Tutorial de comandos básicos para Linux

Javier Fernández Rivera - www.aurea.es

En este tutorial de comandos se trataran indistintamente UNÍX o LINUX, cuando hacemos referencia a uno vale para el otro y a la inversa.
A la hora de presentar la sintaxis de los comandos se sigue el formato estándar de todo manual de informática, donde los parámetros introducidos entre signos " < > " son parámetros que deben pasarse de forma indispensable. Mientras que los argumentos o parámetros que estén entre " [ ] " su paso no es necesario sino opcional.

IMPORTANTE!!!: UNÍX distingue entre nomenclaturas. Así pues distingue mayúsculas y minúsculas (case sensitive = caso sensible). Lo que quiere decir que no es lo mismo poner CD a poner cd. ☺

La mayoría de ordenes en UNÍX o por lo menos las mas comúnmente usadas son de 2 letras.

Ayuda de comandos

Comando: man
Etimologia: man (manual).
Sintaxis: man [-fk] <comando>
Donde man nos proporciona una ayuda sobre el comando que especificamos como parámetro. Lo malo de esta ayuda es que viene en ingles. Pero en ella se pueden ver todos los parámetros que permite cualquier comando y su explicación además de la definición del comando y sus ya mencionados argumentos.
Parámetros:

- **-f** esta opción hace que man muestre la definición del comando en una sola línea.
- **-k** la opcion –k de la orden man busca en todas las descripciones, incluidos los nombres de las ordenes, las que incluyen las palabras especificadas.

Ejemplo: man –k connect
Buscara la palabra connect.

Caracteres especiales

Carácter *****
Ejemplo: ls file*
Esto buscaría todos los ficheros que empezaran por file el resto de caracteres que le siguen es sustituido por el asterisco y se muestran en pantalla.
Carácter **?**
Ejemplo: ls file?*
Esto localizara aquellos ficheros que empiecen por file y a continuación un carácter que será sustituido por el ? luego puede o no contener mas caracteres.
Resumiendo: el carácter * absorbe cualquier carácter o caracteres, y el carácter ? absorbe un solo carácter.
Carácter **~** (altgr+126) Este carácter devolverá la ruta del home, ejemplo: ~/dir1/dir2/

Comandos de información

Información de usuarios conectados

Comando: who
Etimología: who+is (Quien?) [es]
Sintaxis: who [-sHqT]
Este comando nos muestra los usuarios coenctados e información sobre tales.
Si ponemos: who me
Nos hará un who a nosotros mismos, dándonos información sobre nosotros.

Información del usuario

Comando: finger
Sintaxis: finger usuario
Este comando mostara información del usuario: directorio home, ultimo mensaje leído, login, shell, etc.

Comandos programables

Comando para mostrar texto en pantalla
Comando: echo
Sintaxis: echo texto
Ejemplo: echo Hola mundo!!!
Esto mostrara en pantalla “Hola mundo!!!”

Creando comandos personalizados
Comando: alias
Etimología: Sobrenombre
Sintaxis: alias [-p] [nombre[=valor]]
Este comando crea un nombre de comando que realiza la función que nosotros predefinamos con anterioridad.
Lo mejor es ver un ejemplo:
alias saludo=`echo nas xaval ☺`
Ahora cuando pongamos en el shell el comando (alias) “saludo”, linux nos mostrara un mensaje “nas xaval ☺”.

Eliminando comandos personalizados
Comando: unalias
Sintaxis: unalias <namealias>
Eliminara el alias pasado como parámetro.

Comandos de datos temporales

Mostrando calendario
Comando: cal
Etimología: Calendario
Sintaxis: cal [mes[año]]
Este comando ofrece un calendario mensual. Puede tener comoargumentos el mes y año.
Ejemplo: cal 01 2001
Mostrara el mes de enero del 2001

Mostrando fecha
Comando: date
Sintaxis: date
Esto mostrara en formato la fecha actual.
Etimología: date (fecha)

Comandos varios

Limpiando pantalla
Comando: clear
Etimología: Limpiar
Sintaxis: clear
Este comando limpia la pantalla.

Ejecutando scripts (guiones)
Comando: sh
Sintaxis: sh script
Donde script es el nombre del guion a ejecutar.

Temporizando proceso de comando.
Comando: time
Etimología: tiempo (hora).
Sintaxis: time [comando]
Devuelve el tiempo de ejecución total, el tiempo que el sistema ha dedicado a ese usuario y el tiempo de preparación del programa o comando pasado como argymento.

Visualizando bloques libres o usados
Comando: df

Comandos para operaciones de comunicación

Los comando write y talk permiten enviar y recibir mensajes y los programas mail y mailx gestionan el correo electrónico.

Comunicación bidireccional unica
Comando: write
Etimología: Escribir
Sintaxis: write <user> [terminal]
Ejemplo:
write Quasi
Hola, Quasi
Como te va ese OrioN ScripT?
^d
El usuario Quasi vera en pantalla lo siguiente:
Message from pepe tty0 [ved mar 9 09:02:34]
Hola, Quasi
Como te va ese OrioN ScripT?
EOF

Lógicamente es necesario que los dos usuarios estén conectados. Tal información la podemos obtener con el ya visto comando who.

Este comando sirve para proveer un tipo de comunicación de corto-cambio (duplex = bidireccional unico (en un solo sentido)).
Para realizar una comunicación mas fluida (fullduplex = bidireccional simultanea (en ambos sentidos a la vez)). Se usa el comando talk

Comunicacion bidireccional simultanea
Comando: talk
Etimología: Hablar
Sintaxis: talk <user>
Ejemplo:
talk Quasi

Y Quasi recibe el mensaje: talk: conexión solicitada por pepito ...
Si Quasi pone: talk pepito
La pantalla se divide en 2 partes una para teclear y otra para recibir. Para terminar teclear control+c.

Impidiendo la recepción de mensajes.
Comando: mesg
Etimología: message (mensajes)
Sintaxis: mesg [y|n]
Ejemplo:
mesg n
Suponiendo que ese comando lo halla ejecutado Quasi, cuando alguien intente mandar un mensaje (talk o write) a Quasi este no lo recibirá al usuario que lo intenta le pondrá en pantalla “permiso denegado”.
Para saber nuestra situación actual, ósea si el cierre de mensajes esta activado o no, basta con poner: mesg

Mensaje massivo
Comando: wall
Sintaxis: wall
Ejemplo:
wall
El mejor ScripT del mundo mundias: OrioN scripr http://ircorion.cjb.net
^d

Gestionando correo
Comando: mail
Sintaxis: mail [user1, user2, user3, etc]
Si no especificamos el usuario al que queremos mandar el mail. Nos entrara en el programa de gestión de nuestro propio correo, desde el cual podremos ver y editar nuestro correo.
Si especificamos un parámetro con el nombre de usuario de algún equipo, el programa procesa unos datos (subject, texto, etc.) y lo envía al user.

Ejecutando mail sin pasarle argumentos entramos en el entorno del programa de gestión de mails. En el cual podemos usar las siguientes ordenes.

- **?** muestra ayuda
- **n** donde n es el numero del mail que queremos visualizar en pantalla.

- **p** muestra el mensaje actual
- **h** muestra las cabeceras de los mensajes
- **d** borra el mensaje
- **q** sale del mail
- **s** file Donde guarda el mensaje actual dentro de file. Si no se indica file se guardara en mbox.
- **R nº** Si deseamos responder al mensaje numero n.

Comandos para operaciones con directorios

Moviéndose por el sistema de ficheros (directorios y subdirectorios)

Comando: cd
Sintaxis: cd <directorio>
Donde directorio es el directorio al que queremos ir.
Etimología: change directory, cambio de directorio.
Parámetros:

- **cd ..** Nos dirigiría al directorio padre del directorio en el que estamos. NOTA: es distinto al MSDOS, en este caso los ".." están separados del cd.
- **cd .** Nos dejaría en el mismo lugar puesto que dirige al directorio en el que estamos.
- **cd** Si no pasamos parámetro o argumento alguno al comando cd, este nos llevara a nuestro directorio de origen. En unís el directorio de origen es llamado HOME podemos devolver su ruta con la variable de entorno $HOME o con el símbolo "~" (alt +126).

NOTA: de monos cuenta que ".." hace referencia al directorio padre, mientras que "." Hace referencia al directorio actual.

Visualizando el contenido de los directorios

Comando: ls
Sintaxis: ls [-FltrCai]
Donde los parámetros pasados al ls, filtraran la información a la forma deseada y predefinida.
Parámetros:
-F Indica que muestre los ficheros y subdirectorios que hay dentro del directorio con signos identificativos a cada uno de ellos.
My-dir/
My-file
My-file-exe*
Si la entrada lleva el signo "/" se trata de un subdirectorio que cuelga del directorio en el que estamos.
Si no se muestra ningún signo se trata de un fichero.
Si se muestra un "*" (asterisco), se estará tratando de un fichero ejecutable.
-l Con este parámetro, el comando ls nos muestra el formato largo de visualización de entradas de ficheros y directorios. En el que se dan en primer lugar los permisos asignados a los ficheros o directorios (si esos permisos empiezan por un "-" se trata de un fichero, si empieza por una "d" se estará tratando de un directorio). A continuación en la segunda columna se muestran el numero de enlaces o links a ese fichero o directorio. La tercera y cuarta columna son el nombre del propietario y su grupo respectivamente. El resto son el tamaño, la fecha de ultima modificación, la hora, etc.
-t Con este parámetro filtramos las entradas de los ficheros y directorios para mostrarlos ordenados según su fecha de creación.
-r Muestra las entradas de ficheros y directorios en orden inverso (por defecto se muestran en orden alfabético).
-C Presenta los nombres de los ficheros en columnas.
-a Muestra los ficheros o directorios ocultos (empiezan por un punto).
-i Con este argumento sé vera el numero de inodo perteneciente a cada fichero.
-F Muestra los directorios con su símbolo identificativo "/"

Creando directorios

Comando: mkdir
Sintaxis: mkdir [-pm]
Si introducimos (mkdir dir1 dir2 dir3), crearemos 3 directorios en la ruta donde nos encontramos.
Etimología: Make directory (crear directorio).
Parámetros:

- **-p** crea los directorios y subdirectorios indicados en la ruta y que no existan ya.
- **-m**

Eliminando directorios

Comando: rmdir
Etimología: remove directory (eliminar directorio).
Sintaxis: rmdir [-p] <directorio>
Donde [parámetros] será para borrar con ciertas opciones. Y directorio será el directorio o ruta de directorios/subdirectorios a eliminar.

NOTA: Para que el comando rmdir sea efectivo y elimine el o los directorios. Deben de cumplirse las siguientes condiciones.

1. El directorio tiene que estar vació.
2. Su userid debe tener permiso, asignado al directorio padre, para escribir y ejecutar.
3. El directorio no puede ser su directorio de trabajo.

Comandos para operaciones con ficheros u archivos

Copiando ficheros

Comando: cp
Etimología: copy directory (copiar directorio)
Sintaxis: cp [-lR] <entrada> <salida>
Donde al comando cp le pasamos como argumentos indespensables, la entrada que serian los ficheros o el fichero que queremos copiar, y la salida que seria el lugar (directorio o subdirectorio) en el cual queremos dejar la copia.
Parámetros:

- **-l** Para crear enlaces.
- **-R** Copia recursivamente todos los ficheros que cuelgan de la ruta especificada.

Ejemplo: cp file1 .
En este ejemplo se copia el fichero (file1) al directorio actual que es devuelto al pasar como parámetro el signo ".", esto indica que se copie al directorio en el que nos encontramos.

Moviendo ficheros

Comando: mv
Etimología: movie directory (miver directorio).
Sintaxis: mv <entrada> <salida>
Donde entrada será el fichero o los ficheros que queremos mover. Y la salida será el directorio al que van dirigidos.
NOTA: Tanto mv como cp sobrescriben los ficheros sin pedir confirmación. ☺

Eliminando ficheros

Comando: rm
Etimología: Remove directory (eliminar directorio).
Sintaxis: rm [-ir] <fichero/s>
Donde al comando rm le podemos pasar opcionalmente unos parametros y donde ficheros serán aquellos archivos que queremos eliminar.
Parámetros:

- **-i** Este parámetro hace que se pida confirmación antes de eliminar un fichero.
- **-r** Con este argumento haremos que el comando rm elimine también directorios (tengan dentro ficheros o no).

Ejemplo: rm –r *
Esto elimara todos los directorios, subdirectorios y ficheros de la ruta en el que nos encontramos.

Mostrando diferencias entre ficheros

Comando: diff
Sintaxis: diff file1 file2

Viendo el contenido de ficheros

Comando: more
Sintaxis: more [-bh!q] <fichero/s>
Donde a more le pasamos el fichero. La función del comando es visualizar y procesar el listado de datos o información que hay dentro del fichero que le pasamos al comando. Tal información se lanza a pantalla.
Dentro del proceso del comando:

- La barra espaciadora: para avanzar a la pagina siguiente.
- **B** para volver a la pagina anterior.
- **¿** o **h** presenta la pagina de ayuda con las opciones de more.
- **!** ejecuta una orden
- **ctrl.+l** reestablece la pantalla que habia antes de ejecutar ? o !.
- **q** o **ctrl+c** finaliza la ejecución de more.

Comando: cat
Sintaxis: cat <files>
Ejemplo: cat file1 file2 file3
Veremos el contenido del fichero 1, del fichero 2 y 3. ☺

Analizando tipo de ficheros

Comando: file
Etimología: file (fichero o archivo)
Sintaxis: file [-f]

Este comando nos dirá el tipo de ficheros del directorio hay en el directorio donde lo ejecutemos. Si se trata de un fichero script (guión, texto o ejecutable)

Imprimiendo ficheros.

Comando: lp
Etimología: lp viene del puerto pararelo de la impresora (lpt1 o lptr).
Sintaxis: lp [-d] <files>
Donde a lp le pasamos los ficheros que queremos que se impriman.
Ejemplo: lp fichero1
Esto imprimirá el fichero1.
Parámetros:

- **-d** Sirve para especificar mas impresoras

Ejemplo: lp –d impresora1 file1 file2
Se lanzan a la impresora1 dos ficheros para su impresión: file1 y file2.
Ordenes para el comando

- Cancel: cancela o elimina los trabajos de la cola de impresión.
- Lpstat: muestra el estado de los trabajos pendientes de impresión.

Localizando ficheros

Comando: find
Etimologias: find (buscar).
Sintaxis: find <directorio> [–name,-u,-mtime] core –print
Donde directorio es la ruta donde se buscaran los patrones especificados.
La sintaxis del find es algo enrevesada, depende de los parametros que le pasemos.
Parámetros:

- **-name** Indicando el nonbre nos localizara aquellos ficheros cuyo nombre sea el pasado y los buscara en el del directorio especificado.

Ejemplo: find . –name core
Aquí buscamos aquellos ficheros con el nombre core dentro del directorio donde nos encontramos “.” Debido al punto.

Entrada y salida de datos

Ordenando contenido de fichero

Comando: sort [-r]
Este comando toma una serie de datos por la consola estándar de entrada (stdin) ósea el teclado. Y la saca por la consola estándar de salida (stdout) monitor.
Sort también puede ordenar el contenido de un fichero.
Ejemplo: sort < file1 > file2
Esto ordenaría el contenido de file1 y lo direcciona al fichero file2.
Peculiaridad: sort > file
Esto redireccionaría lo que se introduce por stdin (teclado) hacia un fichero llamado file.
Esto mismo podria ser hecho de la siguiente forma: cat file
Parámetros:

- **-r** Ordena en orden inverso.

Redirección no destructiva

Si hacemos: ls > listfiles
Crearemos un fichero llamado listfiles, este fichero contendrá el listado de ficheros y directorios hecho por ls. Si volvemos a realizar ese comando se recreara el mismo fichero.
Si hacemos: ls >> listfiles
De esta forma no perdemos el contenido del fichero listfiles en caso de que ya estuviera creado, lo que se hace es añadir al final de este fichero los datos del ls.

Uso de pipes (tuberías)

El uso del signo “|” (altgr+1) sirve para redireccionar la salida de un comando con la entrada del comando posterior. De esta forma podemos enlazar comandos y obtener los resultados de uno para dárselos al comando siguiente y en este operar como se quiera con los datos.
Comando: head
Sintaxis: head [-n] [file]
Uso: head –4
Este comando sacara la 4 línea que recibe de otro comando por medio de un pipe.
Ejemplo: head –3 file
Muestra las 3 primeras líneas del fichero file.
Este comando funciona de forma análoga al comando tail.

Ejemplo: tail [–n] <file>
Donde n se especifica el numero de líneas que se quieren mostrar del file determinado.
Ejemplo: tail –15 file
Esto sacara las ultimas 15 lineas del fichero file.

Ejecución de un comando dentro de otro

Si por ejemplo queremos usar un comando dentro de otro y que este se ejecute. Se usaran los signos “ ` ` ” .
Ejemplo: echo Hoy es: `date`
Esto hara que se ejecute dentro del comando echo el comando date y este devolverá un valor. Así pues al final se mostrara en pantalla: Hoy es: (la fecha que sea)

Permisos

En Linux cada fichero o directorio tienen unos permisos asignados por el usuario que los ha creado y dependiendo de esos permisos el resto de usuarios podrán o no realizar operaciones con estos ficheros o directorios (leer, ejecutar, eliminar, etc).

Para visualizar los permisos asignados a los distintos ficheros y directorios, basta con ejecutar un ls en formato largo, ósea: ls –l

Veamos unos ejemplos prácticos.

-rwxrwxrwx prueba1
drwxrwxrwx prueba2

El fichero llamado prueba1 se sabe que es un fichero porque el primer carácter es “-“. En el segundo caso se trata de un directorio puesto que el primer carácter es “d”.

El resto de caracteres son los referidos propiamente dichos con los permisos.

Haber, existen permisos para 3 clases de usuarios.
Los permisos para el propietario.
Los permisos para el grupo
Los permisos para todos los usuarios (el resto de usuarios que no son ni el propietario ni los del grupo).

Haber los permisos se dividen pues en 4 campos

-,---,---,---

El primer campo es el que indica si es fichero o se trata de un directorio como ya antes explique.
El segundo campo indica los permisos para propietario
El tercer campo indica los permisos para los usuarios del grupo
El cuarto campo indicara los permisos para el resto de usuarios.

Hay 3 tipos de permisos

De lectura: r
De escritura: w
De ejecución: x

Supongamos que tenemos un fichero con los siguientes permisos:

-rwxrw-r-- prueba
Haber, en ese fichero llamado prueba el propietario tiene asignados todos los permisos (lectura, escritura y ejecución), el grupo tiene permisos para leer y escribir el fichero, y por ultimo el resto de usuarios pueden solo leer el fichero.

Cambiando los permisos

Comando: chmod
Etimología: change mode (cambio de modos).
Sintaxis: chmod {a,u,g,o} {+,-} {r,w,x} <filenames>
Parámetros para referirnos a quien va dirigido el permiso:
a Referimos a todos los usuarios.
u Referimos al propietario.)
g Referimos al grupo.
o Referimos a otros.
Parámetros para decidir si el permiso se añade o se quita:
+ Añade permiso
– Quita permiso

Parámetros que decide que permiso se añade o quita:

- **r** De lectura
- **w** De escritura
- **x** De ejecución

Ejemplos:
chmod a+r stuff
Da a todos los usuarios acceso al fichero.

chmod +r stuff
Como arriba_ si no se indica a, u, g o o por defecto se toma a.

chmod og-x stuff
Quita permisos de ejecución a todos los usuarios excepto al propietario.

chmod u+rwx stuff
Permite al propietario leer, escribir y ejecutar el fichero.

chmod o-rwx stuff
Quita permisos de lectura, escritura y ejecución a todos los usuarios menos al propietario y a los usuarios del grupo del fichero.

Cambiando el grupo
Comando: chgrp
Etimología: change group (cambio de grupo).
Ejemplo: chgrp alumnos /home/joaquin/datos
Esto cambia el grupo de los tres ficheros a alumnos.

Mascara de creación de ficheros
Comando: umask
Etimología: User mask (mascara de usuario).
Sintaxis: umask [nnn]
Con este comando podemos ver la mascara actual, usada al crear los ficheros por defecto.

Manejo de enlaces.

Creando enlaces duros (hard links) y simbolicos
Comando: ln
Etimología: ln es la abreviatura de link (enlace).
Sintaxis: ln [-s] <file1> <file2>
Donde file1 sera el fichero real, el fichero físico. Y file2 siempre será el fichero link, el fichero enlace a file1. Con lo que file2 pasara a ser enlace duro del file1. Por tanto file2 pasa a tener el mismo numero de inodo que file1.
Parámetros
-s Este argumento servirá para crear enlaces simbólicos.

Que es un numero de inodo?: En Linux cada entra en el sistema de ficheros, ya sea directorio o fichero es identificada por un numero llamado inodo. Linux internamente direcciona siempre y maneja ese inodo. El nombre del fichero es una utilidad que nos brindan los sistemas operativos para poder recordar fácilmente el nombre de estos. Tarea tediosa seria trabajar con ficheros que fueran secuencias de números ☹.
Con la opción ls –l (formato largo) podemos listar los ficheros y directorios y ver que numero de enlaces hay a una entrada (fichero o directorio).

Ejemplo:
ls -l file1 file2
-rw-r--r-- 2 root root 12 Aug 5 16:51 file1
-rw-r--r-- 2 root root 12 Aug 5 16:50 file2

En este caso file1 y file2 tienen 2 enlaces.

Creando ahora enlaces simbólicos
En los enlaces simbólicos el numero de inodo no será el mismo, simplemente se crea un enlace imagen o simbólico.

Ejemplo:
ln –s file1 file2
Creamos un enlace simbólico llamado file2 que funciona de imagen y apunta directamente a file1. Si usamos ls –i, veremos que los dos ficheros tienen inodos diferentes.
Usando ls –l, veremos que file2 apunta con el signo “->” a file2.

Buscando patrones en ficheros

Comando: grep

Sintaxis: grep [-cinvw] <patron> <file1> <file2> <file3> ...

Donde patron es una expresión regular, lo que se va a buscar y file1/2/3 son los ficheros donde se procesara esa búsqueda

Parámetros:

- **-i** Con este argumento el comando grep no distinguirá entre nomenclaturas (mayúsculas y minúsculas).
- **-v** Error de salida
- **-c** Cuenta los ficheros donde se da el patrón.
- **-n** Nos muestra el numero de línea donde se ha localizado el patrón
- **-w** Muestra las líneas que contienen el patrón como palabra completa, no como cadena de una palabra mayor.

Cambiando la presentación en pantalla

Comando: pr

Sintaxis: pr –dl[n] file

Donde fichero es el nombre que queremos para cambiar su presentación en pantalla.

Comandos para operaciones con procesos

Cada vez que usted ejecuta un programa, lo que esta haciendo para el ordenador internamente es lanzando lo que se conoce como un “proceso” que no es mas que el nombre que recive un programa en tiempo de ejecución (cuando se esta ejecutando).

Listando procesos

Comando: ps

Sintaxis: ps [-feautd]

Este comando visualiza la lista de procesos que se estan ejecutando actualmente.

Parametros:

- **-f** Información completa. UID (identificador de user), nos dice quien inicio el proceso y stime hora del lazamiento del proceso.
- **-e** Muestra todos los procesos que se están ejecutando, no solo los que lanzamos nosotros. Por tanto también se muestran los llamados demonios (procesos que se ejecutan en segundo plano). También se muestran los procesos llamados “lideres de grupo de proceso” estos son procesos que solo sirven para que se ejecuten otros procesos.
- **-d** igual que el anterior pero no muestra los lideres de grupo.
- **-a** Solo lista procesos relacionados con el terminal
- **-u** Visualiza que es lo que hace un usuario
- **-t** visualiza que es lo que hace un terminal

Ejemplo:

```
ps
PID TT STAT TIME COMMAND
24  3  S   0:03  (bash)
161 3  R   0:00    ps
```

Haber, la información que nos muestra es esta. Donde PID (identificador de proceso)
Cada proceso es identificado con un numero y el sistema lo reconoce como tal.
En la columna command se encuentra el nombre del proceso que se esta ejecutando.
Bash, es el shell o interprete de comandos usado por linux.
Lógicamente el bash se ejecutara siempre concurrentemente con el comando que lancemos.

Chequeando el estado de un proceso

Comando: jobs

Sintaxis: jobs [-lrs]

Parámetros

-l Visualiza el PID (identificador de proceso) de cada proceso.
-r Visualiza los procesos que se encuentran en estado de ejecución (running)
-s Visualiza los procesos que están en estado suspendido (stopped).

Eliminando procesos

Comando: kill

Etimología: kill (matar, eliminar, asesinar)

Sintaxis: kill [señal] [n]

Este comando toma como argumento un numero de tarea o un PID (process identifier).
En el caso de que recurramos a no pasarle el PID y si la tarea debemos anteponer al numero de la tarea el signo del tanto-porcinito “%”.

Ejemplo: kill %1
Eliminamos la tarea numero 1
Ejemplo: kill 156
Eliminaremos el proceso con el PID numero 156.

Parada y relanzamiento de tareas

Hay otra manera de poner una tarea en segundo plano. Usted puede lanzarlo como un proceso normal (en primer plano), pararlo, y después relanzarlo en segundo plano.
Primero, lance el proceso yes en primer plano como lo haría normalmente:
/home/larry# yes > /dev/null
De nuevo, dado que yes corre en primer plano, no debe retornar el prompt de la shell. Ahora, en vez de interrumpir la tarea con |_ctrl-C_|, suspenderemos la tarea. El suspender una tarea no la mata: solamente la detiene temporalmente hasta_que_Ud. la retoma. Para hacer esto usted debe pulsar la tecla de suspender, que suele ser |_ctrl-Z_|.
/home/larry#_yes > /dev/null
|_ctrl-Z_|
[1]+ Stopped yes >/dev/null
/home/larry#

Mientras el proceso está suspendido, simplemente no se esta ejecutando. No gasta tiempo de CPU en la tarea. Sin embargo, usted puede retomar el proceso de nuevo como si nada hubiera pasado. Continuará ejecutandose donde se dejó. Para relanzar la tarea en primer plano, use el comando fg (del inglés "foreground").
/home/larry# fg
yes >/dev/null
La shell muestra el nombre del comando de nuevo, de forma que_tenga_conocimiento_de que tarea es la que ha puesto en primer plano. Pare la tarea de nuevo, con |_ctrl-Z_|. Esta vez utilice el comando bg para poner la tarea en segundo plano. Esto hará que el comando siga ejecutandose igual que si lo hubiese hecho desde el principio con "&" como en la sección anterior.
/home/larry# bg
[1]+ yes >/dev/null &
/home/larry#
Y tenemos de nuevo el prompt. El comando jobs debería decirnos que yes se está ejecutando, y podemos matar la tarea con kill tal y como lo hicimos antes. >Cómo podemos parar la tarea de nuevo? Si pulsa |_ctrl-Z_|no funcionará, ya que el proceso está en segundo plano. La respuesta es poner el proceso en primer plano de nuevo, con el comando fg, y entonces pararlo. Como puede observar podrá usar fg tanto con tareas detenidas, como con las que estén segundo plano. Hay una gran diferencia entre una tarea que se encuentra en segundo plano, y una que se encuentra detenida. Una tarea detenida es una tarea que no se está ejecutando, es decir, que no usa tiempo de CPU, y que no está haciendo ningún trabajo (la tarea aun ocupa un lugar en memoria, aunque puede ser volcada a disco). Una tarea en segundo plano, se está ejecutando, y usando memoria, a la vez que completando alguna acción mientras usted hace otro trabajo. Sin embargo, una terea en segundo plano puede intentar mostrar texto en su terminal, lo quepuede resultar molesto si está intentando hacer otra cosa. Por ejemplo, si usted usó el comando
/home/larry# yes &
sin redirigir stdout a /dev/null, una cadena_de_y-es_se mostrarán en su monitor, sin modo alguno de interrumpirlo (no puede hacer uso de |_ctrl-C_|para interrumpir tareas en segundo plano). Para poder parar esas interminables_y-es,_ tendría que usar el comando fg para pasar la tarea a primer plano, y entonces usar |_ctrl-C_|para matarla. Otra observación. Normalmente, los comandos "fg" y "bg" actuan sobre el último proceso parado (indicado por un "+" junto al número de tarea cuando usa el comando jobs). Si usted tiene varios procesos corriendo a la vez, podrá mandar a primer o segundo plano una tarea especifica indicando el ID de tarea como argumento de fg o bg, como en
/home/larry# fg %2
(para la tarea de primer plano número 2), o
/home/larry# bg %3
(para la tarea de segundo plano número 3). No se pueden usar los ID de proceso con fg o bg.
Además de esto,si usa el número de tarea por si solo, como
/home/larry# %2
es equivalente a
/home/larry# fg %2
Solo recordarle que el uso de control de tareas es una utilidad de la shell. Los comandos fg, bg y jobs son internos de la shell. Si por algún motivo usted utiliza una shell que no soporta control de tareas, no espere disponer de estos comandos. Y además, hay algunos aspectos del control de tareas que difieren entre Bash y Tcsh. De hecho, algunas shells no proporcionan ningún control de tareas_sin embargo, la mayoría de las shells disponibles para Linux soportan control de tareas.

Usando el editor vi

Un editor de texto es simplemente un programa usado para la edición de ficherosque contienen texto, como una carta, un programa en C, o un fichero de configuración del sistema. Mientras que hay muchos editores de texto disponibles en Linux, el único editor que está garantizado encontrar en cualquier sistema UNIX es vi_el "visual editor". vi no es el editor más fácil de usar, ni es muy autoexplicativo. De cualquier forma, como es tan común en el mundo UNIX y es posible que alguna vez necesite usarlo, aquí encontrará algo de documentación.
La elección de un editor es principalmente una cuestión de gusto personal y estilo. Muchos usuarios prefieren el barroco, autoexplicativo y potente Emacs_un editor con más características que cualquier otro programa único en el mundo UNIX. Por ejemplo, Emacs tiene integrado su propio dialecto del lenguaje de programación LISP y tiene muchas extensiones (una de ellas es el programa "Eliza"- como programa de IA). Pero como Emacs y todos sus ficheros de soporte es relativamente grande, puede que no tenga acceso a él en muchos sistemas. vi, por otra parte, es pequeño y potente, pero más difícil de usar. De cualquier modo, una vez conozca la forma de funcionamiento de vi, es muy fácil usarlo. Simplemente la curva de aprendizaje es bastante pronunciada al comienzo. Esta sección es una introducción coherente a vi_no discutiremos todas sus características, solo aquellas necesarias para que sepa como comenzar.

Conceptos

Mientras se usa vi, en cualquier momento estará en uno de tres posibles modos de operación. Estos modos son conocidos como modo órdenes, modo inserción y modo última línea. Cuando inicia vi, está en el modo órdenes. Este modo le permite usar ciertas órdenes para editar ficheros o cambiar a otros modos. Por ejemplo, tecleando "x" mientras está en el modo órdenes, borra el carácter que hay debajo del cursor. Las teclas del cursor mueven este por el fichero que estamos editando. Generalmente, las órdenes usadas en este modo son solo de uno o dos caracteres de longitud. Habitualmente insertará o editará texto desde el modo inserción. Usando vi, probablemente dedicará la mayor parte del tiempo en este modo. Inicia el modo de inserción al usar una orden como "i" (para "insertar") desde el modo de órdenes. Una vez en el modo de inserción, irá insertando texto en el documento desde_la posición actual del cursor. Para salir del modo de inserción y volver al de órdenes, pulse |_esc_|. Modo última línea es un modo especial usado para proporcionar ciertas órdenes extendidas a vi. Al usar esos comandos, aparecen en la última línea de la pantalla (de ahí el nombre). Por ejemplo, cuando teclea ":" desde el modo de ódenes, entrará en el modo última línea, y podrá usar órdenes como "wq" (para escribir el fichero a disco y salir de vi), o "q!" (para salir de vi sin guardar los cambios). El modo de última línea es habitualmente usado por órdenes vi mayores_de_un carácter. En el modo de última línea, introduce una orden de una sola línea y pulsa |_enter_|para ejecutarla.

La sintaxis de vi

vi <filename>
donde <filename> es el nombre del fichero que desea editar.

Ordenes en el vi
Hay muchas formas de insertar texto aparte de la orden “i”.
a >> Inserta texto comenzando detras de la posición actual.
i >> Añade texto delante del cursor
A >> Añade texto al final de la línea actual
I >> Añade texto al principio de la línea actual
o >> Inserta una línea delante de la actual y pasa a modo entrada
O >> Inserta una línea detrás de la línea actual y pasa a modo entrada.

En modo ultima línea se usan las ordenes que empiezan por “:”
:q! >> Salir sin grabar
:wq >> Salir grabando
:w >> Grabar sin salir del vi
:/lamer >> Busca la palabra lamer.
:g /casa /s/hogar/g Sustetuye la palabra casa por hogar en todo el fichero.
:el >> Si cometes muchos errores y quieres el fichero según se edito al inicio.
:. Repite la ultima orden
:u Deshace el ultimo cambio
:U Deshace todos los cambios de la línea actual
:n va a la línea n
:r! <patron> Este parámetro lee la salida del comando o fichero (patrón) que le damos. Si patrón es un fichero, se incluye el contenido de ese otro fichero en el que estamos editando. Si patrón es un comando, inserta en el fichero que estamos editando el resultado del proceso de ese comando.
El signo “!” le dice a vi que realmente usted quiere editar ese fichero sin salvar los cambios del primero.

Comandos para operaciones con temporizadores

Ejecutando temporizadores

Comando: at
Sintaxis: at hora [fecha] <intro> comandos <control+d>
Ejemplo:

at 16:45 <intro>
at > ls –l > directorio
at > control +d

Eliminando temporizadores

Comando: atrm
Sintaxis: atrm <ID>
Borra el trabajo con ese identificador asignado al lanzar el at

Visualizando temporizadores

Comando: atq
Muestra los trabajos en la cola para ser ejecutados

Xwindow

Sistema xwindow: Es prácticamente el estándar para entornos graficos de usuarios en linux, también se le ha denominado x11.

XFree86: Es una implementacion libre del servidor x para sistemas unix a base de PC incorporado en linux.

Linux cuenta con xfree86config que sirve para la configuración: del entorno grafico, raton, etc.
Las distintas distribuciones de linux implementan sistemas de configuración mas amigable en modo grafico no texto, es el caso de SUSE con SAX.

Sax2 y xf86config y sax: permiten la configuración general del sistema xwindows.
El sax2 para configuración de xfree86 v4.0 en modo grafico.
Sax para xfree86 v6.3.x en modo grafico.
XF86 para cualquier versión en modo texto y sin raton por lo que se aconseja instalar con yast el paquete de SAX.

La estructura del sistema xwindow esta organizada en capas separadas:

- 1.- SO: la forma el sistema operativo. Se encarga de tareas basicas unidas al hardware como puede ser: administrar la memoria.
- 2.- X-server o servidor x: Se encarga de tareas de acceso a la tarjeta grafica de dibujo: para líneas, textos, circulos, rectangulos, etc.Y distribuye estos servicios al ordenador local a través de la red. Permite por ejemplo que se ejecute un programa en el ordenador servidor y que se visualice los resultados en los demas.
- 3.- Windows manager o gestor de ventanas: Se encarga de gestionar tamaños, colores, efectos 3d, barras de desplazamiento, botones, maximizar, minimizar, abrir, cerrar, obtención en 1º plano y segundo plano, etc.

Herramienta SAX

Herramienta grafica paro la configuración del teclado, monitor, tarjeta grafica, etc.
Una vez que arrancamos linux y entramos en el escritorio podemos pasar a una consola de texto pulsando ctrl.+alt+f1.
Una vez ahí se arranca la utilidad sax, teclando sax desde la linea de comandos.
Si no conoces las caracteristicas de la tarjeta grafica estas se pueden averiguar con los comandos “/sbin/lspci” o “superprobe”.
Otra forma de arrancar la herramienta sax es a través de yast, entrando en la administración del sistema y seleccionando la operación configurar xfree86.
Antes de modificar la configuración se aconseja realizar una copia del fichero “/etc/xf86config”, para sobrescribirlo en caso de meter la pata.

RAM DAC: Ramdon access memory digital analogic conversor: Es un microchip situado en la tarjeta que trnaslada la representación digital en señal analogica que el monitor puede visualizar y se mide en megahercios (mhz).