EL SHELL: COMANDOS BÁSICOS DE *LINUX*

Cualquier usuario de ***Linux*** acabará antes o después relacionándose con el sistema empleando el modo texto, y no los entornos gráficos. Este modo se basa en la ejecución de una serie de comandos, que son interpretados por un programa o ***shell***. ***Linux*** dispone de varios de estos programas pero el más habitual es conocido como ***bash*** o ***Bourne Shell***. Si ***Linux*** se ha arrancado en modo texto el sistema arranca de forma directa el ***shell*** y queda a la espera de introducción de nuevos comandos. Si se ha arrancado en modo gráfico se puede acceder al ***shell*** de dos formas:

- Se puede acceder al ***shell*** del sistema presionando alguna de las siguientes combinaciones de teclas:
  - ***<ctrl>+<alt>+<F1>***
  - ***<ctrl>+<alt>+<F2>***
  - ***<ctrl>+<alt>+<F3>***
  - ***<ctrl>+<alt>+<F4>***
  - ***<ctrl>+<alt>+<F5>***
  - ***<ctrl>+<alt>+<F6>***

Esto hace que el sistema salga del modo gráfico y acceda a alguna de las seis consolas virtuales de ***Linu***x, a las cuales también se puede acceder cuando se arranca en modo de texto. Para volver al modo gráfico hay que presionar

- ***<ctrl>+<alt>+<F7>*** o
- ***<ctrl>+<alt>+<F8***>

- La segunda forma es más cómoda y menos radical permitiendo acceder al shell desde el mismo entorno gráfico. Para esto hay que abrir un programa llamado terminal o consola, por ejemplo: ***kconsole*** (en el entorno ***KDE), xterm, gnome-terminal*** (en ***GNOME***), etc.

Introducción

Existen una serie de nociones básicas que hay que tener en cuenta a la hora de introducir los comandos. En primer lugar citaremos las siguientes:

- Los comandos hay que teclearlos exactamente.
- Las letras mayúsculas y minúsculas se consideran como diferentes.
- En su forma más habitual (los ***shells*** de Bourne o de Korn), el sistema operativo utiliza un signo de ***$*** como ***prompt*** para indicar que está preparado para aceptar comandos, aunque este carácter puede ser fácilmente sustituido por otro u otros elegidos por el usuario. En el caso de que el usuario acceda como administrador este signo se sustituye por **#.**

- Cuando sea necesario introducir el nombre de un fichero o directorio como argumento a un comando, ***Linu***x, permite escribir las primeras letras del mismo y realiza un autorrellenado al presionar la tecla del ***tabulado***r. Si no puede distinguir entre diversos casos rellenará hasta el punto en el que se diferencien. Por ejemplo, supongamos una carpeta con los siguientes directorios:

  Programas

  Documentos_proyecto

  Documentos_privados

  Al escribir ***cd Pr<tab> Linux*** rellenará el resto del contenido hasta escribir ***cd Programa***s. Por el contrario al escribir ***cd D<tab>*** escribirá ***cd Documentos***

Sintaxis de los comandos

Los comandos tienen la siguiente sintaxis:

```
# programa arg1 arg2 ... argn
```

Se observa que, en la ``línea de comandos'', se introduce el programa seguido de uno o varios argumentos. Así, el intérprete ejecutará el programa con las opciones que se hayan escrito.

Cuando se quiere que el comando sea de varias líneas, se separa cada línea con el carácter barra invertida (\). Además, cuando se quiere ejecutar varios comandos en la misma línea, los separa con punto y coma (;). Por ejemplo:

```
# make modules ; make modules_install
```

En los comandos, también se puede utilizar los comodines:

- El asterisco (*) es equivalente a uno o más caracteres en el nombre de un archivo. Ejm: ls *.c lista todos los archivos con extensión c.
- El signo de interrogación (?) es equivalente a un único carácter. Ej.: ls curso.te? lista el archivo curso.tex completando el último carácter.
- Un conjunto de caracteres entre corchetes es equivalente a cualquier carácter del conjunto. Ej.: ls curso_linux.t[aeiou]x lista curso_linux.tex seleccionando la e del conjunto. .

Alias

Un ``alias'' es un nombre alternativo para un comando. Así, en lugar de escribir el comando propiamente dicho, escribiríamos el alias de dicho comando.

Un alias se puede definir por varios motivos, por ejemplo:

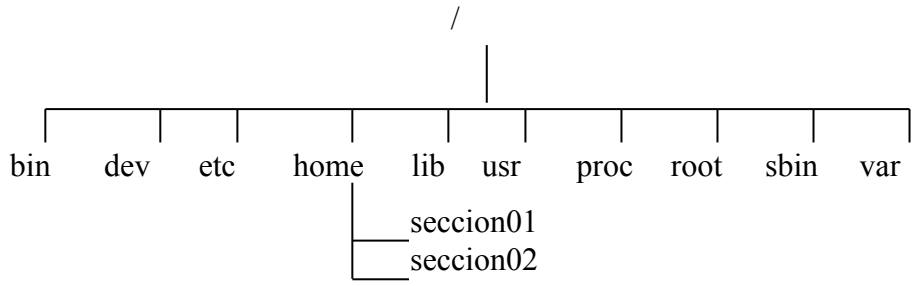
- Dar nombres familiares a comandos comunes:
  alias md='mkdir'
  Crearía un alias para el comando mkdir, similar al de DOS.

- Dar nombres a comandos largos:
  alias tbz2='tar -cv --use-compress-program=bzip2 -f'
  Crearía un alias para el comando tar para que use el compresor bzip2 en lugar de gzip.

Para no tener que escribir todos los alias siempre que entremos al sistema, escribiríamos dicho alias en el archivo /.bash_profile

Directorios en Linux:

Bajo Linux el espacio de archivos que es visible a los usuarios está basado en una estructura de árbol, con la raíz en la parte superior (/). El directorio más alto, /, es conocido como directorio raíz.



Por lo general, la mayor parte del sistema operativo reside en dos sistemas de archivos: el sistema de archivos raíz, conocido como /, y el sistema de archivos montado bajo /usr

El directorio bin contiene programas ejecutables conocidos como binarios. Estos programas son archivos de sistema esenciales. Muchos de los comandos como ls, son en realidad programas que se encuentran en este directorio.

El directorio sbin también se utiliza para guardar archivos binarios de sistema. La mayoría de los archivos de este directorio se usa para propósitos de la administración del sistema.

El directorio **etc.** contiene muchos de los archivos de configuración del sistema Linux. Son los archivos que personalizan Linux. P. ej. el archivo de contraseñas, passwd, se encuentra aquí.

Las bibliotecas compartidas que emplean los programas cuando se ejecutan están guardadas en el directorio /lib. Mediante el uso de estas bibliotecas compartidas, muchos programas pueden utilizar el mismo código, y estas bibliotecas se pueden guardar en un lugar común, reduciendo el tamaño de los programas al momento de su ejecución.

El directorio /dev contiene archivos especiales conocidos como archivos de dispositivo, los cuales se usan para tener acceso a todo tipo de hardware del sistema. Por ejemplo, el archivo /dev/mouse se utiliza para la lectura de entrada del mouse.

/proc es un sistema de archivos virtual. Se usa para leer de la memoria información de procesos.

El directorio /home es el directorio base para los directorios personales de los usuarios.

El directorio /var guarda los archivos que tienden a cambiar de tamaño a lo largo del tiempo. Por lo general, los diversos archivos de bitácora de sistema se encuentran bajo este directorio. El directorio /var/spool y sus subdirectorios se utilizan para guardar datos que son de manera transitiva, como el correo y las noticias recién recibidas o puestas en cola para transmisión a otro sitio.

El directorio /usr y sus subdirectorios contienen algunos de los programas más importantes del sistema. Por lo general, los subdirectorios de /usr contienen los grandes paquetes de software que se instalan.

/usr/bin	Se guardan ahí muchos de los programas ejecutables de Linux
/usr/etc	Archivos de configuración misceláneos
/usr/include	Archivos de inclusión para el compilador de C.
/usr/lib	Bibliotecas para los programas durante el enlace
/usr/man	Páginas del manual de Linux
/usr/src	Ahí se almacena el código fuente de diferentes programas del sistema
/usr/local	Está diseñado para la personalización local del sistema. Por lo general, gran parte del software local se instala en los subdirectorios de este directorio.

Algunos Comandos Sencillos de *LINUX*

Passwd

Para efectuar el cambio o la introducción de un password o contraseña o agregar alguna a una cuenta que no la tenga. Primero se especifican la nueva y antigua contraseña, y luego se verifica la nueva.

El proceso a seguir es el siguiente:

(current) UNIX password: (se teclea la contraseña actual; no aparece en pantalla)

New UNIX password: (se teclea la nueva contraseña; no aparece en pantalla)

Retype new UNIX password: (se teclea de nuevo la nueva contraseña comprobando que se ha tecleado bien. Si no coincide no se cambia produce el cambio).

Changing password for <user>
Enter old password : password
Enter new password:password
Re-type new password:password

date Muestra por pantalla el día y la hora.

cal *1949* Muestra el calendario del año 1949.

cal ***05 1949*** Muestra el calendario de mayo de 1949.

who Indica qué usuarios tiene el ordenador en ese momento, en qué terminal están y desde qué hora.

whoami Indica cuál es la terminal y la sesión en la que se está trabajando.

man comando

Todos los manuales de ***Linux*** están dentro del propio sistema operativo, y este comando permite acceder a la información correspondiente al comando ***comando***. Por ejemplo con ***man who*** aparecerá por pantalla y de forma formateada por páginas, la explicación del comando ***who***. Se puede navegar a través de estas páginas con los cursores del teclado, y presionando **q** para salir.

man ls: muestra la información sobre el comando de visualización de la pantalla.

man dir: muestra la información sobre el comando que contiene los directorios.

man mkdir: muestra la información sobre la creación de archivos.

man rmdir: muestra la información sobre borrar ficheros.

man pico: muestra la información sobre el uso de este editor de texto.

clear

Este comando limpia la consola.

Directorio Personal

Como se ha visto anteriormente el directorio personal es un directorio con un determinado nombre asignado a un usuario. Los directorios personales habitualmente son subdirectorios de ***/home*** (en algunos casos se utiliza ***mn***t, u otro subdirectorio de orden inferior). Generalmente el nombre coincide con el del ***nombre de usuari***o, aunque puede no ser así, y varios usuarios pueden estar trabajando en el mismo directorio. Cada usuario de ***Linux*** puede crear una estructura en árbol de subdirectorios y archivos tan compleja como desee bajo su directorio personal pero normalmente nunca fuera de él.

Listado del contenido de directorios: comando ls

Una de las acciones más habituales a la hora de trabajar es mostrar el contenido de un directorio, como se ha visto existen herramientas gráficas con este fin, no obstante el ***shell*** incluye un programa con este mismo fin: ***l***s, ***ls*** Muestra los nombres de los ficheros y subdirectorios contenidos en el directorio en el que se está. Sólo se obtienen los nombres de los ficheros, sin ninguna otra información.

ls -a Muestra todos los ficheros incluyendo algunos que ordinariamente están ocultos para el usuario (aquellos que comienzan por un punto). Se recuerda que el fichero punto . indica el directorio actual y el doble punto .. el directorio padre, que contiene, al actual.

ls -l Esta es la opción de lista larga: muestra toda la información de cada fichero incluyendo: protecciones, tamaño y fecha de creación o del último cambio introducido,...

ls -c Muestra ordenando por día y hora de creación.

ls -t Muestra ordenando por día y hora de modificación.

ls -r Muestra el directorio y lo ordena en orden inverso.

ls subdir Muestra el contenido del subdirectorio ***subdi***r.

ls -l filename Muestra toda la información sobre el fichero.

ls --color Muestra el contenido del directorio coloreado.

Las opciones anteriores pueden combinarse. Por ejemplo:

ls -cr Muestra el directorio ordenando inversamente por fechas.

El comando ***ls*** admite los caracteres de sustitución o metacarácteres ***(****) y ***(***?). El carácter ******* representa cualquier conjunto o secuencia de caracteres. El carácter ***?*** representa cualquier carácter, pero sólo uno. Por ejemplo:

ls *.gif Muestra todos los nombres de ficheros que acaben en ***.gi***f, por ejemplo ***dib1.gif, a.gi***f, etc..

ls file? Muestra todos los ficheros cuyos nombres empiecen por ***file*** y tengan un nombre de cinco caracteres, por ejemplo: ***file1, file2, file***a, etc.

Creación de subdirectorios. Comando mkdir

El comando ***mkdir (make directo***r**y**) permite a cada usuario crear un nuevo subdirectorio:

mkdir subdir1

donde ***subdir*** es el nombre del directorio que se va a crear.

mkdir -p, --parents: crea los directorios padre que faltan para cada argumento.

mkdir --verbose: muestra un mensaje para cada directorio creado.

mkdir -help: muestra un mensaje de salida estándar de ayuda

Borrado de subdirectorios. Comando rmdir

Este comando borra uno o más directorios del sistema ***(remove directo***r**y**), siempre que estos subdirectorios estén **vacío**s. Por ejemplo:

rmdir subdir1

donde ***subdir*** es el nombre del directorio que se va a eliminar.

rmdir -p: borra el archivo aunque tenga mas componentes en el.

rmdir --help: muestra un archivo de ayuda.

rm --versión: muestra el tipo de versión con el que esta trabajando la máquina.

Cambio de directorio. Comando cd

Este comando permite cambiar de directorio a partir del directorio actual de trabajo. Por ejemplo,

cd /home/Pedro En este ejemplo pasamos del directorio actual de trabajo al nuevo directorio

/home/Pedro, que será desde ahora nuestro nuevo directorio.

cd dire Nos traslada al subdirectorio ***dire*** (que deberá existir como subdirectorio en el directorio actual).

cd .. Retrocedemos un nivel en la jerarquía de directorios. Por ejemplo, si estamos en ***/home/Pedro*** y usamos este comando, pasaremos al escalafón inmediatamente superior de la jerarquía de directorios, en este caso a ***/home***.

Nota: al contrario que en ***MS-DOS*** en ***Linux*** no existe la forma **cd..** sin espacio entre cd y los dos puntos.

cd Nos sitúa nuevamente en el directorio personal del usuario.

Situación actual. Comando pwd

El comando ***pwd (print working director***y) visualiza o imprime la ruta del directorio en el que nos encontramos en este momento. Este comando es uno de los pocos que no tiene opciones y se utiliza escribiendo simplemente ***pwd***.

Acceso a unidades de disco

Linux a diferencia de ***Windows*** no utiliza letras ("a:", "c:", "d:", ...) para acceder a las distintas unidades de disco de un ordenador. En ***Linux*** para acceder al contenido de una unidad de disco o de un CD-ROM este tiene que haber sido previamente ***"montado***". El ***montado*** se realiza mediante el comando **mount**, con lo que el contenido de la unidad se pone a disposición del usuario en el directorio de ***Linux*** que se elija. Por ejemplo para acceder al CD-ROM se teclearía el siguiente comando:

mount -t iso9660 /dev/cdrom /mnt/cdrom.

donde ***-t iso9660*** indica el tipo de sistema que usa la unidad de disco para guardar los ficheros (las más usuales son: ***iso9660*** en el caso de un CD-ROM, ***vfat*** en el caso de ***Window***s, y ***ext2*** en el caso de ***Linu***x), ***/dev/cdrom*** indica el dispositivo que se va a montar. Todos los dispositivos están representados por un fichero del directorio ***/de***v, por ejemplo en el caso de un disquete será seguramente ***/dev/fd0***, por último ***/mnt/cdrom*** es el directorio en el que se pondrá a disposición del usuario el contenido del CD-ROM. Para montar disquetes se suele utilizar el directorio ***/mnt/floppy***.

De todas formas el usuario siempre puede crear un directorio vacío con el nombre que el elija para montar las unidades de disco que desee donde desee. Cuando el usuario

haya dejado de usar ese disco deberá ***"desmontarlo***" mediante el comando ***umount*** antes de sacar el disquete o el CD-ROM. En este último caso debería escribir:

umount /mnt/cdrom

Para utilizar el comando ***mount*** de la forma anterior hace falta ser administrador o ***root***. Para que un usuario común pueda utilizar disquetes, CD-ROM, etc. hay que editar el fichero ***/etc/fstab*** Por ejemplo para que cualquier usuario pueda acceder a un disquete habrá que indicar la siguiente línea:

/dev/fd0 /mnt/floppy vfat user,noauto 0 0

También habrá que asegurarse de que el directorio ***/mnt/floppy*** sea accesible por todos los usuarios.

Una vez seguidos los pasos anteriores cualquier usuario podrá "montar" un disquete escribiendo el siguiente comando:

mount /mnt/floppy

Al igual que antes el usuario deberá ejecutar el comando ***umount /mnt/floppy*** antes de sacar el disquete.

Nota: Existen en la actualidad distribuciones (p. ej. ***Linux Mandrak***e) que realizan este proceso de forma automática por lo que las unidades de disquete y CD-ROM quedan accesibles a todos los usuarios de una forma sencilla, empleando los comandos:

mount /mnt/floppy

umount /mnt/floppy

siempre que ***/mnt/floppy*** sea la ruta adecuada.

Copia de ficheros. Comando cp

Este comando tiene la siguiente forma,

cp file1 file2

y hace una copia de ***file1*** y le llama ***file***2. Si ***file2*** no existía, lo crea con los mismos atributos de ***file***1. Si ***file2*** existía antes, su contenido queda destruido y es sustituido por el de ***file***1. El fichero ***file2*** estará en el mismo directorio que ***file***1. Tanto file1 como file2 indican el nombre de un archivo, que puede incluir el la ruta al mismo si alguno de ellos no se encuentra en el directorio actual. Otra posibilidad es:

cp file1 file2 namedir

que hace copias de ***file1*** y ***file2*** en el directorio ***namedir***.

cp -f: borra los ficheros ya existentes en un destino dado.

cp -i: pregunta si deseas sobreescribir ficheros.

cp -p: preserva los permisos, el propietario y el grupo de los ficheros.

cp -r: copia directorios recursivamente y hace lo correcto cuando se encuentran objetos distintos de ficheros.

Traslado y cambio de nombre de ficheros. Comando mv

Este comando tiene una forma similar al anterior,

mv file1 file2

El comando ***mv*** realiza la misma función que el anterior ***(c***p) pero además destruye el fichero original. En definitiva traslada el contenido de ***file1*** a ***file***2; a efectos del usuario lo que ha hecho es cambiar el nombre a ***file***1, llamándole ***file***2. De igual forma,

mv file1 file2 namedir

traslada uno o más ficheros ***(file***1, ***file***2,...) al directorio ***namedir*** conservándoles el nombre. El comando,

mv namedir1 namedir2

cambia el nombre del subdirectorio ***namedir1*** por ***namedir2***.

mv -i: con esta variación al mover o renombrar algún fichero el comando pide una confirmación si ya hay un fichero destino.

mv -f: este comando no pide confirmación.

mv -f, --force:borra todos los ficheros de destino existentes, sin preguntar al usuario.

mv -v, --verbose: con este comando se borra el nombre de cada fichero antes de moverlo.

mv -u, --pdate: no mueve un fichero que tenga un destino con el mismo tiempo de modificación.

Hay que recalcar que el comando ***mv*** sirve así mismo para cambiar el nombre de los ficheros.

Enlaces a ficheros. Comando ln

En ***Linux*** un mismo fichero puede estar repetido con más de un nombre, ya que con el comando ***cp*** se pueden realizar cuantas copias se desee del fichero. Pero, a veces, es más práctico tener un mismo fichero con varios nombres distintos, y lo que es más importante, poder acceder a él desde más de un directorio. En ***Linux*** esto recibe el nombre de enlaces múltiples a un fichero. El ahorro de espacio de disco es importante al poder compartir un fichero más de un usuario. Estos enlaces son muy prácticos a la hora de utilizar ficheros que pertenecen a directorios distintos. Gracias a los enlaces se puede acceder a muchos ficheros desde un mismo directorio, sin necesidad de copiar en ese directorio todos esos ficheros. La forma de este comando es,

ln file1 file2

A partir de este momento el fichero ***file1*** tiene dos nombres: ***file1*** y ***file***2. A diferencia de los comandos ***cp*** y ***m***v, este comando toma más precauciones, ya que advierte previamente si el nombre ***file2*** está ocupado, y en este caso no se ejecuta.

ln panacea subdir/panacea

Después de este comando el fichero ***panacea*** tendrá el mismo nombre, pero a efectos del usuario estará colocado en dos sitios distintos: en el directorio actual y en el subdirectorio ***subdi***r. Los ficheros enlazados a otro se borran como los ficheros normales. Si se borra el fichero original permanece su contenido en los ficheros enganchados.

Borrado de ficheros. Comando rm

Este comando tiene las formas siguientes,

rm file1 file2

Este comando elimina uno o más ficheros de un directorio en el cual tengamos permiso de escritura. Con este comando resulta facilísimo borrar ficheros inútiles, y desgraciadamente, también los útiles. Por eso es conveniente y casi imprescindible emplear lo opción **-**i, de la forma siguiente:

rm -i file1 file2

Con esta opción, ***Linux*** pedirá confirmación para borrar cada fichero de la lista, de si realmente se desea su destrucción o no. Se recomienda usar siempre este comando con esta opción para evitar el borrado de ficheros útiles. Por ejemplo, si se teclea,

rm -i superfluo

aparecerá en pantalla el aviso siguiente:

remove superfluo?

y habrá que contestar ***y*** (yes) o ***n*** (not). En este comando se pueden utilizar los caracteres de sustitución ***(****** y ?), como por ejemplo,

rm fich*

que borraría todos los ficheros del directorio actual que comiencen por ***fic***h. El comando

rm *

borrará todos los ficheros del directorio actual, mientras que

rm -i *

realiza una labor análoga, pero con previa confirmación.

rm -f: no pide la configuración.

rm -i: pide la configuración.

rm -r: borra los árboles de directorios.

rm -v, --vorse: muestra el nombre del fichero antes de borrarse.

rm --versión: muestra en la salida estándar información sobre la versión.

Características de un fichero. Comando file

Este comando realiza una serie de comprobaciones en un fichero para tratar de clasificarlo. Su formato es:

file fich

Tras su ejecución este comando muestra el tipo del fichero e información al respecto del mismo.

Cambio de modo de los ficheros comandos chmod, chown y chgrp

Los permisos de cada fichero se pueden ver con el comando ***ls -***l. Para cambiar los permisos de un fichero se emplea el comando ***chmo***d, que tiene el formato siguiente:

chmod [quien] oper permiso files

quien Indica a quien afecta el permiso que se desea cambiar. Es una combinación cualquiera de las letras ***u*** para el usuario, ***g*** para el grupo del usuario, ***o*** para los otros usuarios, y ***a*** para todos los anteriores. Si no se da el ***quie***n, el sistema supone a.

oper Indica la operación que se desea hacer con el permiso. Para dar un permiso se pondrá un +, y para quitarlo se pondrá un -.

permiso Indica el permiso que se quiere dar o quitar. Será una combinación cualquiera de las letras anteriores : ***r,w,x,***s.

files Nombres de los ficheros cuyos modos de acceso se quieren cambiar. Por ejemplo, para quitar el permiso de lectura a los usuarios de un fichero el comando es:

chmod a -r fichero.txt

chmod + hase que los permisos señalados se añadan.

chmod - hase que los permisos seleccionados se retiren.

chmod = hase que los permisos del fichero sean únicos.

chmod -r. cambia permisos de directorios.

chmod -c, --changes: muestra un mensaje antes de cambiar el permiso a un fichero.

Los permisos de lectura, escritura y ejecución tienen un significado diferente cuando se aplican a directorios y no a ficheros normales. En el caso de los directorios el permiso ***r*** significa la posibilidad de ver el contenido del directorio con el comando ***l***s; el permiso ***w*** da la posibilidad de crear y borrar ficheros en ese directorio, y el permiso ***x*** autoriza a buscar y utilizar un fichero concreto.

Por otra parte, el comando ***chown*** se emplea para cambiar de propietario ("***c***hange ***ow***ner") a un determinado conjunto de ficheros. Este comando sólo lo puede emplear el actual propietario de los mismos. Los nombres de propietario que admite ***Linux*** son los

nombres de ***usuari***o, que están almacenados en el fichero ***/etc/passw***d. La forma general del comando ***chown*** es la siguiente:

chown newowner file1 file2 ...

chown -r: cambia propietario de directorios.

chown -c, --changes: muestra los ficheros cuyos propietarios cambian.

chown -f, --silent, -quiet: no muestra mensajes de error sobre ficheros que no puedan cambiarse.

chown -help: muestra un mensaje de ayuda.

Análogamente, el grupo al que pertenece un fichero puede ser cambiado con el comando ***chgr***p, que tiene una forma general similar a la de ***chow***n,

chgrp newgroup file1 file2...

Los grupos de usuarios están almacenados en el fichero ***/etc/grou***p.

Exit

Termina la sesión de Linux en una terminal

More

Exhibe pantallas completas de un archivo de texto. Filtro de fichero para la visualización en terminales.

more -num: especifica el tamaño de la pantalla.

more -p: limpia toda la pantalla.

more -u: suprime el subrayado.

more -l: deshabilita el avance de página.

more -s: reducir múltiples líneas en blanco a una.

mail

Manda y recibe correos electrónicos.

mail -v: cambia al modo verbose.

mail -i: ignora los bloques de interrupción.

mail -i: fuerza al mail correr en un modo interactivo.

mail -c: manda copias a la lista de usuarios.

mail -f: lee el contenido de los archivos especificados.

Pico

Pico es un simple editor de texto.

ctrl -k :borra texto seleccionado

ctrl -h: muestra la opción de comandos.

ctrl -m: muestra el funcionamiento del ratón.

ctrl -c: enable files name completion.

pine

Programa diseñado para enviar o recibir correos electrónicos.

adress: manda correos a una dirección dada.

pine -attach file: manda correos a una lista de archivos.

pine -file: abre archivos de texto.

pine -h: ayuda

Espacio ocupado en el disco Comandos du y df

El comando ***du*** permite conocer el espacio ocupado en el disco por un determinado directorio y todos los subdirectorios que cuelgan de él. Para usarlo basta simplemente colocarse en el directorio adecuado y teclear, ***d***u, éste comando da el espacio de disco utilizado en bloques. Para obtener la información en bytes se debe emplear el comando con la opción –h: ***du -h***

El comando ***df*** por el contrario informa del espacio usado por las particiones del sistema que se encuentren montadas.

Visualización sin formato de un fichero. Comando cat

Este comando permite visualizar el contenido de uno o más ficheros de forma no formateada. También permite copiar uno o más ficheros como apéndice de otro ya existente. Algunas formas de utilizar este comando son las siguientes,

cat filename Saca por pantalla el contenido del fichero ***filenam***e.

cat file1 file2... Saca por pantalla, secuencialmente y según el orden especificado, el contenido de los ficheros indicados.

cat file1 file2 >file3 El contenido de los ficheros ***file1*** y ***file2*** es almacenado en ***file***3.

cat file1 file2 >>file3 El contenido de ***file1*** y ***file2*** es añadido al final de ***file***3.

cat >file1 Acepta lo que se introduce por el teclado y lo almacena en ***file1*** (se crea ***file***1). Para terminar se emplea ***<ctrl>d***

Comando head

head -7 filename escribe las 7 primeras líneas del fichero filename

Visualización de ficheros con formato. Comando pr

Este comando, a diferencia de ***ca***t, imprime por consola el contenido de los ficheros de una manera formateada, por columnas, controlando el tamaño de página y poniendo cabeceras al comienzo de las mismas. Está muy en relación con el comando ***lp*** de salida por impresora. Las formas más importantes que admite son las siguientes:

pr file Produce una salida estándar de 66 líneas por página, con un encabezamiento de 5 líneas (2 en blanco, una de identificación y otras 2 líneas en blanco).

pr -ln file Produce una salida de n líneas por página (cuando el tamaño de papel de impresora, por ejemplo, tiene un número de líneas distinto de 66)

pr -p file Hace una pausa para presentar la página, hasta que se pulsa ***<return>*** para continuar

pr -t file Suprime las 5 líneas del encabezamiento y las del final de página.

pr -wn file Ajusta la anchura de la línea a n posiciones.

pr -d file Lista el fichero con espaciado doble.

pr -h `caracteres` file el argumento o cadena de caracteres ***`caracteres`*** se convertirán en la cabecera del listado.

pr +n file Imprime el fichero a partir de la página n.

Además de los ejemplos anteriores, se pueden combinar varias opciones en un mismo comando, como por ejemplo en: ***pr -dt file*** la salida de este comando es por la consola, pero puede redireccionarse a otro fichero, por ejemplo, si ejecutamos el comando: ***pr file1 > file2*** se crea un fichero nuevo llamado ***file2*** que es idéntico a ***file***1, pero con formato por páginas y columnas.

Visualización de ficheros pantalla a pantalla. Comandos more y less

Estos comandos permiten visualizar un fichero pantalla a pantalla. El número de líneas por pantalla es de 23 líneas de texto y una última línea de mensajes, donde aparecerá la palabra more. Cuando se pulsa la barra espaciadora (el espacio en blanco), se visualizará la siguiente pantalla. Para salir de este comando (terminar la visualización) se pulsa ***<ctrl>d*** o q. Por ejemplo: ***more file***

El comando ***less*** es muy similar al anterior pero permite el desplazamiento a lo largo del texto empleando las teclas de cursores pudiendo desplazarse hacia arriba o abajo de un fichero.

Búsqueda en ficheros. Comandos grep, fgrep y egrep

El comando ***grep*** localiza una palabra, clave o frase en un conjunto de directorios, indicando en cuáles de ellos la ha encontrado. Este comando rastrea fichero por fichero, por turno, imprimiendo aquellas líneas que contienen el conjunto de caracteres buscado. Si el conjunto de caracteres a buscar está compuesto por dos o más palabras separadas por un espacio, se colocará el conjunto de caracteres entre apóstrofes ('). Su formato es el siguiente:

grep 'conjuntocaracteres' file1 file2 file3

siendo 'conjuntocaracteres' la secuencia de caracteres a buscar, y ***file***1, ***file***2, y ***file31*** los ficheros donde se debe buscar. Veamos un nuevo ejemplo:

grep 'TRIANGULARIZACION MATRIZ' matrix.f scaling.f

Este comando buscará ***TRIANGULARIZACION MATRIZ*** entre las líneas de los ficheros ***matrix.f*** y ***scaling.f.*** Este comando permite seleccionar, entre todas las líneas de uno o más ficheros, aquellas que contienen un motivo que satisface una expresión regular determinada.

grep [-opción] expresión_regular [referencia...]

Las opciones principales son:

c lo único que se hace es escribir el número de las líneas que satisfacen la condición.

i no se distinguen mayúsculas y minúsculas.

l se escriben los nombres de los ficheros que contienen líneas buscadas.

n cada línea es precedida por su número en el fichero.

s no se vuelcan los mensajes que indican que un fichero no se puede abrir.

v se muestran sólo las líneas que no satisfacen el criterio de selección.

A continuación se muestra una serie de ejemplos.

• ***grep '^d' text*** líneas que comienzan por d.

• ***grep '^[^d]' text*** líneas que no comienzan por d.

• ***grep -v '^C' file1 > file2*** quita las líneas de file1 que comienzan por C y lo copia en file2.

Comandos tar y gzip

Tanto el comando ***tar*** como ***gzip*** son ampliamente empleados para la difusión de programas y ficheros en ***Linu***x. El primero de ellos agrupa varios ficheros en uno solo o ***"archiv***o", mientras que el segundo los comprime. En conjunto estos dos programas actúan de forma muy similar a programas como ***Winzi***p. Para crear un nuevo archivo se emplea:

tar –cvf nombre_archivo.tar fichero1 fichero2 …

donde fichero1, fichero2 etc. son los ficheros que se van a añadir al archivo ***ta***r. Si se desea extraer los ficheros se emplea

tar –xpvf nombre_archivo.tar fichero1 …

Al contrario que tar que agrupa varios ficheros en uno, ***gzip*** comprime un único fichero con lo que la información se mantiene pero se reduce el tamaño del mismo. El uso de ***gzip*** es muy sencillo

gzip fichero

con lo que se comprime fichero (que es borrado) y se crea un fichero con nombre ***fichero.g***z. Si lo que se desea es descomprimir un fichero se emplea entonces:

gzip –d fichero.gz

recuperando el fichero inicial. Como se ha comentado al principio es típico emplear ***tar*** y ***gzip*** de forma consecutiva, para obtener ficheros con extensión ***tar.gz*** o ***tgz*** que contienen varios ficheros de forma comprimida (similar a un fichero ***zi***p). El comando ***tar*** incluye la opción z para estos ficheros de forma que para extraer los ficheros que contiene:

tar –zxf fichero.tar.gz

Comandos de impresión.

Comando lpr

El comando ***lpr*** se emplea para imprimir una serie de ficheros. Si se emplea sin argumentos imprime el texto que se introduzca a continuación en la impresora por defecto. Por el contrario,

lpr nombre_fichero

imprime en la impresora por defecto el fichero indicado.

REDIRECCIONES Y TUBERÍAS

Redirecciones

Los comandos de ***Linux*** tienen una entrada estándar (número 0) y dos salidas estándar (número 1 para la salida normal del comando, y número 2 para la salida de los mensajes de error que se puedan producir en su ejecución). Por defecto tanto la entrada como las salidas estándar de los comandos son la propia terminal, a no ser que por la propia naturaleza del comando se den en él los nombres de algunos ficheros que hagan el papel de entrada y de salida. Por ejemplo, en el comando

cp file1 file2

file1 es la entrada y ***file2*** es la salida; aquí no intervienen las entradas y salidas estándar. Sin embargo, cuando utilizamos por ejemplo el comando ***ls*** (listado de directorio), la salida de este comando se dirige hacia la terminal. Si queremos que la salida de este comando se dirija a un fichero llamado ***fil***e, podríamos escribir,

ls >file

el ***(***>) es uno de los llamados operadores de redirección y dirige la salida estándar hacia el fichero indicado a continuación; si este fichero no existe, se crea en ese momento. Otros operadores de redirección son el operador ***(***<) que redirige la entrada estándar desde un determinado fichero, y el operador ***(***>>) que redirige la salida estándar hacia otro fichero, pero añadiendo dicha salida al final de ese fichero, sin sobrescribir el contenido original. Por ejemplo, si cada vez que entramos en el sistema ejecutamos el comando,

date >>archivo

tendremos un fichero llamado ***archivo*** que contiene información sobre todas las veces que hemos entrado en el sistema. Otro ejemplo, para añadir al fichero ***file2*** al final de ***file1*** y al conjunto llamarle ***file***3, sería

cat file1 file2 >file3

o, si quisiéramos que el fichero resultante fuera el mismo ***file***1,

cat file2 >>file1

Un ejemplo en redirección a la entrada podría ser el siguiente,

mail juan <carta

que envía al usuario ***juan*** el contenido del fichero ***cart***a.

Tuberías

Siguiendo con los ejemplos anteriores, si quisiéramos enviar a ***juan*** una lista de nuestros ficheros podríamos utilizar los comandos,

ls >fichero

mail juan <fichero

rm fichero

Es decir que hemos conectado la salida estándar de ***ls*** con la entrada estándar de ***mai***l, a través de un fichero transitorio ***filelis***t. ***Linux*** permite hacer esta operación directamente, sin pasar por el fichero de almacenamiento transitorio: esto se hace mediante el concepto de tubería (pipe), que consiste en empalmar la salida estándar de un comando con la entrada estándar de otro. Para el ejemplo anterior esto se hace en la forma,

ls | mail juan

Con el operador de tubería ***(***|) se pueden empalmar tantos comandos como se desee.

Bifurcación o T (comando tee)

A veces interesa que la salida de un comando, además de redirigirse a un determinado fichero, se bifurque también hacia la terminal, con objeto de observar inmediatamente el resultado. Esto se consigue con el operador ***te***e, que podría emplearse de la siguiente forma:

ls | tee file

la salida de ***ls*** se bifurca hacia la terminal y hacia ***fil***e.

Si quisiéramos que la salida de este comando se añadiera al final de ***fil***e, deberíamos utilizar la opción **-**a,

ls | tee -a file

Redirección de la salida de errores

Los mensajes de error se dirigen a la salida número 2, que normalmente es también la terminal. A veces, por ejemplo cuando se quiere ejecutar un comando en background (ejecutar un comando en background es lanzar su ejecución y recuperar el control de la terminal sin esperar a que termine, lo cual se hace añadiendo el carácter ***&*** al final del comando), interesa evitar que los mensajes de error aparezcan en la pantalla, pues en ella habremos empezado a hacer otra cosa. Supongamos por ejemplo que queremos compilar y montar en background un conjunto de ficheros, dirigiendo los listados a un fichero llamado ***listado***s, y los mensajes de error a un fichero llamado ***errore***s. Lo haríamos en la forma,

gcc prueba.c 2>errores

con lo cual la salida 2 (errores) se redirige hacia el fichero ***errore***s. Para redirigir la salida estándar de errores al mismo fichero que la salida estándar se emplea un comando como:

program <datos.d >resultados.r 2>&1

EJECUCIÓN DE PROGRAMAS

Ejecución en el fondo & , kill, nice y nohup

Para ejecutar un programa en el fondo, es decir, recuperando inmediatamente el control del terminal, basta añadir el carácter ***&*** al final del comando de ejecución:

program <datos.d >resultados.r &

inmediatamente aparecerá en el terminal, debajo de esta línea, un número que es el ***número de proceso*** de la ejecución de este programa. Para detener definitivamente dicha ejecución (no se puede detener temporalmente) se puede utilizar el comando ***kil***l:

kill númerodeproceso

La ejecución de un programa en el fondo no impide que aparezcan en la pantalla los mensajes de error que se produzcan (a no ser que se haya redirigido la salida de errores), y que el programa se pare cuando se salga del sistema. Para que el programa continúe ejecutándose aún cuando nosotros hayamos terminado la sesión, hay que utilizar el comando ***nohu***p:

nohup program

Si no se utilizan redirecciones todas las salidas del programa se dirigen a un fichero llamado ***nohup.ou***t. Cuando se utiliza ***nohup*** el ordenador entiende que el usuario no tiene prisa y automáticamente disminuye la prioridad de la ejecución. Existe un comando, llamado ***nic***e, que permite realizar ejecuciones con baja prioridad, es decir se le indica al ordenador que puede ejecutar de forma más lenta esta aplicación si existen otras que sean más urgentes. Se utiliza en las formas,

nice program &

nice nohup program &

Para darle al programa la prioridad mínima habría que utilizar el comando,

nice -19 program &

donde el -19 indica la mínima prioridad.

Comando time

El comando ***tim***e, precediendo a cualquier otro comando, suministra información acerca del tiempo total empleado en la ejecución, del tiempo de CPU utilizado por el programa del usuario, y del tiempo de CPU consumido en utilizar recursos del sistema. Por ejemplo para saber el tiempo utilizado en la compilación y montaje del programa ***prueba.c*** utilizaríamos el comando,

time gcc prueba.c

Comando top

Linux incluye una aplicación llamada ***top*** cuya finalidad es manipular la ejecución de programas de una forma interactiva. Esta aplicación muestra una lista de los procesos que se están ejecutando. Los principales comandos de top son: u que muestra los procesos que pertenecen a un determinado usuario, k equivalente al comando kill para matar un proceso y h que muestra la ayuda del programa.

PROGRAMAS DE COMANDOS

El sistema operativo ***Linu***x, al igual que otros sistemas operativos, permite realizar programas de comandos, esto es, programas constituidos por distintos comandos que podrían teclearse interactivamente uno por uno en una terminal, pero que es muchas veces más cómodo agruparlos en un fichero, y ejecutarlos con una sola instrucción posteriormente.

Los comandos de ***Linux*** pueden ser *externos* - que implican la creación de un nuevo proceso, cuyo código está en ***/bin*** o ***/usr/bi***n- e *internos* - cuyo código está incluido en el del intérprete shell que los ejecuta.

Una cierta primera forma de agrupar comandos la ofrece ***Linux*** por medio del carácter ;. Por ejemplo, tecleando el comando,

date; ls; who

el ordenador ejecutará sucesivamente los comandos ***dat***e, ***ls*** y ***wh***o. También podría crearse con un editor de textos un fichero llamado ***comandos*** que contuviera las líneas siguientes:

date

ls

who

Para ejecutar este fichero de comandos puede teclearse,

sh comandos

o bien convertir el fichero ***comandos*** en directamente ejecutable por medio del comando ***chmod*** en la forma,

chmod a+x comandos

de modo que el programa de comandos ***comandos*** puede ejecutarse simplemente tecleando su nombre,

comandos

Los comandos ***sh comandos*** y ***comandos*** no son enteramente equivalentes. Así, el primero de ellos exige que el fichero ***comandos*** esté en el directorio de trabajo, mientras que el segundo sólo exige que el fichero ***comandos*** esté en uno de los directorios de búsqueda de comandos especificados en la variable ***PAT***H.

Cuando se ejecuta un fichero de comandos ***Linux*** abre lo que se llama un nuevo shell, es decir un nuevo entorno para la ejecución de los comandos. Para que las variables del caparazón original conserven su valor en el nuevo caparazón es necesario prepararlas con la sentencia ***export*** antes de abrir el nuevo shell. Por ejemplo, como consecuencia de lo que se acaba de decir, si en el interior de un fichero de comandos se cambia de directorio con el comando ***c***d, al acabar la ejecución de dicho fichero volveremos automáticamente al directorio inicial.

Introducción de comentarios

Para introducir líneas de comentarios en un programa de comandos basta comenzar dichas líneas con el carácter #. Hay que tomar la precaución de que este carácter no sea el primer carácter del fichero de comandos, porque entonces el ordenador interpreta que el programa está escrito en ***C-shell*** (una variante especial de UNIX desarrollada en la Universidad de Berkeley) y el resultado es imprevisible. Puede ser buena práctica comenzar todos los ficheros de comandos con una línea en blanco.

Variables del shell

UNIX permite definir variables en un fichero de comandos en la forma,

USER=/mnt/mecan/juanto

TERM=hp2392

...

Es una práctica habitual el utilizar nombres con letras mayúsculas para las variables del caparazón. Para recuperar el valor de una variable hay que precederla con el carácter . Por ejemplo, utilizando en otra parte del programa ***TER***M, en dicho lugar se sustituiría ***TERM*** por su valor, esto es, hp2392. El shell del ***Linux*** tiene definidas para cada usuario unas variables estándar. Para averiguar cuáles son basta teclear el comando siguiente,

set

Para definir otras variables propias de cada usuario puede utilizarse el fichero ***.profil***e, que es un fichero de comandos propio de cada usuario que se ejecuta automáticamente al hacer el login. Para definir variables que contengan espacios en blanco deben encerrarse entre caracteres ***(***') o ***(***"), como por ejemplo,

FECHA="31 de Diciembre de 1986"

más adelante se verá la diferencia entre el carácter (') y el carácter (").

Comando echo

El comando ***echo*** imprime un determinado texto en la terminal. Un ejemplo de utilización de dicho comando puede ser el siguiente:

echo Me gusta el sistema operativo UNIX

El comando ***echo*** es de gran utilidad en los ficheros de comandos. Cuando el texto que se desea escribir en la terminal contiene alguno de los caracteres especiales de UNIX (**** ? [ ] > >> < & ;*** \ ') hay que tomar precauciones especiales desconectando su significado. Una forma de hacerlo es precediendo dicho carácter con la barra invertida (\). Así, para escribir mediante el comando ***echo*** tres asteriscos, utilizaríamos

echo ***

si no utilizáramos la barra invertida, el asterisco se interpretaría como un carácter de sustitución y se imprimiría el nombre de todos los ficheros del directorio. Otra forma de anular el significado de los caracteres especiales es encerrando el texto a escribir mediante comillas (") o entre apóstrofos normales ('). Los apóstrofos (') anulan el significado de todos los caracteres comprendidos entre ellos. Así pues, el triple asterisco lo podríamos escribir con el comando,

echo ''***

Las comillas (") son menos restrictivas, y anulan el significado de todos los caracteres excepto los tres siguientes: (` \). Esto es muy importante porque si VAR es el nombre de una variable, y VAR aparece en un comando echo entre apóstrofos se escribe VAR, mientras que si aparece entre comillas se escribe el valor de la variable, al cumplir el carácter su cometido.

El carácter (\) tiene otros significados, además del ya visto de anular el significado especial de otros caracteres. Así, sirve como indicador de que un comando continúa en la línea siguiente. Cuando se utiliza en la definición interactiva de un comando, en la línea siguiente aparece el prompt secundario (>), que indica que se debe seguir tecleando el comando. Cuando en un comando echo aparecen los caracteres (\c) y (\n) quiere decir, respectivamente, que no se cambie de línea y que se salte de línea, al escribir por la pantalla.

El carácter apóstrofo inverso o acento grave (`) tiene también un significado especial. Cuando en un comando echo aparece el nombre de otro comando encerrado entre apóstrofos inversos (por ejemplo, `date`, `who`, `ls`, ...), el nombre de dicho comando se sustituye por el resultado que genera al ejecutarse interactivamente. Un ejemplo podría ser el siguiente:

echo "Los usuarios del sistema son \n\n `who`"

El lector puede hacer la prueba y observar el resultado correspondiente.

Parámetros de los ficheros de comandos

A los ficheros de comandos pueden pasárseles como parámetros un conjunto de una o más variables. Dentro del fichero de comandos estas variables o parámetros se conocen

con los nombres 0, 1, 2, ..., 9. La variable ***0*** representa el propio nombre del fichero de comandos, y 1, 2, ..., ***9*** son los nombres de los parámetros propiamente dichos.

Vamos a comenzar viendo un ejemplo muy sencillo de programa de comandos al que se le pasa sólo una variable o parámetro. El comando de borrar de ***Linux rm*** no confirma la operación de borrado si no se le pone la opción ***(-***i). Esto es peligroso porque uno fácilmente puede olvidarse de teclear dicha opción y borrar lo que no quería borrar. Vamos a crear un fichero de comandos llamado ***del*** que incluya dicha opción. Dicho fichero podría estar formado por,

echo "Quiere borrar el fichero 1?"

rm -i 1

Después de darle a este fichero el correspondiente permiso de ejecución con el comando ***chmo***d, podríamos borrar con confirmación el fichero ***file*** tecleando, ***del file.***

Dentro del fichero de comandos, ***0*** valdría ***del*** y ***1*** valdría ***fil***e.

Un programa de comandos más complicado y que utiliza dos parámetros podría ser el contenido en el fichero ***cambi***o, que intercambia el nombre de dos ficheros:

mv 1 ficheropufo

mv 2 1

mv ficheropufo 2

Este fichero se ejecutaría en la forma,

cambio file1 file2

En este ejemplo ***0*** es ***cambi***o, ***1*** es ***file1*** y ***2*** es ***file***2. En realidad a un fichero de comandos se le pueden pasar todos los argumentos que se deseen, aunque sólo hay nombre específico para los nueve primeros (más el propio nombre del comando). El número de argumentos que se le pasa está contenido en la variable #. La variable * contiene el conjunto de todos los parámetros. Un nuevo ejemplo puede aclarar algo más este punto.

Si el programa ***del*** que hemos hecho previamente lo hubiéramos utilizado en la forma:

del *.f

teóricamente debería de borrar, con confirmación, todos los ficheros ***Fortran*** del directorio. En la práctica no es así, porque ***(*.***f) no representa un único argumento sino muchos argumentos (todos los ficheros ***Fortran*** del directorio). Como resultado sólo se borra el primer fichero ***Fortra***n. Para borrar todos e indicarnos además cuántos ficheros hay, el fichero ***del*** podría estar compuesto por los siguientes comandos:

echo "Hay # programas Fortran \n"

rm -i *

El comando ***shift*** hace posible utilizar y distinguir parámetros que están más a la derecha del noveno lugar en la llamada al programa de comandos. En efecto, cuando se

llama al comando ***shif***t, ***2*** se convierte en 1, ***3*** en 2, etc, y lo que hubiera sido ***10*** en 9, con lo cual ya se puede referenciar. El comando ***shift*** deja inalterado ***0*** y puede utilizarse tantas veces como se desee.

Otras posibilidades de los ficheros de comandos

Los ficheros de comandos tienen muchas más posibilidades que las que se han apuntado en esta Introducción: pueden leer variables, preguntar por la existencia de un fichero y por si es ejecutable o no, y admiten construcciones lógicas del tipo ***I***F, ***D***O, ***DO WHIL***E, etc. Para utilizar estas posibilidades acudir al manual correspondiente.

COMPILANDO PROGRAMAS EN LINUX

Linux como cualquier sistema ***Unix*** que se precie incluye un compilador de ***C y C***++. Esto no implica que se esté limitado a estos dos lenguajes de programación. Por el contrario existen una gran cantidad de compiladores gratuitos para los lenguajes más importantes.

El compilador de ***C/C***++ de ***Linux*** es un compilador de línea de comandos, esto es, no dispone de ninguna interfaz gráfica que facilite la programación y compilación del código. Existen eso sí editores de texto capaces de mostrar la sintaxis del código resaltada como ***kwrit***e, aunque la compilación hay que realizarla manualmente a través de una consola o terminal.

Compilación y linkado

El primer paso para crear un programa, por ejemplo en C++, es crear el fichero de código fuente, y guardarlo en un fichero de texto por ejemplo ***e1.cp***p. Tras esto hay que compilar el programa. Para esto se empleará el comando g++, de la siguiente forma:

g++ e1.cpp

Con lo que conseguiremos que se compile el programa. Con este comando hemos conseguido que se cree un programa llamado ***a.out*** en el directorio de trabajo. Para ejecutarlo emplearemos

./a.out

Normalmente no desearemos que el programa se llame ***a.out*** sino que tenga un nombre más significativo, como ***ejemplo_***1, para conseguirlo emplearemos:

g++ -o ejemplo_1 e1.cpp

Si queremos ejecutar el programa emplearemos ***./ejemplo_***1. Si el programa escrito realiza emplea alguna función que no se encuentre en la librería estándar hay que incluirla en la orden de compilación, por ejemplo:

g++ -o ejemplo_2 e2.cpp -lname

Donde name es el nombre de la librería. Por defecto estas se guardan en un fichero cuyo nombre es

libname.so, cuya localización suele ser ***/usr/li***b.

Si el programa no está escrito en C++ sino en C las opciones de compilación son las mismas salvo que en lugar del programa ***g++*** se empleará el comando ***gc***c.

gcc –o ejemplo_3 e3.c

Comando make

Este comando sirve para organizar la compilación y el enlazado de programas complicados que dependen de muchos módulos y librerías diferentes. Cuando se ejecuta este comando, se construye un nuevo ejecutable volviendo a compilar sólo aquellos ficheros fuente que son más recientes que el los ficheros compilados correspondientes, teniendo en cuenta para ello las fechas de última modificación de cada fichero. Este comando se basa en un fichero ASCII (llamado por defecto ***makefil***e) que contiene una relación de dependencias entre los distintos módulos, así como las acciones que hay que realizar para poner a punto cada módulo, es decir para pasar de un fuente a un objeto, por ejemplo. Este comando tiene la siguiente forma general:

make [–f makefilename] [–arg_opt] [exe_name]

El fichero ***makefile*** (con éste o con otro nombre invocado por medio de la opción –f) contiene cuatro tipos de líneas diferentes:

• Líneas de *comentari*o, que comienzan por el carácter (#). Si en una línea cualquiera aparece el carácter (#), se ignora todo lo que aparece a continuación de dicho carácter en dicha línea.

• Líneas de *definición de macro*s. Tienen la forma general,

IDENTIFICADOR = cadena_de_caracteres

Si en alguna otra línea aparece (IDENTIFICADOR), dicha ocurrencia se sustituye por ***cadena_de_caractere***s. No es necesario que el nombre del identificador esté escrito con mayúsculas, pero es una costumbre bastante extendida el hacerlo así. Mediante el uso de macros se pueden representar brevemente pathnames o listas de nombres de ficheros largos. Si el identificador tiene una sola letra, no hace falta poner los paréntesis. El comando ***make*** tiene una serie de macros definidas por defecto que se pueden listar con el comando ***make*** –p.

Líneas describiendo las *relaciones de dependenci*a. Tienen la forma,

file.o fila.o ... : file1.cpp file2.cpp ...

La lista de ficheros que están a la izquierda del carácter (:) dependen de los ficheros que están a la derecha. En estas líneas se realiza la sustitución habitual de los caracteres (? *[])

Líneas de *comandos shel*l, comenzando siempre por un tabulador. Estas líneas representan las acciones que hay que realizar para actualizar los ficheros dependientes, según las relaciones de dependencia descritas en la línea anterior. En una misma línea de comandos puede haber varios comandos separados por un carácter (;), y de este modo se ejecutan en un mismo proceso; si hay varias líneas de comandos, cada línea se ejecuta en un proceso diferente. Estos comandos shell (compilaciones, etc.) se ejecutan o no según las fechas de los ficheros correspondientes. Hay también una lista de dependencias implícitas y de macrodefiniciones standard que se pueden obtener con el

comando ***make -p*** (la salida de este comando puede tener varios cientos de líneas). A continuación se presentan algunos ejemplos de ficheros ***makefil***e.

A continuación se muestran una serie de ejemplos

```
# Esto es un ejemplo de fichero makefile

# Definiciones de macros

ma = ma27ad.o ma27bd.o ma27cd.o

TEST: test.o (ma)

xlf –o TEST –O (ma)

test.o: test.f

xlf –c –O test.f

ma27ad.o: ma27ad.f

xlf –c –O ma27ad.f

ma27bd.o: ma27bd.f

xlf –c –O ma27bd.f

ma27cd.o: ma27cd.f

xlf –c –O ma27cd.f
```

Hay informaciones que no es necesario dar por que se suponen por defecto. Por ejemplo, si el fichero objeto ***(.***o) y el fichero C ***(.***c) o C++ ***(.cp***p) tienen el mismo sufijo, no hace falta definir esas relaciones de dependencia, que por otra parte son triviales.

Búsqueda avanzada en ficheros. Expresiones regulares

A veces se desea encontrar las líneas de un fichero que contienen una palabras o palabras determinadas. Cuando el texto que se desea encontrar es único, lo que hay que hacer es ponerlo tal cual en la sección del comando que define la búsqueda, por ejemplo

grep "PATATAS" Lista_de_la_compra.txt

Sin embargo, en otras ocasiones el texto que se desea buscar no es único, es decir, no está unívocamente determinado como en el ejemplo anterior, sino que debe cumplir unas ciertas condiciones, como la de estar escrito con mayúsculas, comenzar por determinado carácter, estar a principio o final de línea, etc. Este problema se puede resolver en muchos comandos de ***Linux*** por medio de las expresiones regulares que se van a presentar a continuación.

Las *expresiones_regulares* son una forma de describir patrones para la búsqueda de unas determinadas líneas dentro de uno o más ficheros ASCII. Se trata pues de encontrar las líneas cuyo contenido cumple ciertas condiciones, que se definen en la *expresión_regula*r.

Caracteres especiales

En una *expresión_regular* se pueden utilizar algunos caracteres que tienen un significado especial.

Son los siguientes:

[comienzo de la definición de un conjunto de caracteres

. un carácter cualquiera, excepto el <eol>

* un conjunto de caracteres cualesquiera, excepto el primer carácter de una expresión o inmediatamente después de la secuencia \(

] terminación de la definición de un conjunto de caracteres

- sirve para definir el conjunto de caracteres que van del que le precede al que le sigue. Si va detrás del [o delante del], no es especial

ˆ comienzo de línea, si está al comienzo de la expresión

ˆ conjunto complementario (el que no cumple la condición), si está justo después del [que abre la definición de un conjunto

fin de línea, cuando está al final de una expresión.

\ quita el significado especial a un carácter, excepto en la definición de un conjunto de caracteres

Expresiones regulares de un solo carácter

Se trata de buscar palabras o conjuntos de un solo carácter, que cumple ciertas condiciones. A continuación se presentan algunos ejemplos:

* representa el carácter *

. cualquier carácter, excepto el <eol>

[a-f] un carácter cualquiera entre la a y la f

[A-Z] cualquier letra mayúscula

[ˆa-d] cualquier carácter que no sea una letra entre la a y la d

[:clase:] donde *clase* puede ser: *digit* (cifra del 0 al 9), *xdigit* (cifra hexadecimal), *alpha* (letra cualquiera), *upper* (letra mayúscula), *lower* (letra minúscula), *alnum* (letra o dígito cualquiera),

space (un espacio en blanco), *cntrl* (carácter de control), *punct* (un carácter de puntuación) y *print* (carácter imprimible).

Expresiones regulares generales

Se pueden formar de acuerdo con las siguientes reglas:

- una expresión regular de un sólo carácter

[a-z] cualquier letra minúscula

- una expresión regular de un sólo carácter, seguida del carácter *, representando entonces todas las palabras de longitud positiva o nula que se pueden construir con los caracteres aceptados por la una expresiones regulares de un sólo carácter

[a-z]* cualquier palabra escrita con minúsculas

- concatenando (poniendo una a continuación de la otra) dos expresiones regulares construidas previamente

[a-z][A-Z] ***cualquier palabra de dos letras, de las cuales la primera es minúscula y la segunda mayúscula***

- Una expresión_regular definida en la forma \(expresiones regulares\) representa la propia expresiones regulares (es decir, definida ella sola), pero define una forma de referirse luego a esa expresiones regulares. En efecto las expresiones regulares definidas de esta forma quedan afectadas por un número del 1 al 9, y es posible luego hacer referencia a una expresiones regulares por medio del número que le corresponde, en la forma \número. Si \número va seguido de un *, esa subexpresión puede repetirse un número cualquiera de veces.

^\(.*\)\1\1 Al comienzo de la línea, un campo formado por un carácter cualquiera que se repite las veces que sea, volviendo a aparecer dos veces mas antes de que se acabe la línea. Esta expresión detectaría las líneas que contienen palabras (o conjuntos de palabras) triples.

- una expresiones regulares de un sólo carácter seguida de \(*entero*\) representa *entero* apariciones consecutivas de alguno de los caracteres aceptados por la expresiones regulares de un sólo carácter. Si va seguida de \(*entero*,\) representa un número mínimo de *entero* apariciones consecutivas. Si va seguida de \(*entero*1, *entero*2\) representa un mínimo de *entero1* y un máximo de *entero2* apariciones consecutivas. Los números enteros deben estar comprendidos entre 0 y 255.

- toda expresión que comienza con ^ indica que los caracteres buscados deben estar a comienzo de la línea

^[a-z]* selecciona las líneas que sólo contienen letras minúsculas

- toda expresión que termina con indica que los caracteres buscados deben estar a final de la línea

^[a-z]\{3\}[^a-z]\{3\} selecciona las líneas que comienzan con tres minúsculas, terminan con tres caracteres cualesquiera pero que no son minúsculas, y no tiene ningún otro carácter entre medio.

Comando/Sintaxis	Descripción	Ejemplos
cat *fich1* [*...fichN*]	Concatena y muestra un archivos	cat /etc/passwd
diff [-*e*]*arch1 arch2*	Encuentra diferencia entre archivos	diff foo.c newfoo.c
du [-*sabr*] *fich*	Reporta el tamaño del directorio	du -s /home/
find *dir test acción*	Encuentra archivos.	find . -name ``.bak" -print

grep [*-cilnv*] *expr archivos*	Busca patrones en archivos	grep mike /etc/passwd
head *-count fich*	Muestra el inicio de un archivo	head prog1.c
ln [*-s*] *fich acceso*	Crea un acceso directo a un archivo	ln -s /users/mike/.profile .
tail *-count fich*	Muestra el final de un archivo	tail prog1.c
vi *fich*	Edita un archivo.	vi .profile
at [-lr] hora [fecha]	Ejecuta un comando mas tarde	at 6pm Friday miscript
echo string	Escribe mensaje en la salida estándar	echo ``Hola mundo''
finger usuario	Muestra información general sobre un usuario en la red	finger nn@maquina.aca.com.co
id	Número id de un usuario	id usuario
kill [-señal] PID	Matar un proceso	kill 1234
man comando	Ayuda del comando especificado	man gcc
		man -k printer
ps [axiu]	Muestra información sobre los procesos	ps -ux
	que se están ejecutando en el sistema	ps -ef
who / rwho	Muestra información de los usuarios	who
	conectados al sistema.	

/usr/bin/gpasswd -a usuario grupo	Asigna cierto usuario a ese grupo.
at 7:00 cdplay<Ctrl>d	Programa el computador para hacer sonar el cd a las 7:00.
atq	Lista las tareas automáticas programadas.
atrm 8	Elimina la tarea número 8.
bash	Extension del sh.
chs	C shell.
date -s "04/25/05" +"%D"	Cambia la fecha del sistema al 25 de Abril del 2005 .
date -s "15:00:00" +"%T"	Cambia la hora del sistema a las 15:00.
dd if=/dev/fd0 of=imagendedisco	Crea una imagen del disco con el nombre imagendedisco.
dd if=imagendedisco of=/dev/fd0	Graba el archivo imagen a un disco.
df	Muestra el espacio libre del disco.
dpkg -i archivo.deb	Instala un paquete .deb (Debian).
du -c ficheros	Muestra el total que ocupan los ficheros.
du -s /directorio	Muestra el espacio que ocupa el directorio.
echo $PATH	Muestra el path actual.
Editar este fichero para memorizar una tarea: etc/crontab	Ej: 00 00 20 10 * echo 'Ha llegado el invierno' >/dev/null (esto último es para que no se envíe por mail al root). Los siguientes directorios son para meter programas a ejecutar cada cierto tiempo (copiarlos dentro) /etc/cron.hourly/ /etc/cron.daily/ /etc/cron.weekly/
emacs /etc/sysconfig/network	Para cambiar el Gateway (puerta de enlace)en redhat.
emacs fichero	Edita el fichero Ctrl+X Ctrl+S (guarda), Ctrl+X Ctrl+C (sale).
find /directorio -name 'fichero'	Busca a partir de /directorio el fichero especificado.
for x in $(ls /directorio) do echo $x done	Para hacer un bucle en un script bash de los ficheros de un directorio.
grep	Busca una cadena en un fichero. cat fichero.txt | grep texto a buscar
if [-x /path/to/dnetc]; then echo "Starting Distributed.net Client..." /path/to/dnetc -quiet fi	El fichero /etc/rc.d/rc.local es similar al autoexec.bat de ms-dos y por ejemplo podemos poner estas líneas. Si no se está ejecutando arranca dnetc.
ifconfig	Te dice tu dirección IP aunque sirve también para configurar la tarjeta de red.
joe fichero	Editor de texto.
kill -9 numeroPID	Mata el proceso numeroPID que queramos (finaliza su ejecución). kill -HUP numero PID Hace lo mismo. Kill -STOP numeroPID Para hasta nueva orden el proceso NumeroPID. Kill -CONT numeroPID Continua el proceso numeroPID parado anteriormente.
killall nombreproceso	Mata el proceso con ese nombre.
ksh	Korn shell, combina sh y csh.
ln -s /mnt/cdrom/Mame/roms /	Crea un enlace blando al directorio roms del cdrom como si fuera /usr/games/lib/xmame (xmame no debe existir)

Linux	DOS	Significado
cat	type	Ver contenido de un archivo.
cd, chdir	cd, chdir	Cambio el directorio en curso.
chmod	attrib	Cambia los atributos.
clear	cls	Borra la pantalla.
ls	dir	Ver contenido de directorio.
mkdir	md, mkdir	Creación de subdirectorio.
more	more	Muestra un archivo pantalla por pantalla.
mv	move	Mover un archivo o directorio.
rmdir	rd, rmdir	Eliminación de subdirectorio.
rm -r	deltree	Eliminación de subdirectorio y todo su contenido.