COMANDOS EN MODO TEXTO

GENERALIDADES

NO dan igual mayúsculas y minúsculas.

No vamos a ver más que unos pocos comandos y de ellos no veremos todas las posibilidades. Para conocer todas las posibilidades, teclear `man` y el nombre del comando. Ejemplo: `man pwd`

En general los comandos tienen un nombre que se teclea lo primero y luego pueden ir algunas opciones (llevan un – delante) y en muchos casos unos nombres de ficheros sobre los que actúan esos comandos. El comando puede dar una salida o mensaje, normal o de error.

Los términos *carpeta* y *directorio* son sinónimos, así como *archivo* y *fichero*.

Nombres de ficheros

Los nombres de ficheros pueden darse de varias formas:

- directamente un nombre de un fichero en este directorio.
- Un nombre de archivo con la ruta de directorios en la que está. En Linux el separador entre directorios y subdirectorios o archivos es / Ejemplo: el fichero saludo está en el directorio etc que cuelga de la raíz (directorio base general) del sistema. Sería: `/etc/saludo` Se puede usar ~ para referirse a la carpeta del usuario.
- Un patrón de nombre. Se pone sólo una parte; para lo que no se pone tenemos dos indicadores (comodines):
  - * quiere decir cualquier secuencia de caracteres
  - ? quiere decir un carácter

    Por ejemplo `*.sh` son todos los archivos cuyo nombre acabe en `.sh`

    Otro ejemplo: `macro?` son todos los archivos cuyo nombre sea `macro` y una letra
- Puede haber combinaciones. Ejemplo: `/var/config*`

Entrada y salida del comando

La salida del comando normalmente es a pantalla, pero esto podemos cambiarlo. Tenemos las siguientes posibilidades:

- Enganchar la salida de un comando con la entrada del siguiente, usando | (se le llama encadenamiento). Ejemplo: `ls | fgrep sh`
- Mandar la salida normal a un fichero, poniendo > Ejemplo: `cat macro1.sh macro.sh >principio`
- Añadir la salida normal al final de un fichero, poniendo >>

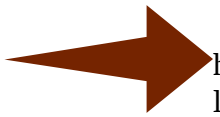
  Ejemplo: catr `macro.sh >>principios`
- Mandar la salida de error a un fichero, poniendo `2>`

  Ejemplo: `mv datos.pdf /mnt/usb2 2>error.log`

Ayuda mientras tecleamos un comando

Si damos a la flecha hacia arriba nos sale el comando anterior, si seguimos dando nos van saliendo anteriores. Podemos movernos por ellos con las flechas, borrar una parte, añadir, etc.

Si cuando estamos tecleando el nombre de un fichero damos al tabulador nos lo completa hasta donde pueda. Si hay más de uno que empiece por lo que llevamos, no lo terminará, sino que pondrá la parte común y esperará que avancemos nosotros. Si le volvemos a dar al tabulador nos sacará la lista de posibilidades.



Además tenemos el comando `history` que nos sacará todos los comandos que hayamos tecleado; como es una lista larga, puede ser conveniente ir viéndola pantalla a pantalla con `less`.

El comando `fc` nos permite editar alguno para reejecutarlo. Por ejemplo `fc -s cat` nos dejaría editar el último comando en que hayamos puesto `cat` y volverlo a ejecutar.

Ejecución de fondo

Si un comando puede tardar y queremos seguir en la misma consola tecleando otros mientras ése termina, le pondremos & al final, para que lo ejecute aparte. Ejemplo: `gedit &` Si no lo hicimos, pero vemos que tarda y queremos ponerlo en pausa, daremos Ctrl+Z Si queremos cancelarlo daremos Ctrl+C

Ejecución en modo Administrador

Algunos comandos necesitan permisos de administrador para ejecutarse. En ese caso, si los ejecutamos sin más, normalmente nos dará error. Para ejecutar el comando en modo administrador, lo que hay que hacer es poner delante el comando `sudo`, por ejemplo:

```
sudo mv fich1 fichero1
```

Antes de ejecutar el comando indicado nos pedirá la contraseña.

DIRECTORIOS

Cualquier nombre de fichero en que no indiquemos directorio se supone que está en un directorio por defecto que se llama directorio de trabajo o actual. Al principio será nuestra carpeta personal. Es habitual que aparezca en el mensaje que nos saca la consola para esperar nuestras órdenes.

cd

El comando `cd` nos permite cambiarlo, indicándole cuál queremos que sea. Si ponemos – va al anterior y si ponemos .. se entiende la carpeta que contiene a la actual.

Ejemplos:

- `cd ../prueba1/datos` al subdirectorio `prueba1` del directorio padre del actual y dentro de él al subdirectorio `datos`
- `cd /home/usuario/Documentos` indica la ruta completa desde la base

FICHEROS

Información

ls

Para ver la información de los ficheros de un directorio tenemos el comando `ls`. Ejemplos:

- `ls` muestra los ficheros del directorio actual
- `ls datos` muestra, o los ficheros del subdirectorio `datos`, o información del fichero `datos`

Cambios

mv

Para mover un fichero de un directorio a otro tenemos el comando `mv`

Ejemplo: `mv pruebas/fich1 ..` coge el fichero `fich1` del subdirectorio `pruebas` y lo pasa al directorio padre del actual.

También se puede con él cambiar el nombre del fichero, a la vez que se mueve o sin moverlo. Ejemplo: `mv fich1 fichero1` cambia el nombre de `fich1` a `fichero1`

cp

Cuando en vez de `mv` usamos `cp` hacemos una copia. Ejemplo: `cp *.c fuentes` copia todos los ficheros que acaben en `.c` al subdirectorio `fuentes`.

rm

Con `rm` podemos borrar. Ejemplo: `rm /var/*` borra todos los ficheros del directorio `var`, que cuelga del base

ln

En Linux un fichero puede estar en más de un directorio y no necesariamente con el mismo nombre. Hay dos posibilidades: una duplicidad simbólica y una real.

La real consiste en que hay un único fichero, aunque podemos operar con él desde dos directorios diferentes; cualquier operación que hagamos afecta al fichero realmente.

La “simbólica” es que el fichero original sigue en su sitio y lo que hay en la otra ubicación es un fichero especial que señala al original; de esta forma, si, por ejemplo, borramos el segundo, no hemos borrado el fichero original, sino el señalador.

El comando para hacer duplicaciones es `ln`; si lo usamos a secas hace una real y si usamos la opción `-s` hace una simbólica. Primero se pone el fichero original y luego el duplicado. Ejemplo: `ln -s Np/np np` hace una duplicación simbólica del fichero `np` que está en el subdirectorio `Np` a este directorio, llamándose en él también `np`

chmod

Cada fichero lleva asociados unos permisos: para leerlo, para modificarlo y para ejecutarlo (lo que quiere decir que será un programa o una macro). Se pueden indicar permisos diferentes para el dueño del fichero, para los de su grupo (si es que se ha formado un grupo de usuarios) y para el res-

to. El comando para cambiar permisos es `chmod`. La forma más simple de usarlo es añadir + ó – según que queramos poner o quitar un permiso y luego la letra `r`, `w` ó `x`, respectivamente, según qué permiso queramos cambiar. Ejemplo: `chmod +x macro.sh` da permiso de ejecución para todo el mundo para el fichero `macro.sh`

tar

Para archivar ficheros hablaremos del comando `tar`. Opcionalmente también lo puede comprimir. Aquí comentaremos sus usos más habituales, a través de ejemplos:

`tar -czf archivo.tgz fichero1 fichero2` Esto crearía el archivo **comprimido** *archivo.tgz* (es conveniente esta extensión, para que luego se interprete correctamente) con los ficheros *fichero1* y *fichero2*

`tar -cf archivo.tar fichero1 fichero2` Esto crearía el archivo **no comprimido** *archivo.tar* con los ficheros *fichero1* y *fichero2*

`tar -tvzf arch.tgz` Esto lista los contenidos del archivo comprimido *arch.tgz S*i no fuese comprimido, la extensión del archivo sería *tar* y no estaría la opción `z`

`tar -xzf paq.tgz` Esto extrae los contenidos del archivo comprimido *paq.tgz* Nuevamente, si no estuviese comprimido la extensión sería *tar* y no usaríamos la opción `z,` es decir, que quedaría `tar -xf paq.tar`

Si el archivo no está comprimido, se puede modificar. Si está comprimido, no es posible la modificación, sino que habría que extraer el contenido, quitar o poner los archivos que se quieran y volver a realizar el archivado.

`tar -rf arc1.tar fich*` Esto añade a *arc1.tar* todos los ficheros que empiecen por *fich*

`tar -uf arch.tar fich?` Esto añade a *arch.tar* los ficheros *fich*+una letra sólo si no están o si el que está es más viejo que el que le pasamos

`tar --delete -f envio.tar doc.txt` Esto quita de *envio.tar* el fichero *doc.txt*

VISUALIZACIÓN

cat

El comando `cat` nos saca un listado sin parada de un fichero. Cómo esto es inconveniente si el fichero es largo, se usa más para pegar un archivo al final de otro.

Ejemplo: `cat calen2010 >>calen` agrega el contenido de `calen2010` al final de `calen`

less

Éste es el comando más cómodo para ver un fichero pantalla a pantalla en la consola. Nos permite movernos con las teclas de flechas y de página.

Ejemplo: `less calen2010`

sort

El comando `sort` nos saca las líneas de un fichero, no por el orden en el que están, sino por orden alfabético, con diferencia entre mayúsculas y minúsculas. Este criterio se puede cambiar con las siguientes opciones:

- `-f`: no hacer diferencia entre mayúsculas y minúsculas
- `-n`: orden numérico

- `-r`: por orden inverso
- `-k`: marcar la posición a mirar para el orden (cuando no queremos el principio de la línea)

  Ejemplo: `sort -n serie >serieord` pone en `seriord` los contenidos de `serie` por orden numérico

fgrep

Para buscar textos en fichero, veremos el comando más simple, que es `fgrep` Hay que indicarle el texto a buscar y los ficheros o directorios en qué tiene que mirar. Nos saca las líneas que tienen el texto buscado. Ejemplo: `fgrep bin ~/*` busca las líneas que tengan *bin* en todos los ficheros del directorio del usuario.

PROCESOS

ps

El sistema tiene en todo momento una serie de procesos (entorno gráfico, controladores, correo, etc.) aparte de los propios programas que el usuario lance. Para ver los procesos activos está el comando `ps` Tiene diversas opciones, pero vamos a comentar sólo dos. Sin opciones saca información resumida de los programas que ha lanzado el usuario. Con la opción `-e` saca todos los procesos que haya y con la opción `-f` amplia la información que da.

kill

Normalmente cuando miramos los procesos es porque algo anda mal y, con frecuencia, queremos interrumpir un proceso. Entre la información que nos saca `ps` hay una columna llamada `PID` que es el número del proceso. Para cancelar un proceso conocido su número usamos el comando `kill` Ejemplo: `kill 2212` termina el proceso 2212

dmesg

En ocasiones, para resolver algún comportamiento del sistema, necesitamos averiguar todos los mensajes que ha sacado en el sistema al funcionar, tanto los simplemente informativos, como cualquier otro aviso o error. Para ello podemos usar el comando `dmesg`. Dado que el listado de avisos del sistema suele ser muy largo, ya que hay montones de ellos informativos, y no sabemos cuál o cuáles son los que estamos buscando, es común combinar este comando con el comando `less`