como lo son **mke2fs** y **mkdosfs**

Montar y Desmontar Sistemas de Archivos

Montar Sistema de Archivos

- Desde el punto de vista de muchas de las partes del sistema GNU/Linux, una partición contiene enteramente data arbitraria
- Al instalar, usted prepara para que una partición contenga el sistema de archivos- una forma de organizar data en archivos y directorios
- Uno de los sistemas de archivos esta compuesto del **root filesystem**: el directorio root en ese sistema de archivos se convierte en el directorio nombrado la */*
- Otros sistemas de archivos pueden ser montados: el directorio root del sistema de archivos es injertado en in directorio del sistema de archivos root
- Esto gestiona para que cada archivo en cada sistema de archivo montado sea accesible desde un singular punto lógico unificado (unified name space)
- El directorio que se le injertado se llama el punto de montaje (**mount point)**

Montar un Sistema de Archivos: mount

- Sistemas de Archivos Importantes se montan durante el arranque del sistema; otros filesystems pueden ser montados o desmontados en cualquier momento
- El comando **mount** monta sistemas de archivos
- Necesitas privilegios de **root** para montar sistemas de archivos
- El comando **mount** facilita montar y desmontar sistemas de archivos preconfigurados por el administrador del sistema
- Por Ejemplo, muchos sistemas vienen configurados para montar

  $ mount /mnt/cdrom

Montará el contenido del CD-ROM en el directorio ***/mnt/cdrom***

Montar Otros Sistemas de Archivos

- La sentencia mount /dev/sdb3 /mnt/extra monta el sistema de archivos almacenado en el dispositivo /dev/sdb3 en el punto de montaje /mnt/extra

  # mount -t vfat /dev/hdd1 /mnt/windows
- Los sistemas de archivos permitidos son listado en la página man mount (8)
- Para ver un listado de los sistemas de archivos actualmente montados, ejecute **mount** sin opciones

  # mount

Desmontar un Sistema de Archivos: umount

- Un sistema de archivos puede ser desmontado con el comando **umount**
- Fíjese bien como se escribe **umount NO unmount**!
- Para desmontar lo que este montado en el punto de montaje ***/mnt/extra***

  # umount /mnt/extra
- Para desmontar el dispositivo **/dev/sdb5** y cualquier sistemas de archivos en el, donde este montado

  # umount /dev/sdb5
- Normalmente necesitas tener privilegios de root para desmontar sistema de archivos
- No es posible desmontar un sistema de archivos que este 'ocupado' o sea en uso
- Un sistema de archivos esta ocupado si un proceso contiene un archivo abierto
- O si un proceso tiene un directorio dentro de su actual directorio

Configurar mount: /etc/fstab

• El archivo ***/etc/fstab*** contiene información acerca de los sistemas de archivos que son reconocidos por el administrador del sistema.

- Al especificar un sistema de archivos en ***/etc/fstab*** se hace posible usar como único argumento solo el punto de montaje
- En el archivo ***/etc/fstab*** también puedes configurar cuales sistemas de archivos montar durante el inicio
- Cada línea en el archivo ***/etc/fstab*** describe un sistema de archivos
- Seis columnas en cada línea

Ejemplo de un archivo /etc/fstab

- Un ejemplo de un archivo ***/etc/fstab***:

```
# device        mount-point     type            options (dump)          pass-no
/dev/hda3       /               ext2            defaults                1   1
/dev/hda1       /boot           ext2            defaults                1   2
/dev/hda5       /usr            ext2            defaults                1   2
/dev/hdb1       /usr/local      ext2            defaults                1   2
/dev/hdb2       /home           ext2            defaults                1   2
 none   /proc            proc            defaults                0   0
/dev/scd0       /mnt/cdrom      iso9660         noauto,users,ro 0   0
/dev/fd0        /mnt/floppy     auto            noauto,users            0   0
```

Tipos de Sistemas de Archivos

- Los sistemas de archivos más comunes son:

Tipo de Uso	
ext2, 3	El sistema de archivos estándar de GNU/Linux
iso9660	El sistema de archivos utilizado en CD-ROMs
proc	No es un sistema de archivos real, así es que usa a none como su device. Utilizado para que el kernel pueda reportar información del sistema a los procesos del usuario
vfat	El sistema de archivos utilizado por Windows 95
auto	No es un sistema de archivos real. Se usa para que el comando mount pruebe para los tipos de sistemas de archivos, particularmente para medios removible

- Sistemas de archivos de Redes (Networked filesystems) incluye **NFS** (Especifico a Unix) y el **smbfs** (Windows o Samba)
- Existen otros, menos común; véase: **man 8 mount**

Opciones de Mount

- Opciones separadas por comas en el archivo ***/etc/fstab***
- Alternativamente, use opciones separadas por comas con la **-o** en la línea de comandos
- Las opciones comunes de mount son:

Descripción de Opción	
noauto	en /etc/fstab, previene que un sistema de archivos se monte al inicio. Útil para media removible
ro	Monta un sistema de archivos en modo solo lectura (read-only)
users	Permite que usuarios sin privilegios de root monten y desmonten sistema de archivos
user	Como users, pero usuarios solo pueden desmontar sistemas de archivos que ellos montaron

- Existen muchas otras opciones véase la pagina man de mount (8)

Otras columnas en /etc/fstab

- La quinta columna se llama **dump**
- Usada por **dump** y **restore** de las utilidades de backup
- Muy pocas personas utilizan esas herramientas
- Solo use 1 para sistemas de archivos normales, y 0 para sistemas de archivos removible
- La sexta columna se llama **pass-no**
    - Controla el orden en la cual filesystems montados automáticamente son revisados por fsck
    - Use 1 para los sistemas de archivos root
    - Use 0 para los sistemas de archivos que no se montan al inicio
    - Use 2 para los otros sistemas de archivos

Montar un Archivo

- Usar **loop devices**, GNU/Linux puede montar un sistema de archivos almacenado en un archivo normal, en ves de en un disco
- Útil para probar imágenes de CD-ROMs antes de quemar el disco
- Por Ejemplo, para crear un sistema de archivos aproximadamente del tamaño de un floppy:
  # dd if=/dev/zero of=disk.img bs=1024 count=1400
  # mke2fs -F disk.img
- Para montar el archivo para que su contenido este disponible en ***/mnt/disk***:
  # mount -o loop disk.img /mnt/disk

Mantener la Integridad del Sistema de Archivos

Conceptos de Sistemas deArchivos

- Los archivos almacenados en una partición de un disco son organizados en un **sistema de archivos**
- Existen varios tipos de sistemas de archivos; los comunes GNU/Linux son llamados **ext2/ext3**
- Un sistema de archivos contiene un número fijo de **inodes**
- Un inode es la estructura de data que describe un archivo en un disco
- Contiene información del archivo, incluyes su tipo (archivo/directorio/dispositivo), fechas de modificación, permisos, etc.
- El nombre de un archivo se refiere a un inode, no directamente al archivo
- Esto permite **hard links**: muchos nombres de archivos refiriéndose a un mismo inode

Problemas Potenciales

- Al pasar el tiempo, sistemas de archivos pueden desarrollar problemas:
- Se puede llenar, causar que programas individuales o sistemas enteros fracasen
- Se pueden corrumpir, tal vez por una falla eléctrica o si el sistema se cuelgue.
- Se puede acabar el espacio para los inodes, y no se podrá crear ni archivos y directorios nuevos
- Monitoreo y revisión del sistema de archivos regularmente puede ayudar a prevenir y corregir problemas de este tipo.

Monitorear el Espacio en Discos: df

- Ejecute **df** sin argumentos para listar el espacio libre en los sistemas de archivos montados

- Muy útil usar la opción **-h**, la cual despliega en unidades más legible el espacio libre:

$ df -h

FileSystem	Size	Used	Avail	Use%	Mounted on
/dev/hda8	248M	52M	183M	22%	/
/dev/hda1	15M	5.6M	9.1M	38%	/boot
/dev/hda6	13G	5.0G	7.4G	41%	/home
/dev/hda5	13G	4.6G	7.8G	37%	/usr
/dev/hda7	248M	125M	110M	53%	/var

- La columna **Use%** muestra el porcentaje del sistema de archivos en uso
- Puedes pasarle a **df** directorios como argumentos para hacerlo mostrar el espacio de los directorios en los sistemas de archivos que esos directorios están montados

Monitorear los Inodes: df

- Sistemas de archivos muy raramente que se le agoten sus inodes, pero puede sucede si el sistema de archivos contiene muchos archivos pequeños
- Ejecute **df -i** para revisar la información sobre el uso de los inodes en los sistemas de archivos montados:

$ df -i

FileSystem	Inodes	IUsed	IFree	IUse%	Mounted on
/dev/hda8	65736	8411	57325	13%	/
/dev/hda1	4160	30	4130	1%	/boot
/dev/hda6	1733312	169727	1563585	10%	/home
/dev/hda5	1733312	138626	1594686	8%	/usr
/dev/hda7	65736	1324	64412	2%	/var

- En este Ejemplo, cada sistema de archivos ha usado un porcentaje más pequeño de sus inodes (IUse%) que su espacio de archivos
- Es muy buena indicación!

Monitorear Uso del Disco: du

- El comando **df** muestra un resumen del espacio libre en un a partición
- El comando **du** muestra la información del espacio en disco utilizado en un árbol de directorio
- •Toma uno o más directorios como argumentos en la línea de comandos:

$ du /usr/share/vim
2156 /usr/share/vim/vim58/doc
2460 /usr/share/vim/vim58/syntax
36 /usr/share/vim/vim58/tutor
16 /usr/share/vim/vim58/macros/hanoi
16 /usr/share/vim/vim58/macros/life
40 /usr/share/vim/vim58/macros/maze
20 /usr/share/vim/vim58/macros/urm
156 /usr/share/vim/vim58/macros
100 /usr/share/vim/vim58/tools
5036 /usr/share/vim/vim58
5040 /usr/share/vim

Opciones de du

Descripción de las Opciones

-a Muestra todos los archivos, no solo los directorios
-c Imprime un total acumulativo para todos los directorios nombrados en la línea de comandos
-h Imprime uso del disco en unidades más legible para los humanos
-s Imprime un resumen para cada directorio nombrado en la línea de comandos
-S Hace que el tamaño reportado para un directorio sea el tamaño de los archivos en ese directorio, no el total incluyendo el tamaño de los subdirectores

Encontrar y Reparar Sistemas de Archivos Corrompidos: fsck

- Algunas veces sistemas de archivos se corrompen
- Tal vez hubo una falla eléctrica
- O la versión de su kernel tiene algún bug
- El programa **fsck** revisa la integridad del sistema
- Y puede lograr los arreglos necesarios
- Actualmente tiene dos partes principales:
- Un 'drive program', **fsck**, el cual maneja cualquier sistema de archivos
- Un 'backend program' para cada tipo de sistema de archivos
- El 'backend program' para ext2 es e2fsck, pero siempre es invocado desde el fsck

Ejecutar fsck

- El comando fsck es normalmente ejecutado durante el inicio del sistema
- Así es que se ejecuta automáticamente si el sistema se apagó (shut down) incorrectamente
- También puede ser ejecutado manualmente:
  # fsck /dev/sdb3
- Interactivamente pregunta si deseas reparar los problemas a medida los encuentra
- Use la opción **-f** para forzar la revisión del sistema de archivos, aunque **fsck** crea que fue desmontado limpiamente
- Use la opción **-y** para automáticamente responder 'yes' a todas las preguntas
- No es muy buena idea ejecutar **fsck** sobre un sistema de archivos montado!

Encontrar y Colocar Archivos en su Lugar

Organización de un Sistema de Archivos Unix

- Muchos de los nombres de archivos son abreviaturas de palabras reales
- Estructura tradicional la cual se ha desarrollado durante muchos años
- La mayoría de los archivos del sistema ocupan su lugar apropiado
- Los programas dependen de que estén en su correcto lugar
- Usuarios familiarizados con la estructura de Unix pueden perfectamente bien interactuar con cualquier sistema Unix o GNU/Linux
- Pero los directorios home de los usuarios pueden ser estructurados a su manera

El Estándar del Sistema de Archivos Jerárquico

- Se inicio como un intento de estandarizar el sistema de archivos GNU/Linux
- Llamado el **FSSTND** en su primera versión la cual fue publicada en 1994
- Ampliamente aceptada por los distributores
- Pero solo unos cuantos sistemas GNU/Linux están dentro de estos estándares 100%
- La intención es para evitar la fragmentación de las distribuciones GNU/Linux

- renombrada "**File Hierarchy Standard"**, o "**FHS"**
- Ahora la intención es que aplique a todos los sistemas operativos derivados de Unix (Unix-Like)

Data compartible y no-compartible

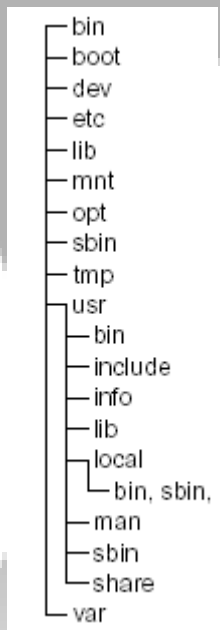
- Algunos archivos pueden ser compartidos entre múltiples computadores, utilizando sistemas de archivos de redes "networked filesystems" como lo es el **NFS**
- Esto puede economizar espacio en disco, aunque ya esto no es tan importante hoy día
- Más importante es que puede ayudar a centralizar la administración de una red
- A menudo programas, email y directorios home son compartidos vía redes
- Archivos de diarios (Log) y de configuración especifica de maquina no son compartibles

Data Estática y Dinámica

- Algunos archivos raramente cambian, mientras que otros cambian siempre
- Es aconsejable almacenar archivos estáticos separados de esos que cambian regularmente:
- los archivos estáticos pueden estar en una partición montada read-only (por ejemplo un CD-ROM)
- Programas y librerías son usualmente estática (excepto cuando se instala el nuevo software)
- Los directorios home y archivos son usualmente más variable

Vistazo al FHS

/ contiene los archivos esenciales necesitados para iniciar el sistema
/usr contiene otro paquetes de software
/usr/local contiene software no empaquetada



FHS: Software Instalado

- Los programas se encuentran a menudo en los directorios ***bin*** y ***sbin***
- Estos se encuentran en */*, ***/usr*** y ***/usr/local***
- Se usa ***sbin*** para almacenar programas de uso del administrador del sistema y no los usuarios (mail daemon,

web server, etc.)
- Estos directorios se nombran por los **binarios**
- La mayoría de los programas en ellos son binarios (programas compilados), aunque algunos son legible scripts en formato de texto
- Las librerías son almacenadas en directorios llamados ***lib***, encontrados en los mismos sitios que ***bin***
- Los directorios listados en ***/etc/ld.so.conf***

FHS: Otros Directorios debajo de /usr

- ***/usr/include-*** contiene archivos cabezales usados por programas de **C/C++**
- ***/usr/X11R6-***contiene archivos usados por el sistema X Window, incluyendo programas, librerías, archivos de configuración y documentación
- ***/usr/local***- Donde software se instala cuando se compila desde código fuente y no se instala un paquete
- ***/usr/share***- Contiene archivos que no son específicos a arquitectura de maquina, Ej., fuentes y iconos
- Teóricamente puede compartir entre diferente tipos de maquinas sobre una red.
- ***/usr/src*** siempre contiene el código fuente del Kernel Linux
- Usualmente se mantiene en un directorio, por ejemplo: ***Linux-2.4.20***, con un vinculo simbólico a ***Linux***

FHS: Directorios Debajo de /var

- ***/var/run-*** contiene los pid de los archivos (archivos con los id de los procesos para programas tipo daemons que se encuentran en ejecución)
- También contiene ***utmp***, un record de los usuarios en sesión
- ***/var/mail*** o ***/var/spool/mail***- es donde el queue de correo de cada usuario se mantiene hasta ser eliminado o salvado
- ***/var/log-*** contiene los logs producido por varios programas, incluyendo **syslog**
- ***/var/cache-*** contiene data generada por programas la cual es cached para salvar tiempo
- Data **Cached** puede ser regenerada si es eliminada

FHS: Otros Directorios

- ***/etc-*** contiene archivos de configuración
- ***/mnt-*** se usa para montar sistemas de archivos externos temporalmente
- Por Ejemplo, los disquetes floppy se montan en ***/mnt/floppy*** (aunque en Debian es en /floppy)
- ***/boot-*** contiene los archivos utilizados por LILO para iniciar el sistema (también GRUB)
- ***/dev-*** contiene archivos de dispositivos, los cuales proveen acceso al hardware como los son disk drives o puertos seriales
- ***/tmp-*** es usado por muchos programas para almacenar sus archivos temporales
- ***/opt-*** Puede contener paquetes de software no nativos o denominado de terceros (Ej., OpenOffice, Forte)

FHS: Otros Directorios

- ***/proc-*** provee acceso a información desde el kernel, particularmente acerca de los procesos en ejecución
- ***/home-*** contiene los directorios que le pertenecen a cada usuario
- Use echo ~ para saber donde esta su directorio home
- ***/root-*** es el directorio home del usuario root

Encontrar Programas con which

- Busca programas que pueden ser ejecutados
- Busca en los mismos directorios que el Shell
- Determinado por la variable de entorno el **$PATH**
- Use **echo $PATH** para ver que directorios son buscados
- Por Ejemplo, para saber donde esta el comando aumix:

$ which aumix

- Es muy útil tener diferentes versiones de un mismo programa instalados en diferentes lugares

El comando Built-in type

- El comando **type** es parecido al comando **which**, pero es parte del shell, mejor dicho un built-in.
- Nos informa de los aliases del shell y funciones
- No esta disponible para el **C** Shell
- El comando **type -p** es lo mismo que el comando **which**
- El comando **type -a** nos muestra todos los comandos de el nombre que damos
- Útil para detectar programas duplicados, o aliases que nos esconden programas reales
- Véase las paginas man para más detalles

Revisando los Comandos Propios del Shell con type

- Algunos comandos son parte del Shell, denominados "built-in" del shell
- Ejemplos incluyen a **cd**, **test**, **pwd** y **ulimit**
- El shell de ***Bash*** tiene un built-in llamado **type** el cual reporta si un comando es un built-in
- Por Ejemplo, para ver si el comando **test** ejecutara un shell built-in, o un programa real:

$ type test

- El Ejemplo nos muestra que **test** ejecutara un **shell built-in**, aunque existe un programa real con el mismo nombre
- El comando **type** también identificara los aliases del shell y funciones

El comando uname

El comando uname lista la información referente al sistema operativo. Use el comando *uname* para visualizar que saber de *nix o GNU/Linux y la versión que usted esta usando y sobre que tipo de hardware esta usted ejecutándola. Usuarios Avanzados, aprenden a usar el comando *uname* en sus scripts del shell.

Descripción

uname [*options*]

uname [*opciones*]

El comando u*name* escribe información acerca del sistema operativo a la salida estándar. Por ejemplo,

$ uname
Linux

Nos dice que estamos usando Linux, que podría ser FreeBSD, Solaris, etc.

Las opciones de la línea de comandos para el comando *uname* se describen en esta tabla que sigue.

Opción	Descripción
-s	Nombre del sistema (e.j. Linux, BSD, FreeBSD, HP-AUX, ...etc)
-n	Nombre del host o node dentro del network
-r	Nivel de lanzamiento del Sistema
-v	Nivel de la Versión de este lanzamiento del Sistema Operativo
-m	Nombre del hardware (tipo de hardware que ejecutamos el sistema)
-a	Todo (se comporta como si fuese especificado -snrvm)

Si no especificamos ningunas opciones, *uname* escribe el nombre del sistema, como si hubiésemos escrito *uname -s*.

Cuando requerimos más de una información del sistema, el comando *uname nos muestra la salida en el siguiente orden con cada pedazo de información separada por espacio en blanco*.
<*Nombre del sistema*> <nombre del equipo> <Lanzamiento> <*versión*> <nombre del *hardware*>

Nota: Si usted usa Solaris, *uname* reporta la información del hardware un poco diferente que los otros sabores de *nix. En ves de solo la opción *-m*, existen tres opciones de la línea de comandos que reportan información del hardware.

Opción	Descripción
-m	Reporta la arquitectura del kernel (Clases de maquinas con la misma arquitectura que pueden ser iniciadas con el kernel de este sistema operativo.)
-p	El tipo de procesador (e.j. Sparcs de maquinas Sun, i836 de Sistemas PC basadas en plataforma Intel, etc.)
-i	Nombre de la plataforma del hardware (Por ejemplo, SPARCstation 5 y no con la -p que solo reporta **sparc**.)

Ejemplos

1. Muestra el nombre del sistema. Típicamente este será el sabor de *nix o GNU/Linux que este ejecutando.

   $ uname
   FreeBSD

   Esto es idéntico a usar *uname* con la opción *-s*.

   $ uname -s
   Linux

2. Muestre el nombre del nodo del sistema, o el host más su domain.

   $ uname -n
   desktop1.dominio.net

   El nodo del sistema es el nombre del computador en el network. Este nombre variara dependiendo del tipo de network en que que el equipo esta integrado. En el ejemplo anterior, el nombre del node es el nombre completo calificado del dominio o el FQDN. En otros casos, puede ser solo el nombre del computador o el hostname. Por ejemplo,

   $ uname -n
   contabilidad-07

   Nota: Host es el nombre único por el cual su computador es conocido en la red o network.
3. Muestre el nivel de lanzamiento del sistema.

   $ uname -r
   2.4-RELEASE

4. Muestre el nivel de la versión del lanzamiento de su sistema.

   $ uname -v
   miguel@linux:~/TEMP$ uname -v
   #1 SMP Fr Sep 19 17:55:45 CEST 2003

En Linux igual que en FreeBSD, *uname* da una descripción bien extensa para la versión. En otros sistemas, la repuesta es más corta. En este ejemplo *uname -v* en un sistema Solaris.

$ uname -v
Generic_103093-06

5. Muestra el nombre del hardware.

   $ uname -m
   i686

   La salida de i686 indica que es un sistema PC basado en Intel.

6. Muestre toda la información del sistema.

   $ uname -a
   Linux linux 2.4.22-xfs #1 SMP Fr Sep 19 17:55:45 CEST 2003 i686 GNU/Linux

   Recordemos que la información es presentada en este orden <nombre del sistema> <nombre del nodo> <lanzamiento> <*versión*> <*hardware*> separadas por espacios.

   Otro ejemplo de salida de *uname -a*; esta vez en un sistema Solaris.

   $ uname -a
   SunOS workstation1 5.5 Generic_103093-06 sun4m sparc SUNW,SPARCstation-5

7. Muestre el nombre del sistema y el Lanzamiento.

   $ uname -sr
   Linux 2.4.22-xfs

   Cuando requerimos más de una sola información del sistema, la salida es separada por uno o más espacios en blanco. Note el orden de la salida, no es relacionado con las opciones que especificamos. Por ejemplo,

   $ uname -rs
   Linux 2.4.22-xfs

Requisición de información de lanzamiento (release (r)) información antes del nombre del sistema (s) pero la salida aun imprime en el mismo orden de <*system name*> <*release*>.

Ejemplos Avanzados

1. El comando *uname* es comúnmente usado dentro de scripts del shell para agregar código que sea independiente del sistema en que se va a ejecutar. Por ejemplo,

```
#!bin/sh
case $(name ) in
Linux)
#Código específico a Linux
echo "Usted esta ejecutando GNU/Linux."
   ;;
SunOS)
#Codigo específico a SunOS/Solaris
echo "Usted esta Usando SunOS o Solaris."
;;
AIX)
#Codigo específico a AIX
echo "Usted esta usando AIX."
;;
FreeBSD)
#Codigo específico a FreeBSD
echo "Usted esta usando FreeBSD."
;;
*)
#Codigo específico a Desconocido
echo "Usted esta usando un Sistema Operativo desconocido."
;;
exit 1
;;
esac
exit 0
```

Nota: Cuando usa este tipo de scripts que dependen en repuestas del sistema, no solo uname, debe estar seguro que la salida es la que se espera o su script del shell no funcionara.

2. *El comando uname* es también comúnmente usado en los scripts de shell para verificar el número de Lanzamiento o la Versión del sistema en uso. Por ejemplo, en muchos sistemas, *uname* imprime la información del lanzamiento en el siguiente formato.

   $ uname -r
   2.4.22

   El siguiente script de shell revisa a ver si el sistema actual que usted esta usando es lanzamiento 2.6.x (e.j. 2.6, 2.6.1, etc).

```
#!bin/sh
rel=’ uname -r | cut -f2 -d"."`
if [ $rel –ne 6 ]; then
  echo “Debe estar Ejecutando el Kernel 2.6”
     exit 1
     fi
      # continuar con el resto del codigo
     exit 0
```

Actualizar la base de datos de locate

- Use el programa **updatedb** para refrescar la base de datos utilizada por **locate**
- Versiones modernas son configuradas pasándole una opción a **updatedb**
  - -e provee una lista de los directorios donde no se buscara

- -f los nombres de los sistemas de archivos que no se incluirán
- Véase las paginas man para más detalles; **man updatedb**

- El comando **updatedb** es muy a menudo automatizado para ejecutarse todas las noches
  - Eche un vistazo en ***/etc/cron.daily*** para ver el script que lo ejecuta

updatedb.conf

- Versiones anteriores de GNU **updatedb** usaban el archivo de configuración en ***etc/updatedb.conf***
- Por razones de compatibilidad, algunas versiones modernas aun la leen
- La configuración se logra estableciendo variables de entorno
- Por Ejemplo, para ignorar ciertos sistemas de archivos:

  PRUNEPATHS="/tmp /usr/tmp /var/tmp /mnt /var/spool"
  export PRUNEPATHS
- La variable $PRUNEFS lista los nombres para los sistemas de archivos que deben ser ignorados (Ej., nfs, iso9660, etc.)
- Estas variables son equivalente a las opciones **-e** y **-f**

El comando whatis

- El comando **whatis** encuentra paginas man con el nombre dado y devuelve una lista
- Es solo útil si el nombre del comando ya es conocido
- Por Ejemplo, para encontrar páginas man acerca de bash:

  $ whatis bash
- La base de datos buscada es actualizada con el comando **makewhatis**
- Esto debe ser ejecutado cuando una nueva pagina man es instalada
- Debian mantiene un script cron en ***/etc/cron.daily/man-db***, el cual además elimina las paginas ya caducadas en ***cache***

Encontrar páginas Man con apropos

- El comando **apropos** es similar al **whatis**
- La diferencia es que cualquier palabra en el titulo de la pagina man puede coincidir
- El comando ***apropos palabra*** es idéntico a ***man -k palabra***
- Por Ejemplo, para encontrar comandos relacionados a directories:

  $ apropos directories
  $ man -k directories
- Cuando usemos estos comandos es preferible poner las **palabras** en ingles ya que existen muchas paginas man que aun no han sido traducida
- El comando **apropos** también utiliza la base de datos construida con el comando **makewhatis**

Establecer y Ver Cuotas de Discos

¿Que son las Quotas?

- Quotas es una manera de limitar el monto de espacio en disco que usuarios utilizan
- Algunas organizaciones (tal vez esas con usuarios externos no de toda confianza) que absolutamente tienen que asegurase de intrusos:
- Ningún usuario puede prevenir a otro de utilizar un espacio razonable en disco
- Ningún usuario puede impedir el correcto funcionamiento del sistema

- Algunas organizaciones no tienen este tipo de problema – todos sus usuarios son de confianza y no abusaran del sistema ni sus recursos
- Desafortunadamente, la administración de cuotas es un poco difícil en GNU/Linux
- Se podría educar a los usuarios para así no tener que utilizar cuotas?
- Espacio en disco hoy día es BARATO!

Limites Hard y Soft

- Las Cuotas tienen hard limits y Soft limits
- Un usuario puede exceder su soft limite sin penalización
- Pero solamente por un tiempo de gracia - grace period
- El usuario es advertido de que su soft limite ha excedido
- Un hard limite nunca puede ser sobrepasado
- Si un usuario trata de sobrepasar su hard limite (o su soft limite expirado), el intento fracasará
- El programa recibe el mismo mensaje de error como si al sistema se le ha agotado el espacio en disco
- Grace periods- Periodos de gracias se colocan por-sistema de archivos

Cuotas Por-Usuario y Por-Grupo

- La mayoría de las cuotas se establecen por-usuario
- Cada usuario tiene su propio limite soft y hard
- Cuotas también pueden ser establecida por-grupo
- Se le puede otorgar a un grupo limites soft y hard
- Cuotas de grupos aplican a todos los usuarios en el grupo
- Si el limite hard de un grupo se completo, ningún usuario en el grupo tendrá espacio disponible
- Incluyendo los usuarios que aun no han agotado sus limites de cuota

Limites de Block e Inode

- Quotas puede ser establecidas para los **blocks**
- Limita el monto de espacio que se puede utilizar para almacenar data
- Quotas también se puede establecer para los **inodes**
- Limita el número de archivos que pueden sercreados

Mostrar Límites de Quota: quota

- El comando quota muestra las quota disponibles
- Si especificas un nombre de usuario o de un grupo como argumento al comando quota se desplegara la información de quotas de ese usuario o grupo:
  # quota miguel
- La opción **-v** mostrara la información completa de todas las cuotas, aunque no tenga limites establecidos

Opciones en /etc/fstab

- Las opciones en ***/etc/fstab*** especifica cual sistema de archivos debe tener quota habilitado
- Agregue la opción **usrquota** para habilitar quotas de usuarios
- Use **grpquota** para habilitar quotas para los grupos
- Una o la otra se puede utilizar para cada sistema de archivos:

```
/dev/hda1      /                ext2    defaults
/dev/hdb1      /home            ext2    defaults,usrquota
/dev/hdb2      /work/shared     ext2    defaults,usrquota,grpquota
```

- El sistema de archivos con quota habilitado debe contener archivos llamados quota.user y quota.group en los directorios de root
- Los siguientes comandos los creará:

 # touch /partición/quota.{user,group}
 # chmod 600 /partición/quota.{user,group}

Habilitar Quota: quotaon

- El comando **quotaon** inicia el soporte para cuotas
- Solo puede ser ejecutado por el root
- Soporte debe ser compilado en el kernel, pero esto ya existe por defecto en todas las distros modernas
- El comando **quotaoff** deshabilita el soporte de **quota**
- Por Ejemplo, para habilitar quota en todos los sistemas de archivos:
  # quotaon -av
- Quota pueden ser encendida o apagada para los sistemas de archivos individuales

Cambiar Límites de Cuota: setquota

- Programa de línea de comandos para alterar los limites para un usuario o grupo
- Especifica el nombre de un usuario o grupo con **-u nombredeusuario** o **-g *nombredegrupo***
- Especifica el sistema de archivos a alterar después de las opciones **-u** o **-g**
- Finalmente, los límites a colocar deben ser especificados en la siguiente orden:
- Soft limites para los blocks
- Hard limites para los blocks
- Soft limites para los inodes
- Hard limites para los inodes
- Establecer cualquier límite a **0** removerá ese límite

EL comando edquota

- El comando **edquota** permite que las quotas sean editadas interactivamente, en un editor de texto
- El archivo en el editor de texto será un archivo temporal
- El comando **edquota** lo leerá al terminar el editor
- Use la opción **-g** para editar las cuotas de los grupos
- Algunas versiones de RedHat tienen un bug que necesitas borrar un espacio foráneo antes de que la unidad de tiempo antes de ejecutar **edquota -t**

El comando repquota

- El comando repquota imprime la información de los limites de quota asignado a cada usuario
- También muestra el número actual de blocks y inodes usados
- Use la opción **-a** para obtener información sobre todos los sistemas de archivos, o lo puedes especificar el sistema de archivos en la línea de comandos
- Use la opción **-g** para mostrar quotas de los grupos
- Use **-v** para una información más completa

Práctica9

Ejercicio 1

1) Use el comando **mount** para saber que sistema de archivos están montados.
2) Revise a ***/etc/fstab*** para ver si el floppy esta configurado apropiadamente, y saber su punto de montaje.
3) Monte un floppy en el punto de montaje por defecto.
4) Copie un archivo al floppy. Se escribe de inmediato?
5) Desmonte el floppy para asegurarse que todo se ejecuto apropiadamente, y que está bien removerlo.
6) Pruebe los comandos de arriba de crear un archivo para montarlo, y después de todo trate de copiarle archivos pequeños. Con el comando **df** revise el espacio disponible en el archivo. Desmonte ***/mnt/disk*** como lo hiciese con cualquier otro sistema de archivos.

Ejercicio 2

1) Revise su espacio libre en disco en su computador.
2) Muestre solo la información del uso para la partición que contiene a ***/usr/***. Muéstrela en unidades fáciles de lectura para los humanos.
3) Primero mire en el espacio libre e inodes de la partición ***/var/tmp***.
4) Entonces ejecute los comandos:

$ mkdir /var/tmp/prueba

$ seq -f '/var/tmp/prueba/bar-%04.f' 0 2000 | xargs touch

5) ¿Que ha pasados? Mire al espacio libre e inodes de nuevo.
6) Borre los archivos cuando haya terminado.

Ejercicio 3

1) Navega al directorio ***/var/***. Ejecute cada uno de los siguientes comandos como root, y explique la diferencia en las salidas:

 # du, du –h, du -h *, du –hs, u -hs *, du -hsS *, du -hsc *, du -bsc *

2) Investigue si **ls** ejecuta un programa directamente, o si es un alias de un shell o función.
3) ubique el binario del programa **traceroute**.
4) Use **whatis** para investigar que hace el comando **watch**.
5) Use **apropos** para encontrar programas que editan tablas de particiones de los discos.
6) Revise si su instalación de GNU/Linux contiene un ***updatedb.conf*** actualizada, y mire a su actual configuración.
7) Ingrese como root y actualice la base de datos de **locate** con el comando **updatedb**.


FCLD
http://www.codigolibre.org

Capítulo10

Un sistema operativo es un programa (o una colección de programas) que permite administrar los recursos de una computadora: Memoria, CPU, dispositivos de E/S (Unidades de Discos, monitor, teclado, etc). También proporciona un entorno para escribir programas de aplicación.

Definición de u Sistema Operativo

Conceptos de Arrancar y Deter el Sistema

Los Objetivos de este Capítulo son:

1. **El boot loader**
2. **Trabajar con LILO**
3. **Otras maneras de iniciar GNU/Linux**
4. **Especificar Parámetros del Kernel**
5. **Manejar los Runlevels**
6. **Apagar el Sistema**

Arrancar el Sistema

Boot Loaders (Cargadores de Inicio)

- Al iniciar **GNU/Linux**, el **kernel** se carga en memoria por un cargador (**boot loader)**
- Pasa parámetros al kernel de **GNU/Linux**
- Permite cargar uno de varios sistemas operativos
- . Múltiples versiones del kernel de GNU/Linux con una sola distribución
- Arranque dual (**Dual-booting) con** Windows y otros Sistemas Operativos
- El más popular de los gestores de arranque es LILO (el **GNU/Linux loader**)
- Completa documentación del usuario
- Busque un directorio de nombre similar a: ***/usr/share/doc/lilo/*** o ***/usr/doc/lilo-0.21/***
- La guía del usuario estará en un archivo de nombre ***user.ps*** o ***User Guide.ps***

LILO

- LILO se ejecuta al inicio del sistema
- El comando **/sbin/lilo** configura como LILO ejecutara en el próximo arranque
- El archivo ***/etc/lilo.conf*** específica la configuración y los parámetros a establecer por el comando **lilo**
- Necesitas ejecutar el comando **/sbin/lilo** para que los cambios tomen efectos
- Pagina de Manual de ***lilo.conf (5)***
- ***lilo.conf*** tiene opciones de la forma siguiente: ***nombre = valor***
- Opciones especificas para Sistemas Operativos
- **Kernel GNU/Linux** son introducidos con: image= /ruta/donde/se/guarda
- Otros **SOs** son introducidos con: other=Windozze
- Otras opciones genéricas, o que son por defecto para los SOs

Ejemplo Archivo de Configuración lilo.conf

```
boot = /dev/hda # poner el cargador en el MBR
root = /dev/hda1 # dispositivo a montar en /
delay = 40 # retraso de 4 segundos
compact # puede lograr que el inicio sea más rápido
read-only # necesario para permitir al root que se le ejecute fsck
image = /vmlinuz-2.4.20 # kernel estable (por defecto por ser el 1st)
label = Linux-2.4.20
alias = Linux # Un nombre más corto
vga = ask # Nos permite elegir el tamaño de la consola
image = /vmlinuz-2.5.1 # kernel de desarrollo
label = Linux-2.5.1
other = /dev/hda3 # Windows instalada en una partición diferente
label = windozze
table = /dev/hda
```

Seleccionando que Arrancar

- Cuando LILO se ejecuta despliega el prompt LILO:
- Si solo parte de las letras aparecen, el proceso de inicio fracaso en algún punto
- El prompt espera el tiempo especificado en la variable **delay**
- Puedes cargar unas de las opciones de kernel o SOs solo con digitar su label o alias
- Solo debes presionar enter para Entrar en el por defecto
- Presionando a Tab te lista los labels disponibles
- Algunas versiones de LILO te presentan un menú para seleccionar con las teclas del cursor
- Si al final del delay ninguna tecla es presionada, el primer kernel o SO se cargara

Otra manera de Iniciar GNU/Linux

- Grub - El reemplazo de LILO, el por defecto en las mayorías de distribuciones, más potente y reconoce más sistemas operativos
- Loadlin - un programa DOS que puede iniciar GNU/Linux desde el DOS
- Usado especialmente después que un driver DOS configuro cierto hardware

Especificar Parámetros del Kernel

- Los kernel GNU/Linux toman parámetros que afectan su ejecución
- Parámetros pueden ser especificados en el momento de arranque:
    - En el prompt de LILO
    - Después del label de la imagen

  LILO: Linux-2.2.20 root=/dev/hda3
- Especifica el sistema de archivos root
- Los detalles de estos parámetros se encuentran en el famoso HOWTO: BootPrompt

Especificar Parámetros del Kernel en lilo.conf

- Parámetros Kernel también pueden ser especificado en lilo.conf
- Sensible a probarlo primero en el prompt de LILO
- Parámetros comunes tienen opciones de nombres lilo.conf
- Cualquier parámetro puede ser establecer con la opción de append

```
image = /vmlinuz-2.2.0
```

```
label = GNU/Linux-2.2.20
root = /dev/hda3
append = "hdc=ide-scsi"
```

Parámetros Útiles del Kernel

- root=device - Establece el sistema de archivos a montar como root
- ro y rw - monta el sistema de archivos root como (solo lectura) read-only o read-write (lectura y escritura), respectivamente
- Lo normal es que sea solo lectura en ***lilo.conf***, para permitir el **fscks**
- nfsroot=server... - usado en sistema de archivos de redes como root (estaciones de trabajos diskless)
- init=program - El nombre del primer programa a ejecutar el kernel, cual casi siempre es **/sbin/init**
- Se puede establecer a **/bin/sh** si el **init** esta roto
- Existen otros parámetros para configurar módulos de hardware específico

Mensajes de Arranque (Boot Messages)

- Cuando el kernel inicia imprime mucha información en la pantalla
- Esta información puede ser muy útil para diagnosticar problemas
- Un diario "log" de esta información se mantiene en **/var/log/dmesg**
- El comando dmesg puede imprimir el mensaje más reciente
- Esto puede mostrar problemas que ocurrieron desde el último arranque
- Después del boot, la mayoría de los mensajes log son manejados por el syslog

Módulos del Kernel

- Muchas características del kernel GNU/Linux pueden cargadas como módulos
- Pueden ser cargadas a medida que se necesiten, y más tarde descargadas
- Módulos compilados se almacenan en ***/lib/modules/***
- Estos comandos administran los módulos:
    - lsmod - lista los módulos actualmente cargados
    - rmmod - remueve módulos que no están en uso
    - insmod - carga un módulo
    - modprobe - carga un módulo, y cualquier otro que se necesite
- El archivo ***/etc/modules.conf*** configura estos comandos
    - ***/etc/conf.modules*** en algunos sistemas
    - Tiene su página man, ***modules.conf (5)***

Cambiar Runlevels y Apagar o Reiniciar el Sistema

Entender los Runlevels

- Un sistema GNU/Linux ejecuta en diferentes niveles de ejecución denominados **runlevels** - modos que proveen diferentes características y niveles de funcionalidad
- Los sistemas GNU/Linux normalmente tiene siete **runlevels**, enumerado del 0-6:
- De los cuales tres son obligatorios **(0 = halt, 6 = reboot, 1 = single-user)**
- Cuarto son definido por el usuario (2-5)
- No existe un consenso entre los administradores ni las distribuciones en como organizar los runlevels definidos por los usuarios

- Algunos dependen (parcialmente) de runlevels para definir cuales subsistemas se están ejecutando
- Otros prefieren la flexibilidad de arrancar y detener subsistemas individualmente, sin cambiar el runlevel
- En toda distribución, existe al menos un runlevel definido por el usuario el cual tiene los mismos servicios que otros

Runlevels Típicos

Descripción de los Runlevels

0 Powerdown -un runlevel de transición, usado para decirle al sistema que se apague de manera segura. Una ves complete este nivel y se apague el sistema deberá ser encendido manualmente.
1 Single-user mode, usado para dar mantenimiento. Usuarios no podrán ingresar, la gran mayoría de los servicios (incluyendo todo el networking) no están disponibles. Solo un terminal esta disponible, y root es ingresado automáticamente.
2-5 Modos Multi-usuarios. En algunos sistemas todos estos niveles son idénticos. Otros deshabilitan redes (o NFS y compartir archivos) en runlevel 2, y/o habilitan un login gráfico en el runlevel 5 (pero no entro runlevels).
6 Otro runlevel de 'transición', usado para pedirle a sistema que reinicie.

Modo de Usuario Único (Single-User Mode) y el sulogin

- Muchas distribuciones GNU/Linux usan in programa llamado sulogin para restringir acceso single-user mode
- **sulogin** se ejecuta cuando el sistema entra en **single-user mode**
- Requiere el password de root en la consola antes de cambiar a single-user mode
- Si no se ingresa el password, sulogin retorna el sistema a al runlevel normal
- Porque es necesario sulogin?
- Usuarios no confiables pueden tener acceso al teclado durante el arranque
- En muchas configuraciones, esto permitiese que inicien el sistema en modo de usuario único o single-user mode

Apagar y reiniciar el Sistema

- Para apropiada y con seguridad apagar su sistema, ejecute el comando halt como root
- Esta es la manera más segura de apagar un sistema: detiene todos los servicios, deshabilita todas las interfaces de redes, y desmonta todos los sistemas de archivos
- Para apropiadamente reiniciar, ejecute el comando **reboot** como root
- La mayoría de los servicios le permiten desde la consola pulsar **Ctrl+Alt+Del**
- Alternativamente, el comando shutdown le permite programar un powerdown o reinicio, para darle tiempo a los usuarios ingresados tiempo para que salven sus trabajos
- Apagar a las 6pm:
  # shutdown -h 18:00
- Reiniciar en treinta minutos:
  # shutdown -r +30

Establecer el Runlevel Por Defecto

- El runlevel por defecto del sistema se configura en el archivo ***/etc/inittab***
- Para configurar un runlevel por defecto en 3, ***/etc/inittab*** debe contener la siguiente línea:
  id:3:initdefault
- Solo debe existir un solo debe existir una línea con **initdefault** en ***/etc/inittab***

Seleccionar Diferente Runlevel al Inicio

• La mayoría de los gestores de arranque (incluyendo LILO) le dan la habilidad a ingresar argumentos en la línea de comandos del kernel (**kernel command line)**
• Nombrar un runlevel en la línea de comandos del kernel selecciona que runlevel ha usar en el sistema al tiempo de inicio
• Para iniciar en single-user mode: **linux 1**
• Para iniciar en modo de emergencia: **linux -b**
Modo de Emergencia provee nada más que un shell para ingresar comandos - útil para reparar corrupción seria de archivos

Determinar el Runlevel Actual

- El comando **runlevel** imprime los runlevels actual y anterior:
  $ /sbin/runlevel
  N 3
- Si no hay un runlevel previo (por Ejemplo, si el runlevel no se ha cambiado desde el default), N es se imprime para indicarlo.

Cambiar de Runlevel

- El sistema tiene un proceso llamado el **init**, con un **pid de 1**, el proceso ancestral de todos los procesos
- El **init** es responsable de controlar los runlevels, así es que para cambiar de runlevels es necesario decírselo al init, algo similar a esto:
- Ejecutado como root
  # telinit 1

para cambiar al runlevel indicado

- Puedes alternativamente usar al mismo **init**, con la misma sintaxis:
  # init 5
- Obviamente, cambiar de runlevels no debe ser tomado a la ligera
- En particular, cambiar de runlevel puede terminar servicios importante y afectar la disponibilidad de ingreso al sistema a usuarios (log-in)

Servicios en cada Runlevel: el directorio init.d

- ***/etc*** contiene un directorio ***init.d***, y uno ***rcN.d*** para cada runlevel N
- Algunas distribuciones (muy notable Red Hat) ponen todos los directorios en ***/etc/rc.d***, no directamente debajo del ***/etc***
- ***init.d*** contiene un **script de init script** para cada servicio que puede ser iniciado
- El directorio ***rcN.d*** contiene los vínculos simbólicos a los scripts de inicio, **init scripts**

• Estos symbolic links controlan cual servicio esta disponible en cada runlevel

Vínculos Simbólicos en rcN.d

- Vínculos Simbólicos en el directorio ***rcN.d*** o son vínculos de iniciar o detener servicios (**Stara links** o **stop links)**
- **Start links** son nombrados ***SNNservicio***, donde ***NN*** es un número y ***servicio*** es el nombre del servicio
- Stop links son nombrados *KNNservicio*
- Los vínculos de inicio en el directorio (**start links)** de un runlevel indican cual servicio debe iniciarse al

entrar en ese runlevel
- Así por igual, los **stop links** indican cuales servicios deben ser detenidos al entrar en ese runlevel
- El shell script **rc** (en ***/etc/rc.d/rc*** o ***/etc/init.d/rc***) ejecuta los scripts de init apropiados para los **start links** y **stop links**

Arrancar y Detener Servicios Individuales

- Puedes Detener o Iniciar servicios individualmente sin cambiar de niveles de ejecución
- Un script de init siempre toma un argumento de **start** o **stop** para iniciar o detener u servicio
- Por Ejemplo, Si el servidor de base de datos **MySQL** tiene su script de init en ***/etc/init.d/mysql*****,** puedes iniciarlo con:

 # /etc/init.d/mysql start

o detenerlo con:

 # /etc/init.d/mysql stop

- Algunos scripts de init también aceptan argumentos de reiniciar, recargar y estatus (restart: stop y luego start; reload: recargar el archivo de configuración del servicio)

Práctica10

Ejercicio 1

1) Revise los módulos compilados y disponibles en el sistema
2) Liste los módulos actualmente cargados.
3) Cargue el modulo **parport**, y revise que funciono
4) Descargue el modulo **parport**, y revise otra vez
5) **¿**Intente descargar un modulo que este actualmente en uso. Que paso?

Ejercicio 2

1) Reinicie el computador. Puedes hacer esto con cuidado saliendo de todos sus programas, logging out, y entonces pulsando **Ctrl+Alt+Del**. Cuando aparezca el prompt de LILO, listara las seccion disponibles. Cargue por defecto.
2) Reinicie otra vez. Esta vez pásele al kernel el parámetro **init=/bin/sh**. Que sucede?
3) ¿En cual directorio esta usted?
4) Cual es la salida del comando hostname?
5) ¿Puedes crear un nuevo archivo? Salga del shell y reinicie de nuevo

Ejercicio 3

1) Has un backup de lilo.conf, luego agrega una nueva sección al final de la original:
2) Copie las opciones para el Kernel por defecto.
3) Cambia el label a “shell” (y remueva cualquier alias).
4) Establezca el primer programa que ejecute el kernel sea /bin/sh. Haga los cambios en vivo, entonces reinicie para probarlo. Luego retorne y restablezca su lilo.conf.

Ejercicio 4

1) Mire en ***/etc/init.d*** o ***/etc/rc.d/init.d*** para ver que servicios pueden ser iniciados por **init**.
2) Trate de ejecutar el script para el **crond**, y usarlo para detener el servicio de **cron**, y arránquelo de nuevo.
3) Tome un vistazo al programa en un editor de texto (es un pequeño script shell) para tener una idea de lo hace y como lo logra.
4) Mire en el directorio ***rc3.d*** para ver que servicios son exterminados (killed) e iniciados durante el cambio al runlevel 3.
5) Use la utilidad **telinit** para cambiar al modo de usuario único (single-user mode).
6) Una vez en single-user mode, use **top** para ver los procesos que aun se ejecutan.

Ejercicio 5

1) Reinicie la computadora cambiándose al nivel 6.
2) Al momento que aparece el prompt de LILO, pulse **Tab** para listar los sistemas operativos disponibles para arrancar. Escriba el nombre del que deseas iniciar seguido por un espacio y el número 1, para indicarle que deseas iniciar en modo de usuario único (single-user mode).
3) Cambie al runlevel 3.


FCLD
http://www.codigolibre.org

Glosario

@ at, en (y no “arroba”)

/ ver slash
\ ver backslash

A

abort- fracaso, interrupción
abort (v)- abortar, fracasar, interrumpir, cancelar (fuera del contexto informático, podrá ser también abortar)
address- dirección
Advanced Power Management (APM)- gestión avanzada de potencia, gestión avanzada de energía
age- edad, antigüedad
alias- alias, acceso directo
allocate (v) -asignar, reservar
alphanumeric -alfanumérico
ampersand &- y (caracter empleado en programación C para señalar direcciones de memoria, y en html para codificar otros caracteres extraños a la lengua inglesa, del latín “et”, al)
anchor-ancla, áncora, anclaje (enlace)
anchor (v) -anclar
anti-aliasing -suavizado de bordes, antisolapamiento
append (v) -juntar, unir, concatenar, añadir
applet -miniaplicación, aplique, applet (programa en Java, ejecutable por un navegador; dicese tambien de cualquier pequeño programa que se acopla al sistema)
Application Program Interface (API) -interfaz de programación de aplicaciones
appraisal -estimación
archive -archivo, paquete (como “archivo” es muy usado también para traducir “file”, puede ser necesario aclarar de que tipo de archivo se trata)
archive (v) -archivar, empaquetar
argument -argumento, parámetro
array -arreglo, formación, estructura, matriz, vector (“arreglo” es considerada por algunos una mala traducción, pero su uso ya está bastante generalizado)
Artificial Intelligence (AI) -inteligencia artificial
ascender -ascendente
ASCII-Armoured text -texto con armadura ASCII
assapps -applet
assembler -1 ensamblador (lenguaje de programacion) 2 montador o ensamblador (persona que monta ordenadores)
assembly -lenguaje ensamblador
assessment -estimación, juicio, impresión
assignment -asignación
associative array -vector asociativo, arreglo asociativo (array es en ocasiones utilizado como arreglo, a pesar de que algunos no concuerden)
Asymmetric Digital Suscriber Line (ADSL) -línea digital asimétrica de abonado
attach (v) -adjuntar, anexar, anexionar
attachment -documento adjunto, anexo
attribute -atributo
authentication -autenticación, autentificación
autoprobe -autocomprobacion

B

back-end -motor (de un compilador o programa), dorsal
backbone -eje principal, red troncal, estructura principal
background -segundo plano, trasfondo
backslash -barra invertida, contrabarra
backup -copia de seguridad
backup (v) -respaldar, hacer copias de respaldo
backward compatible -compatible con anteriores
bandwith -amplitud de banda, ancho de banda

banner -pancarta, aviso
baseline -línea de base, directrices (condiciones generales que un programa, proceso o producto debe cumplir)
batch -lote
batch processing -procesamiento por lotes, procesamiento en lotes
batcher -procesador por lotes
baud -baudio (unidad de medida de la velocidad de transmisión de información)
benchmark -banco de pruebas, prueba comparativa, hito
big-endian -byte más significativo primero
bind (v) -enlazar, ligar
binding -enlace, ligadura
bit -bit (unidad elemental de información, consistente en una variable booleana, con valores 0 o 1)
bit mask -máscara de bits
bitmap -mapa de bits
bitrate -tasa de bits
block -bloque
block (v) -bloquear (impedir el acceso)
blur (v) -tornar más difuso, emborronar
bookmark -marcador, marca-páginas
boot -arranque, inicio, proceso de arranque
boot (v) -arrancar, iniciar
bootrom -ROM de inicio
bootstrap -rutina de arranque, arranque autónomo
bot -final
breakpoint -punto de ruptura, punto de corte
broadcast -difusión, broadcast
broadcast (v) -anunciar, difundir
browser -navegador, visualizador, ojeador (navegador es más usada cuando se trata de hipertexto y visualizador en otros casos Existe alguna polémica acerca de "visualizador" y han sido propuestas otras posibilidades como visor o examinador, que no son muy usadas)
brush -pincel, brocha
bubble sort -ordenación por el método de la burbuja
buffer -búfer, memoria tampón, memoria intermedia
bug -error, fallo, gazapo (gazapo ha sido propuesta por algunos especialistas, pero no es muy usada)
bug-fix -corrección de fallo
built in -incorporado, incluido
Bulletin Board System (BBS) -tablón de anuncios electrónico, foros, sistema de foros
burst page -página en bruto, página de separación (página añadida por muchos gestores de impresión para separar los trabajos)
bus -bus, línea de datos, cable de datos
byte -byte, octeto (unidad de información compuesta por ocho bits; una variable de 1 byte puede contener 256 valores diferentes)

C

cache -almacén, deposito (algunos usan caché que suena parecido mas no traduce bien su significado)
cache memory -antememoria, memoria inmediata, memoria cache (ver cache)
callback -retrollamada
camel caps -mayúsculas mediales
camera ready -preparado para cámara, preparado para su publicación (se usa para indicar la manera de mandar artículos a una revista listos para su publicación)
canvas -lienzo, tapiz
capability -capacidad
caps -letras mayúsculas
card -tarjeta
cardinality -cardinalidad
caret -circunflejo (el símbolo o acento ^ usado para mostrar que algo va a ser insertado en material escrito o impreso en el lugar en el que se encuentra)
case sensitive -distingue mayúsculas de minúsculas
cast molde, plantilla
catch-up (v) -actualizarse, ponerse al día
cellular automata -autómata celular
channel -canal

character set -conjunto de caracteres (conjunto de signos que se representan mediante un código El más conocido de estos códigos es el ASCII, que utiliza los 256 caracteres que se pueden representar con un byte)
chat -chat, charla, tertulia
chat (v) -chatear, conversar, charlar
check button -botón de verificación
check out (v) -descargar
checkbox -caja de selección, casilla de selección
checker -1 Corrector, cuadrado de un tablero de ajedrez, cajero
checkpoint -punto de control
checksum -suma de control, suma de verificación, suma de comprobación
chess -ajedrez
chief architect -desarrollador jefe
child process -proceso hijo
chip -chip, circuito integrado
chipset -chipset, conjunto de chips
choke -obturador, estrangulador, sofocamiento
class -clase
clause -cláusula
clean -limpio
clean (v) -limpiar, despejar
clear (v) -borrar
click -click, pulsación
click (v) -hacer clic, pulsar, pinchar
client -cliente
clipboard -portapapeles
clock rate -velocidad de reloj
clone -clon
closure -clausura, cierre
clumsy -torpe, difícil de manejar
cluster -grupo, cúmulo
cluster (v) -agrupar
coder -programador, codificador, codificador
cold boot -arranque en frío
colon -dos puntos (signo de puntuación :)
command -comando, orden, instrucción, mandato (el uso de “comando” está bastante generalizado, aunque algunos lo consideren erróneo)
commit (v) -enviar, comprometer, aplicar, llevar a cabo, efectuar
Common Gateway Interface (CGI) -interfaz común de acceso (un estándar para elaborar pequeños programas que permiten la interacción entre un navegador y un servidor web)
compile (v) -compilar
compiler -compilador
compliant -en conformidad, conforme con, compatible
compose (v) -redactar
composer -Redactor (de correo, por ejemplo), compositor (de música)
compress (v) -comprimir
compression -compresión
computable -calculable
computer -computadora, ordenador, computador
Computer Aided Design (CAD) -diseño asistido por ordenador (computadora)
computer nerd -empollón informático
concatenate (v) -concatenar
concurrency -concurrencia, simultaneidad (término usado para expresar la capacidad de realizar varias tareas a la vez)
conjunction -conjunción (conector lógico de dos proposiciones que en castellano se expresa mediante la conjunción “y”; el valor de la conjunción de dos proposiciones es cierto cuando las dos proposiciones son ciertas; en los otros tres casos, el valor de la conjunción es falso)
connect (v) -conectar
connected graph -grafo conexo
cons -contras
constraint -restricción
constructor -constructor
context -contexto
converse -contrario, opuesto
converse (v) -conversar

converter -convertidor, conversor
convex hull -envoltura convexa, cierre convexo
cookbook -recetario
cookie -galleta (mensaje enviado por un servidor web a un navegador para que éste lo guarde en el ordenador del usuario y sea enviado de nuevo al servidor, cada vez que el usuario consulta una de sus páginas)
coprocessor -coprocesador
copyleft -copyleft, derecho de copia
copyright -copyright, derechos de autor
copyrighted -sujeto a derechos de autor
cordless -inalámbrico
core- corazón, núcleo, motor (program core: motor del programa; ver también "core file")
core dump -volcado de memoria
core dump (v) -Hacer un volcado de memoria (cuando un programa acaba de forma inesperada)
core file -archivo (fichero) core, archivo (fichero) imagen de memoria, archivo (fichero) de volcado de memoria
core voltage -voltaje interno
courseware -software de apoyo (a cursos de formación)
cover -portada
Central Processing Unit (CPU) -unidad central de proceso
crack (v) -invadir, penetrar
cracker -cracker, maleante informático
crash -ruptura, caída (del sistema)
crash (v) -colgarse (un ordenador), fallar (un programa)
crawler -gateador
cross-assembler -ensamblador cruzado
cross-compiler -compilador cruzado
cross-platform -multiplataforma
cross-post -envío cruzado, envío múltiple, correo con destinatarios múltiples (envío de un mismo mensaje a múltiples grupos de noticias)
cue point -punto de referencia
current -Actual, en vigor, en curso, corriente (por ejemplo eléctrica)
cursor -cursor
customize -personalizar
cut and paste (v) -cortar y pegar
cyber -cíber (prefijo griego Todo aquello relacionado con la comunicación empleando medios electrónicos)
cyberspace -ciberespacio (es decir, el espacio de la comunicación)

D

daemon -demonio, proceso en segundo plano, duende (proceso de ejecución independiente)
daisy chain -conexión en serie
daisywheel printer -impresora de margarita
dash -raya
database -base de datos
datagram -datagrama
de facto standard -estándar de hecho, norma de facto, regulación de facto
dead lock -bloqueo mutuo, abrazo mortal
deadlock -interbloqueo
debug (v) -depurar, corregir errores (en un programa)
debugger -depurador
declarative language -lenguaje declarativo
decode (v) -decodificar, descodificar
decoder -decodificador, descodificador
default -por omisión, de manera predeterminada, predefinido, por definición
default file -archivo predeterminado, fichero predeterminado
deferral -posposición
deflate (v) -deshinchar
defragment (v) -desfragmentar
delay -demora
delete (v) -borrar, eliminar
delimiter -delimitador, separador
demo -demo, demostración
demodulate (v) -desmodular, traducir tonos a señales digitales (en un modem)

denial of service -rechazo de servicio, denegación de servicio
deny (v) -denegar, recusar
descender -descendente
descriptor -descriptor
desktop -escritorio
detach (v) -descolgar, desenganchar, separar
developer -desarrollador
device -dispositivo
devise (v) -inventar, diseñar, planear
dial-up link -enlace telefónico, enlace por red telefónica
dial-up login -ingreso por red telefónica
dialog box -cuadro de diálogo, caja de diálogo
diffusion -difusión
digest -recopilación, resumen
dike (v) -contener
directory -directorio
disclaimer -renuncia de responsabilidades, descargo
discussion groups -grupos de debate
dispatch (v) -despachar, enviar
display -pantalla, visualizar
display (v) -mostrar
display menu -menú de visualización
disposable -desechable
distribution -distribución
dithering -difuminado
documentation -documentación
doorstop -tope (de una puerta)
dot matrix printer -impresora de matriz de puntos
down -fuera de servicio
downgrade -versión anterior
downgrade (v) -Menoscabar, disminuir, instalar una versión anterior
download (v) -descargar, transferir, recibir, bajar, obtener
downsizing -reducción, disminución
downstream -flujo descendente
downstream port -puerto de recepción
downtime -tiempo de inactividad
draft -borrador
drag and drop -arrastrar y soltar
drill -ejercicio, entrenamiento
driver -controlador, manejador, gestor, driver (driver de video, driver de sonido)
dumb -sin procesamiento, bobo, pantalla tonta
dumb terminal -Terminal sin procesamiento
dummy -mudo
dump -volcado, vuelco
dungeon -mazmorra

E
e-mail -correo electrónico, mensaje (send me an e-mail: envíame un mensaje)
eg -por ejemplo (del latín exemplia gratia; en castellano se usa vg dellatín verbi gratia)
edge -límite
electronic mail -correo electrónico
elevation grids -mapas de elevación
ellipse -elipse
embed (v) -empotrar, embeber
embedded -empotrado, embebido
enable (v) -activar
enable (v) -habilitar
enabling -habilitación
encode (v) -codificar
encoder -codificador
encryption -cifrado, encripción, encriptación

endian -vease “big-endian” y “little-endian”
endless -interminable
enhancement -mejora
enlarge (v) -ampliar
entity -entidad
entries -entradas, líneas, renglones
environment -entorno, ambiente
erase (v) -borrar
error -error
escape (v) -escapar, preceder con escape, exceptuar
evaluator -evaluador
event -evento, suceso
event-driven -basado o gestionado por eventos, orientado a eventos, dirigido por eventos
executable -ejecutable
execute (v) -ejecutar
expire time -tiempo de caducidad
extrication -liberación, rescate, extricación

F

facility -instalación, equipo
fade in -comienzo gradual
fade out -final gradual
fade (v) -atenuar, desvanecer
failure -fallo
fake -falso
feature -funcionalidad, característica, dispositivo
feed -fuente, suministro
feed (v) -suministrar
feedback -realimentación, comentarios y sugerencias, retroalimentación
fetch (v) -obtener
field -campo
file -archivo, fichero (la mayoría de las personas usan exclusivamente una o la otra)
file (v) -archivar
file system -sistema de archivos, sistema de ficheros
filehandle -identificador de ficheros (programación), descriptor de archivos (ficheros), manejador de archivos (ficheros)
fill rate -tasa de relleno
filter -filtro
fingerprint -huella dactilar, huella digital
firewall -cortafuegos
firmware -microcódigo, soporte lógico incorporado
fix -enmienda, corrección
fix (v) -Corregir, arreglar, reparar, enmendar, fijar
flag -bandera, indicador, parámetro
flame -llama, insulto destructivo, comentario airado, crítica destructiva, soflama
flanger -desdoblador
flat shading -sombreado plano
flip (v) -voltear
floating -flotante
floating point -punto flotante, coma flotante (en diferentes países se usa el punto o la coma para separar dígitos enteros y decimales)
floppy disk -disquete, disco flexible
flow chart -diagrama de flujo
flush (v) -vaciar
folder -carpeta, directorio
follow-up (v) -responder (a un grupo de noticias)
font -tipo de letra (algunos usan “fuente”, por su parecido con el término inglés, que no traduce bien su significado)
footprint -huella, rastro
foreground -primer plano, interactivo
foreign agent -agente externo
fork -bifurcación
fork (v) -bifurcar, desdoblar
format -formato

format (v) -dar formato, formatear
forum -foro
forward (v) -reenviar, remitir, redireccionar, adelantar
fragmentation -fragmentación, partición
frame -marco, fotograma
frame buffer -memoria de imagen, marco de memoria intermedia
frame relay -conmutación de tramas
frames -cuadros
framework -infraestructura, armazón
front end -entorno, interfaz, fachada, frontal
fully qualified domain name -nombre de dominio completo
function -función
function inlining -expansión de funciones (se copia la función entera en lugar de hacer una referencia a la misma)
further -consiguiente, posterior, más extenso, más avanzado
fuzzy -difuso

G

gateway -pasarela, portal, compuerta, puerta de enlace
gaussian blur -desenfoque gausiano
getting started -primeros pasos
glyph -glifo
grab (v) -capturar
graph -grafo, gráfico
graphic display -representación gráfica
Graphical User Interface (GUI) -interfaz gráfica de usuario
Graphics Interchange Format (GIF) -formato para intercambio de gráficas
grid -rejilla, grilla, cuadrícula
guidelines -directivas
gzipped -comprimidos con gzip, comprimidos, compactados

H

hack -adecuación, alteración (“a quick hack”)
hack (v) -alterar, modificar (“hack a program”)
hacker -hacker, genio de la informática (no confundir con “cracker”), experto en informática
handheld -de mano
handle (v) -manipular
handler -manipulador
handover -traspaso (de un nodo móvil desde una subred a otra)
handshaking -asentimiento, negociación, sincronismo
hang (v) -colgar, colgarse, bloquearse
hard disk -disco duro, disco rígido, disco fijo
hard link -enlace físico, enlace rígido, enlace duro
hardware -hardware, máquina, equipo, dispositivo, soporte físico
hash -resumen criptográfico, picadillo, arreglo asociativo (Perl)
hash table -tabla de dispersión, tabla de referencias, tabla hash
hassle -lío, enredo, complicación
hassle (v) -molestar, confundir
header -cabecera (header file), encabezado, encabezamiento (page header)
heap -montón
heuristic -heurístico
hi-color -color de alta densidad
hi-tech -tecnología de punta
hide (v) -esconder, esconderse
hierarchy -jerarquía
high-color -color de alta densidad
high-tech -tecnología de punta
highlight -realce, destaque
highlight (v) -realzar, destacar, resaltar
hit -golpe, éxito, acierto, visita (a una página web)
hits -golpes, accesos (en una web)
home -casa, portada (ver también home page)
home agent -agente local

home directory -directorio del usuario, directorio principal del usuario
home page -página principal, página inicial
host -anfitrión, máquina anfitriona, puesto
host (v) -alojar
hostname -nombre de anfitrión
hub -concentrador, distribuidor
hyphen -guión
hyphenate -enguionar, cortar palabras incorporando guiones

I

ie -esto es, o sea (del latín id est)
icon -icono, icono
iconize (v) -miniaturizar, iconizar
idle -ocioso, inactivo
illustrator -ilustrador
imaging -proceso de imágenes, trabajo con imágenes, diseño gráfico, diseño de imagen, generación de imagen, ilustración
inbox -bandeja de entrada
indent (v) -sangrar (empezar un renglón más adentro que los otros)
indentation -sangría
index -índice
indexed -indexado
inflate (v) -inflar (descomprimir)
inherit (v) -heredar
inheritance -herencia
inkjet -inyección de tinta
inode -nodo i, inodo
input encoding -codificación
installer -instalador, asesor para la instalación
instance -instancia, ejemplar
interactive -interactivo
interface -interfaz (femenino), definición de gestión de hardware
interlace (v) -entrelazar, interfoliar
interlaced -entrelazado
Internet -Internet
Internet Protocol (IP) -protocolo Internet
interpolation -interpolación
interrupt -interrupción
Interrupt Request (IRQ) -Solicitud de interrupción, petición de interrupción
introducer -presentador
isochronous -isócrono (del prefijo griego iso, igual, y de la palabra griega crono, tiempo)
isomorphism -isomorfismo
italic -cursiva
item -elemento, objeto
iteration -iteración (del latín iteratio, -onis)

J

jabber -torrente de palabras ininteligibles
jabber (v) -hablar mucho, hablar incoherentemente, farfullar
jagged picture -imagen serrada
jigsaw puzzle -rompecabezas
jitter -ruido, nieve
job -trabajo
journaling file system -sistema de ficheros transaccional
joystick –vídeo-mando, ludomando, mando para jugar, palanca para juegos
jumper -puente, puente deslizable, puente configurable, conector
junk-mail -correo basura
justify (v) alinear

K

kernel -núcleo
kerning -interletraje (ajuste de espacio entre ciertos pares de caracteres para que estos se impriman con un toque estético)

key - llave, tecla, clave, tono, tonalidad, crucial, de importancia, significante
key escrow -depósito de claves
key fingerprint -huella de clave
key pair -par de claves
keyboard -teclado
keyboard shorcuts -métodos abreviados de teclado
keyring -anillo de claves, archivo de claves
keyword -palabra clave
kit -conjunto, juego, paquete
knowbot -robot, buscador, buscador en la red (programa que busca y clasifica información automáticamente en una red, a diferencia de buscador en una base de datos propia)

L

label -etiqueta
latency -latencia
lattice -red, trama
layer -capa
layout -esquema, diseño, composición, gestor de geometría (en algunos programas gráficos)
leak -fuga (de un gas o líquido por un agujero), escape, pérdida
legalese -condiciones legales, jerga legal
library -librería, biblioteca (cuando library se refiere al edificio donde se almacenan libros, sin lugar a dudas que la traducción correcta es biblioteca; pero en el contexto informático es más usada librería, ya que además de una tienda de venta de libros, librería también es un mueble donde se guardan documentos)
lightning effects -efectos de iluminación
line -línea, renglón
line wrap -encapsulamiento de línea, retorno automático de líneas
link -enlace, vínculo, liga, eslabón
link (v) -enlazar, conectar, vincular, crear vínculos
linker -enlazador
Liquid Cristal Display (LCD) -pantalla de cristal líquido
list view -lista de elementos
little-endian -byte menos significativo primero
Local Area Network (LAN) -red de área local
lock -cerrojo, candado, cerradura, bloqueo
lock (v) -cerrar con llave, trancar
lock file -fichero de bloqueo
log -registro, bitácora
log (v) -registrar
log in (v) -ingresar, entrar en, comenzar la sesión, entrar al sistema, conectarse
log on (v) -ver “log in”
log out (v) -salir de
login -ingreso
login banner -mensaje de ingreso, mensaje de bienvenida
look and feel -aspecto y funcionalidad, aspecto visual y operacional
loop -ciclo, bucle
loopback -circuito cerrado
lossy -con pérdida, perdida, compresión resumida, compresión con pérdida (de información)
luminance -luminancia
lvalue -valor a la izquierda, valor-l

M

mail -correo, mensaje
mail (v) -enviar por correo
mail hub -distribuidor de correo
mailbox -buzón
mailer -gestor de correo, agente de correo, corresponsal, cartero
mailing list -lista de correo, lista postal, lista de distribución
mainframe -macrocomputadora, ordenador de escala superior
maintainer -responsable del mantenimiento, encargado del mantenimiento
map -mapa
map (v) -mapear, asignar
markup marcado

mask -máscara
mask (v) -enmascarar, ocultar
masking -enmascaramiento
masquerading -enmascarado, enmascaramiento, mimetización
master -maestro, amo
match -concordancia (objeto o persona que se encuadra bien con otra)
match (v) -coincidir, encuadrar, encajar, concordar
measure -medida, métrica
merge (v) -mezclar, fusionar, incorporar
mesh -malla
message digest -condensado de mensaje
mirror -réplica
mirror site -réplica
misplaced -extraviado
mistake -equivocación, error
mix (v) -mezclar
mixer -mezclador
mobile IP protocol -protocolo IP móvil
mobile node -nodo móvil, ordenador móvil
modem -modem
monitor (v) -supervisar, controlar
mount (v) -montar
mouse ratón

N

named pipes -tuberías designadas, tuberías con nombre, cauces designados
nest (v) -anidar, conectar
nested -anidado
netmask -máscara de red
newbie -principiante
news feed -proveedor de noticias, fuente de noticias, suministro de noticias
newsgroups -grupos de noticias, grupos de discusión, foros de discusión
nickname -apodo
noise gate -bloqueador de ruidos

O

object -objeto
object oriented -orientado por (a) objetos
octet -octeto, byte
ocurrence -aparición
ocurrences -casos
off topic -fuera de temática, fuera de tema
off-line -desconectado, fuera de línea
offset -offset, desplazamiento
ok -aceptar
on-line -conectado, en línea
open source -código fuente abierto
option -opción
outline -bosquejo
overall -por encima, en general
overflow -desbordamiento
overhead -sobrecarga
overload -sobrecarga
overload (v) -sobrecargar
override (v) -redefinir, reescribir, reemplazar
owner -propietario

P

pager -buscapersonas, paginador, conmutador (tal como se usa en gestores de ventanas)
pan (v) -mover

parameter -parámetro
parse (v) -analizar sintácticamente
partition -partición
passphrase -contraseña
password -contraseña, palabra de paso, palabra clave
patch -parche, modificación
patch (v) -actualizar, parchear, emparchar
patch file -archivo (fichero) de parche
path -camino, trayectoria, ruta
pattern -patrón
peer-to-peer -entre iguales
penalty -penalización
perform (v) -realizar (una acción)
performance -rendimiento, desempeño
period -punto
piggybacking -confirmaciones superpuestas, superposición de confirmaciones
pin -patilla, pata, contacto
pipe -tubo, tubería, filtro
pipe (v) -entubar, redireccionar, derivar, redirigir la salida a
pipeling -redireccionamiento
pitch -tono, altura
pixel -píxel, punto
placer -posicionador
plaintext -texto llano
play -reproducir, tocar (música)
player -1 jugador 2 reproductor (de discos compactos), intérprete (de archivos de sonido)
playlist -lista de reproducción
plotter -trazador, graficador
plug and play -enchufar y usar
plug and play (v) -conectado y listo
plug and pray -conecta y reza (para que funcione; véase plug and play)
plug-in -accesorio, añadido, módulo
pluggable -conectable
policy -política, normas, reglas, normativa, directrices, criterios
poligonal mesh -malla de polígonos
poll -sondeo
poll (v) -sondear
polling -sondeo
popup menu -menú emergente
port -1 puerto, puerta (referido al protocolo TCP/IP) 2 migración, porteo (versión de un programa para otra plataforma)
port (v) -portear, portar, adaptar (hacer una versión de un programa para otra plataforma)
portable -portátil
portage -porteo
post -envío
post (v) -remitir, publicar (en un grupo de noticias)
poster -autor (de un artículo o mensaje)
posting agent -agente de envío
postmaster -administrador postal, administrador de correo, postmaster
postponed -pendiente
preemptible -apropiable
preemptive -apropiativo, expropiativo
preview -vista previa, visualización previa
private -privado, confidencial
profile -perfil
profile (v) -perfilar
profiler -perfilador
profiling -parametrización, personalización, perfilado, acción de medir el rendimiento de un programa, personalización (igual que customización), Customización no existe en el VCT ni en el Dic de la RAE Además suena horrible
profiling execution -perfil de uso de recursos (del programa ejecutado)
programmer -programador
programming -programación
prompt -cursor, símbolo de espera de órdenes, punto indicativo
prompt (v) -apremiar

properly -apropiadamente
proprietary software -software de propietario, software en propiedad
provide (v) -proporcionar, proveer, abastecer, habilitar
proxy -proxy, representante, apoderado
punch-in -grabación mediante el método de disparo
purge (v) -purgar, limpiar

Q

query -consulta, pregunta, petición
queue -cola
quit (v) -renunciar, abandonar, finalizar, acabar
quote -1 comilla 2 cita (de un libro, por ejemplo)
quote (v) -citar (referir textualmente)
quoted text -texto citado

R

race condition -condición de carrera
radio button -botón de radio, botón de opción (botón dentro de un grupo en que sólo uno puede estar pulsado a la vez)
radiosity -radiosidad
random -aleatorio
randomizer -generador de aleatoriedad, selector aleatorio, aleatorizador
range -margen, alcance, gama, surtido, línea, intervalo, variedad
rank -rango
rate -tasa
rate (v) -calificar, clasificar
rating -calificación, clasificación
raw -crudo, virgen
raw mode -modo primitivo, modo directo, modo sin formato
ray-tracing -trazado de rayos
re-spawn (v) -reiniciar
readme -leame
realm -reino (conjunto de páginas web cubiertas con el mismo par usuario/contraseña)
realtime -en tiempo real, en vivo
reboot (v) -reiniciar, rearrancar
receiver -receptor, destinatario
recipient -destinatario (de una carta, mensaje, etc)
redirect -redirigir
refresh -actualizar
refuse (v) -rehusar, rechazar
regular expression -expresión regular
relay -1 repetidor, conmutador, relevador, relevo, relé 2 reenvío, conmutación
relay host -nodo de reenvío, conmutador
release -lanzamiento, publicación, entrega, versión, revisión
release (v) -lanzar, publicar, sacar
rely on (v) -depender de, confiar en, delegar en
remailer -reexpedidor
remove (v) -remover, retirar, quitar, sacar (la traducción remover desagrada a algunos, pero otras alternativas que proponen como “borrar” o “desechar” pueden causar confusión; por ejemplo “remove the disk” no debe ser traducido como “borre el disco”)
rendering -síntesis de imágenes, renderizado, representación
reply (v) -responder (al autor de un artículo o mensaje)
repository -repositorio
request -pedido
require (v) -necesitar, exigir
requirement -requisito
reset -reinicio
reset (v) -reiniciar
reset button -botón de reinicio
resolver -sistema de resolución, traductor de direcciones, resolutor
ripper -extractor de audio
root -superusuario, root
root exploit -explotación de root

router -encaminador, enrutador
routing encaminamiento, enrutamiento
routing table -tabla de rutas
run -ejecución
run (v) -ejecutar, correr
run out of memory -agotar la memoria
run time -tiempo de ejecución
runtime library -biblioteca de ejecución

S

sample rate -frecuencia de muestreo
scalable -redimensionable
scanner -escáner, digitalizador
scanning -barrido, rastreo
schedule -horario
schedule (v) -planificar, programar
scheduler -planificador, planificador de tareas
scratch (from) -de cero, desde el principio
screen -pantalla
screen saver -salvapantallas, protector de pantallas
screenshot -captura de pantalla
script -guión, macro, script, archivo de comandos
scroll -desplazamiento, lista, rollo
scroll (v) -desplazar
scroll down (v) -avanzar
sroll up (v) -retroceder
scrollable -deslizable
search -búsqueda
search (v) -buscar
search engine -buscador
search wrapped -búsqueda reiniciada desde el comienzo
Secure Socket Layer (SSL) -capa de conexión segura
seek (v) -buscar
segmentation fault -violación de segmento
semicolon -punto y coma (;)
sender -remitente, Remitente (de una carta, e-mail, etc)
sequence -secuencia, sucesión
sequencer -secuenciador (hardware o software destinado a grabar y reproducir música electrónica en tiempo real usando MIDI, con edición simple de las notas)
server -servidor (de correo, noticias, HTTP, etc)
set -conjunto
set (v) -1 colocar 2 definir 3 ajustar 4 fijar
set up -configuración
set up (v) -configurar
setting -configuración
setup (v) -configurar
shadow passwords -contraseñas ocultas
shared memory -memoria compartida
sharpen (v) -1 afilar 2 mejorar la imagen (hacerla más nítida)
shell -shell (femenino), capa, intérprete de comandos
shell script -archivo (fichero) de comandos, script de shell
shift -desplazamiento
shift (v) -levantar, desplazar
shortcut -atajo
shorthand -abreviado, taquigrafía
shrink (v) -reducir
shutdown -apagar, cerrar
signature -1 firma 2 identificación
silently -sin aviso, discretamente, silenciosamente
Simple Mail Transfer Protocol (SMTP) -protocolo simple de transferencia de correo
site -sitio, local, instalación, sede, recinto, conjunto de paginas relacionads entre si por ejemplo esmascom
skin -carátula

skip (v) -omitir
slash -barra
slot -1 ranura 2 posición
snap (v) -agregar
snapping -agregado
snapshot -1 captura de imagen, captura de pantalla, pantallazo 2 imagen instantánea
sniffer -rastreador, escrutador
snippet -recorte, retazo
splashscreen -pantalla de presentación
socket -socket, enchufe, zócalo, conexión
soft link -enlace lógico, enlace flexible
software -software, soporte lógico, lógica, aplicación, programa
sort (v) -ordenar, clasificar
sort of -tipo de, clase de, más o menos
sound effect -efecto sonoro
source -1 origen 2 código fuente
source code -código fuente
spawn (v) -iniciar
specification -especificación
specs -especificaciones
specular highlights -reflexiones especulares
spell -hechizo
spell (v) -deletrear
spelling -ortografía
spike -pico (en una gráfica)
spin lock -cerrojo, spin lock
splitter -divisor
sponsor (v) -patrocinar
spoof (v) -engañar, falsificar
spool -cola, lista de espera, cola de impresión
spool directory -directorio de la cola
spreadsheet -hoja de cálculo
stack -pila
standard -estándar, patrón, norma
stat (v) -verificar
stats -estadísticas
statement -declaración, cláusula
stochastic -estocástico
store -almacen, depósito
stream -corriente, flujo, secuencia (vídeo)
stream (v) -optimizar
stride -espaciamiento (entre elementos consecutivos de un vector)
string -cadena de caracteres
strip (v) -despojar, desnudar (eliminar los símbolos de depuración en un programa o biblioteca)
stroke -1 golpe 2 ataque (he died of a stroke) 3 movimiento 4 trazo
stroke (v) -trazar
submit -remitir, enviar
subject -asunto
subnet -subrred
subscript -subíndice
supersede (v) -sustituir, modificar
support -soporte, apoyo, respaldo, asesoría
support (v) -apoyar, ayudar, colaborar
surfer -navegante
surround sound -sonido envolvente
swap -intercambio
swap (v) -intercambiar
switch -interruptor, conmutador, switch
symbolic link -enlace simbólico
symlink -enlace simbólico
syntax highlighting -resaltado de sintaxis
system call -llamada al sistema

T

tab -pestaña, lengüeta, tira, tabulador, ficha
tag -marca, coletilla, etiqueta
target -destino, objetivo
target partition -partición de destino
task -tarea
template -plantilla
test -prueba, test
test (v) -evaluar, probar
texture mapping -aplicación de texturas
thread -hilo (hilo de mensajes en una lista, o hilo de ejecución en un programa), hebra
threshold -umbral
threshold level -valor umbral
throughput -flujo, caudal de datos, rendimiento total, productividad
thumbnail -miniatura
ticket -tiquete
tile -baldosa
tile (v) -embaldosar
timeout -timeout, expiración de plazo, tiempo de espera agotado
timer -temporizador
timslice -porción de tiempo, partición de tiempo
timestamp -marca de tiempo, fecha y hora
tiny -diminuto
tip -consejo, sugerencia
toggle -conmutado, biestable
toggle (v) -alternar (entre dos estados)
token -símbolo, lexema
token ring -anillo de fichas
toolbar -barra de herramientas
toolkit -juego de herramientas, conjunto de herramientas
trace -traza
trace (v) -trazar, rastrear
trade off -contrapeso, equilibrio, balance
trade off (v) -contrapesar
trailing spaces -espacios finales
transactional integrity -
integridad transaccional
transport -transporte
transport (v) -transportar
tree view -lista jerárquica
trigger -disparador
troll -trole, metepatas, bocazas
troll (v) -meter la pata, reventar un debate
troubleshooting -eliminación de problemas, solución de problemas
trusted -confiable
tune (v) -afinar
tweak -arreglo
tweak (v) -afinar
twisted pair -par trenzado
type -tipo
type (v) -teclear
typing -impresión (en papel, por ejemplo)
typo -errata

U

undefined -indefinido
underflow -desbordamiento por abajo
Uniform Resource Locator (URL) -localizador
unindent -desangrar (?)
Uninterruptible Power Supply (UPS) -sistema de alimentación ininterrumpida
Universal Asynchronous Receiver and Transmiter (UART) -receptor/transmisor asíncrono universal

up -operacional, en funcionamiento
update -actualización
update (v) -actualizar
upgrade -mejora, versión mejorada
upgrade (v) -promover, mejorar, instalar una versión mejorada
upload -subir, cargar (copiar en un servidor remoto)
upstream -flujo ascendente
upstream port -puerto de envío
upstream version -versión original
user -usuario
user friendly -fácil de usar

V

validity -validez
value -valor
variation -variación, variante
verbatim -literal, textual, al pie de la letra
verbose -prolijo, pormenorizado, detallado, verboso
vertex blending -combinación de vértices
view layout -vista de disposición
viewer -visor

W

wallpaper -fondo, mural, papel tapiz, fondo de pantalla, fondo de escritorio, imágen del fondo, telón de fondo
warning -advertencia, aviso
web -1 web (femenino: “búscalo en la web”, “se encuentra en muchos sitios web”) 2 red, trama
Web Mail Folder (WMF) -carpetas de correo web
webcam -cámara de videoconferencia
weblog -portal de noticias
widget -widget, control, componente
wildcard -comodín
window manager -gestor de ventanas
wireless -inalámbrico
word wrap -ajuste de línea, encapsulamiento de palabra, retorno automático de palabras
wrap (v) -encapsular, forrar, envolver
wraparound -envoltura, envolvente
wrapper -envoltura, forro, empacador, envoltorio

Y

yank (v) -insertar un trozo de texto en la posición actual del cursor

Z

zoom in (v) -acercar
zoom-out (v) -alejar

Archivos de Ejemplo....

memo1

Memo : Mazo 25, 2003

A todos los clase B empleados,

Favor tome nota de los siguientes cambios en
la política de la compañía y beneficios:

1) Usted no podrá estacionar en el patio B;
pero, si podrá el C.
Transporte gratis se efectuara desde el patio a
la puerta de la empres todos los días.

2) Un nuevo PPO ha sido agregado al plan
de salud. Vea a su oficial de recursos humanos
para más información

La cafetería estará cerrada para remodelar
desde Abril 1 a Abril 15.

Gracias a Todos
Roberto Williams
CEO y Presidente

memo2

Memo : Mazo 25, 2003

A todos los clase B empleados,

Favor tome nota de los siguientes cambios en
la política de la compañía y beneficios:

1) Usted no podrá estacionar en el patio A;
pero, si podrá el C.
Transporte gratis se efectuara desde el patio a
la puerta de la empres todos los días.

2) Un nuevo PPO ha sido agregado al plan
de salud. Vea a su oficial de recursos humanos
para más información

La cafetería estará cerrada para remodelar

desde Abril 1 a Abril 15.

4) Abril 3 es día de festivo.

Gracias a Todos
Roberto Williams

poema1

carta de Bolivar a fanny
"Querida Prima... Te extraña que piense en ti al borde del sepulcro..? Ha llegado la última aurora; tengo al frente el mar Caribe azul y plata, agitado como mi alma, por grandes tempestades; a mi espalda se alza el macizo gigantesco de la sierra con sus viejos picos coronados de nieve, impoluta como nuestros ensueños de 1805; por sobre mí, el cielo más bello de América, la más bella sinfonía de colores, el más grandioso derroche de luz...

Y tú estás conmigo, porque todos me abandonan... Tú conmigo en los postreros latidos de la vida y en las últimas fulguraciones de la conciencia... Adiós Fanny... Esta carta de signos vacilantes, la escribe la misma mano que estrechó la tuya en las horas del amor, de la esperanza, de la fe; es la letra escritora del Decreto de Trujillo y del mensaje al Consejo de Angostura... No la conoces, verdad..? Yo tampoco la reconocería, si la muerte no me señalara con su dedo despiadado, la realidad de este supremo instante...

si yo hubiera muerto sobre un campo de batalla, dando frente al enemigo, te daría mi gloria, la gloria que entreví a tu lado, a los campos de un sol de primavera... Muero despreciable, proscrito, detestado por los mismos que gozaron mis favores; víctima de intenso dolor, preso de infinitas amarguras. Te dejo mis recuerdos, mis tristezas y las lágrimas que no llegaron a verter mis ojos... No es digna de tu grandeza tal ofrenda..? Estuviste en mi alma en el peligro; conmigo presidiste los Consejos de Gobierno; tuyos fueron mis triunfos y tuyos mis reveses; tuyos son también mi último pensamiento y mi pena postrimera... En las noches galantes de la Magdalena, vi desfilar mil veces la góndola de Byron por los canales de Venecia; en ella iban grandes bellezas y grandes hermosuras, pero no ibas tú: porque tú has flotado en mi alma, mostrada por níveas castidades...

A la hora de los grandes desengaños, a la hora de las íntimas congojas, aparece ante mis ojos moribundos, con los hechizos de la juventud y de la fortuna; me miras, y en tus pupilas arde el fuego de los volcanes; me hablas, y en tu voz oigo las dianas inmortales de Junín y Bombona... Adiós Fanny...

Todo ha terminado... Juventud, ilusiones, sonrisas y alegrías se hunden en nada; sólo tú quedas como visión seráfica, señoreando el infinito, dominando la eternidad. Me tocó la misión del relámpago, rasgar un instante la niebla, fulgurar apenas sobre el abismo y tornar a perderse en el vacío...

Adiós..!"…
Simón Bolívar,
6 de diciembre de 1830…

poema2

carta de Bolivar a fanny

"Querida Prima... Te extraña que piense en ti al borde del sepulcro..? Ha llegado la última aurora; tengo al frente el mar Caribe azul y plata, agitado como mi alma, por grandes tempestades; a mi espalda se alza el macizo gigantesco de la sierra con sus viejos picos coronados de nieve, impoluta como nuestros ensueños de 1805; por sobre mí, el cielo más bello de américa, la más bella sinfonía de colores, el más grandioso derroche de luz...

Y tú estás conmigo, porque todos me abandonan... Tú conmigo en los postreros latidos de la vida y en las últimas fulguraciones de la conciencia... Adiós Fanny... Esta carta de signos vacilantes, la escribe la misma mano que estrechó la tuya en las horas del amor, de la esperanza, de la fe; es la letra escritora del Decreto de Trujillo y del mensaje al Consejo de Angostura... No la conoces, verdad..? Yo tampoco la reconocería, si la muerte no me señalara con su dedo despiadado, la realidad de este supremo instante...

Si yo hubiera muerto sobre un campo de batalla, dando frente al enemigo, te daría mi gloria, la gloria que entreví a tu lado, a los campos de un sol de primavera... Muero despreciable, proscrito, detestado por los mismos que gozaron mis favores; víctima de intenso dolor, preso de infinitas amarguras. Te dejo mis recuerdos, mis tristezas y las lágrimas que no llegaron a verter mis ojos... No es digna de tu grandeza tal ofrenda..? Estuviste en mi alma en el peligro; conmigo presidiste los Consejos de Gobierno; tuyos fueron mis triunfos y tuyos mis reveses; tuyos son también mi último pensamiento y mi pena postrimera... En las noches galantes de la Magdalena, vi desfilar mil veces la góndola de Byron por los canales de Venecia; en ella iban grandes bellezas y grandes hermosuras, pero no ibas tú: porque tú has flotado en mi alma, mostrada por níveas castidades...

A la hora de los grandes desengaños, a la hora de las íntimas congojas, aparece ante mis ojos moribundos, con los hechizos de la juventud y de la fortuna; me miras, y en tus pupilas arde el fuego de los volcanes; me hablas, y en tu voz oigo las dianas inmortales de Junín y Bombona... Adiós Fanny...

Todo ha terminado... Juventud, ilusiones, sonrisas y alegrías se hunden en nada; sólo tú quedas como visión seráfica, señoreando el infinito, dominando la eternidad. Me tocó la misión del relámpago, rasgar un instante la niebla, fulgurar apenas sobre el abismo y tornar a perderse en el vacío...

Adiós..!"…
Simón Bolívar,
6 de diciembre de 1830…