Práctica 1: El Sistema Operativo UNIX

Tabla de Contenidos

1 Introducción

En esta práctica se explican de manera resumida el conjunto de órdenes más utilizadas en la realización de las prácticas y el desarrollo del trabajo de curso.

Aunque las explicaciones de los comandos se refieren al sistema opeativo SunOS, que es el que utiliza la máquina del laboratorio, casi en su totalidad son aplicables a cualquier sistema operativo del tipo UNIX, y en particular al Linux.

1.1 Formato General de las Órdenes

Cuando se ha establecido una sesión y aparece el *prompt* (normalmente el símbolo **$** o el símbolo **>**) significa que el procesador de órdenes está esperando que se le den órdenes. El procesador de órdenes (también llamado *shell* o *intérprete de comandos*) es el programa del sistema operativo que se encarga de analizar las órdenes que se dan en una sesión y ejecutarlas. La línea donde se introducen las órdenes se denomina normalmente *línea de órdenes*.

El formato general de las órdenes es:

```
nombre_de_orden [opciones] [argumentos]
```

Los corchetes '**[**' y '**]**' no hay que escribirlos, lo que quieren decir es que lo que va dentro es opcional. Se recuerda que hay que pulsar la tecla **INTRO** al final para que el intérprete de comandos comience a ejecutar la orden. Las opciones y los argumentos dependerán de cada orden, pero las opciones van precedidas normalmente del signo menos '**–**'. La orden, las opciones y los parámetros van separados unos de otros por una o más espacios en blanco. Algunos ejemplos de órdenes (que se verán más adelante) son `ls`, `passwd` y `who`.

1.2 Ayuda en línea

Dentro de una sesión se puede consultar el modo de utilización de cualquier comando mediante la orden **man**. Su formato es:

```
man nombre_de_la_orden
```

Por ejemplo, para conocer cómo se utiliza y qué opciones tiene el comando `pwd`, escribiremos:

```
man pwd
```

Para saber cómo se usa `man` se puede hacer:

```
man man
```

2 Estructura de Directorios y el Sistema de Ficheros

2.1 La Estructura de Directorios

Los ficheros sirven para guardar información, y normalmente están almacenados en disco magnético. Todo el sistema UNIX está basado en ficheros (por ejemplo las órdenes están en ficheros, y los propios directorios son ficheros). Cada fichero tiene un nombre, un contenido, un lugar para almacenarlo y cierta información de tipo administrativo; por ejemplo quién es el dueño y cuál es su tamaño.

El sistema de ficheros UNIX está organizado en directorios con una estructura jerárquica. Existe un directorio básico llamado directorio raíz y que se especifica mediante el carácter `/`. Dentro de éste puede haber subdirectorios del raíz y dentro de éstos puede haber a su vez otros subdirectorios hasta todos los niveles que se quiera. Para separar los niveles también se usa el carácter `/`.

El nombre del fichero puede tener más de 8 caracteres y normalmente está limitado a 255 (depende de la máquina). No se deben usar caracteres como el guión y el espacio. Se suele intercalar el carácter punto dentro del nombre, y lo que está detrás de él se le llama extensión.

Dentro de un directorio NO pueden coexistir dos ficheros con el mismo nombre, aunque sí lo podrían hacer en directorios distintos. Por otra parte, los nombres de ficheros discriminan entre las mayúsculas y las minúsculas, de manera que `integrales.c` es distinto de `Integrales.c`, `INTEGRALES.C`, `integrales.C`, etc.

Cada usuario tiene un directorio de inicio de sesión, que en el caso de los alumnos del curso 2005/2006 es un subdirectorio de `/alumnos/cuentas` con el mismo nombre del usuario.

Al iniciar la sesión, el directorio de trabajo (o directorio actual) del procesador de órdenes es el directorio de inicio de sesión. El directorio de trabajo se puede cambiar con la orden `cd` (change directory), pero el directorio de inicio de sesión será siempre el mismo. La orden `pwd` muestra en pantalla el directorio de trabajo (*Print Working Directory*).

Para localizar un fichero dentro de una estructura de directorios, podemos utilizar cualquiera de los tres métodos siguientes:

- especificando un camino absoluto desde el directorio raíz. Para especificarlo de esta forma se empieza el camino con **`/`** indicando el directorio raíz, y se continúa con todos los directorios que hay por debajo de él hasta llegar al fichero que se quiere especificar, todos ellos separados por `/`.
- especificando un camino relativo desde el directorio de inicio de sesión. En este caso, la especificación del directorio comienza por **`~/`**, seguida de la ruta desde el directorio de inicio de sesión hasta el fichero deseado.
- especificando un camino relativo desde el directorio de trabajo. El directorio de trabajo se puede especificar con un punto (**`.`**). Para subir al nivel superior en la jerarquía de directorios se utilizan los dos puntos (**`..`**). Si el camino comienza con un nombre de directorio (sin `/` ni `~/`) se considera que el camino comienza en el directorio de trabajo.

Supongamos por ejemplo que una parte de la estructura de directorios fuera similar a la siguiente:

```
                  /
                  |
  +-------------------------+----- . . .
  |       |      |          |
 bin     dev    lib      alumnos   . . .
                            |
                         cuentas
           +--------+-----+------+--------+---- . . .
           |        |     |      |        |
        antonio   jesus jose  alfredo  carmen   . . .
           |
    +-------------+
  fuentes     ejecutables
     |
 integrales.c
```

Si el directorio de inicio de sesión es `/alumnos/cuentas/antonio`, y el directorio de trabajo actual es `/alumnos/cuentas/antonio/ejecutables`, podemos hacer referencia al fichero `integrales.c` de las tres formas siguientes:

```
/alumnos/cuentas/antonio/fuentes/integrales.c
~/fuentes/integrales.c
../fuentes/integrales.c
```

En general, esta forma de especificar un camino es válida para cualquier orden de UNIX que necesite un nombre de directorio o fichero.

2.2 Operaciones sobre Directorios

2.2.1 Orden `pwd`

Muestra el directorio de trabajo. Su formato es:

```
pwd
```

2.2.2 Orden `cd`

Cambia el directorio de trabajo. Su formato es:

```
cd [directorio]
```

Si se utiliza `cd` sin nombre de directorio, vuelve al directorio de inicio de sesión.

El directorio al que se desea cambiar puede especificarse con cualquiera de los tres métodos vistos anteriormente:

- Mediante su ruta absoluta desde el directorio raiz.
- Mediante su ruta relativa al directorio de trabajo.
- Mediante su ruta relativa al directorio de inicio de sesión.

Realice los siguientes ejemplos y compruebe los resultados:

`cd /usr`	cambia al directorio `/usr`
`pwd`	muestra el directorio de trabajo
`cd`	cambia al directorio de inicio de sesión
`pwd`	muestra el directorio de trabajo
`cd ../../`	cambia al directorio que se encuentra en el

	directorio que está dos niveles por encima del directorio de trabajo dentro del árbol de directorios. Suponiendo que el directorio de trabajo es `/alumnos/cuentas/fp_lab`, con el primer `..` nos referimos al directorio `cuentas` (padre del directorio de trabajo) y con el siguiente `..` nos referimos a `alumnos` (padre del directorio `cuentas`)
`pwd`	muestra el directorio de trabajo
`cd`	nos sitúa en el directorio de inicio de sesión
`cd bin`	cambia al directorio `bin` que está dentro del directorio de inicio de sesión. Si no existe da un mensaje de error.
`pwd`	muestra el directorio de trabajo

2.2.3 Orden `ls`

Muestra el contenido de un directorio. El formato de la orden es:

```
ls [opciones] [nombre]
```

Si no se le da ningún argumento, `ls` muestra los nombres de los ficheros del directorio de trabajo. Si se le da como argumento un nombre de fichero, muestra por pantalla el nombre del fichero si existe en el directorio de trabajo. Si se le da como argumento un nombre de directorio, muestra el nombre de todos los ficheros de ese directorio.

Opciones:

`-l`	formato largo. Muestra el modo de acceso (permisos), número de enlaces (que no se estudian en estas prácticas), propietario, tamaño, fecha de modificación y nombre de cada fichero.
`-a`	muestra también los ficheros que empiezan por punto, que sin esta opción no se muestran.
`-t`	los ficheros mostrados aparecen ordenados por fecha. Por defecto es la fecha de última modificación (`mtime`).
`-u`	se usa con la opción `-l` (para imprimir) o `-t` (para ordenar) (`-lu` o `-tu`). Los ficheros mostrados con `-tu` aparecen ordenados por fecha de último acceso (`atime`). Con la opción `-lu` la fecha que aparece es `atime` (pero no aparecen ordenados). Un ejemplo de acceso a un fichero es cuando éste se muestra por pantalla. Se puede utlizar `-ltu` para mostrar y ordenar usando `atime`.
`-c`	igual que la opción `-u` pero con la fecha del último cambio de estado (`ctime`). Un ejemplo de cambio de estado es cuando se cambia algún permiso del fichero.
`-d`	muestra el nombre del directorio y no su contenido. Sirve para ver información acerca del directorio.

`-r`	invierte el orden de salida.
`-R`	muestra recursivamente los directorios. Muestra los ficheros del directorio y también los de todos sus subdirectorios.
`-F`	indica con un carácter al lado del nombre del fichero su naturaleza, con `/` indica los que son directorios y con `*` los que son ejecutables.

Ejemplos para probar:

`cd`	nos sitúa en el directorio de inicio de sesión
`ls`	muestra los ficheros del directorio de trabajo
`ls /dev`	muestra los ficheros del directorio `/dev`
`ls -l /usr/lib`	muestra en formato largo.
`ls -a`	muestra también los ficheros que empiezan por punto
`ls -R /etc`	muestra todos los ficheros y todos los subdirectorios que están en el directorio `/etc`
`ls -ld bin`	muestra no el contenido de `bin`, sino el propio directorio. Ver la diferencia con `ls -l bin`
`ls -F`	muestra el contenido del directorio de trabajo y da información acerca de la naturaleza de los ficheros
`ls -ltc`	muestra los ficheros ordenados por fecha de último cambio de estado
`ls -ltrc`	igual que el ejemplo anterior pero con el orden de salida invertido

2.2.4 Orden `mkdir`

Crea un nuevo directorio. El formato de la orden es:

```
mkdir [opciones] nuevo_directorio
```

Aunque a esta orden se le puedan dar opciones, no se explican en la práctica ya que el uso más común de la orden `mkdir` es sin opciones. Para información acerca de las opciones se puede ver

```
man mkdir
```

Realice la siguiente secuencia de comandos y compruebe el resultado:

`cd`	nos sitúa en el directorio de inicio de sesión
`mkdir bin`	crea el directorio `bin` dentro del directorio de trabajo
`mkdir dirnuevo`	crea el directorio `dirnuevo` dentro del directorio de trabajo
`cd dirnuevo`	cambia al directorio recién creado
`mkdir dirnuevo2`	crea `dirnuevo2` dentro de `dirnuevo`

`mkdir ../bin/otrobin`	crea el directorio `otrobin` dentro del directorio `bin`
`cd`	nos sitúa en el directorio de inicio de sesión
`mkdir bin`	crea el directorio `bin` dentro del directorio de trabajo
`mkdir dirnuevo`	crea el directorio `dirnuevo` dentro del directorio de trabajo
`cd dirnuevo`	cambia al directorio recién creado
`mkdir dirnuevo2`	crea `dirnuevo2` dentro de `dirnuevo`
`mkdir ../bin/otrobin`	crea el directorio `otrobin` dentro del directorio `bin`

2.2.5 Orden `rmdir`

Borra un directorio. Su formato es:

```
rmdir directorio
```

El directorio a borrar no puede contener ningún fichero ni ningún otro directorio, es decir, tiene que estar vacío (más adelante se verá el comando que permite borrar ficheros). El directorio a borrar no puede ser el directorio de trabajo.

Ejemplos para probar:

`cd`	nos sitúa en el directorio de inicio de sesión
`rmdir dirnuevo`	no borra el directorio `dirnuevo` debido a que no está vacío
`cd dirnuevo`	cambia al directorio `dirnuevo`
`rmdir dirnuevo2`	borra `dirnuevo2`
`rmdir ../bin/otrobin`	borra `otrobin` dentro de `bin`

2.3 Permisos

Con la orden `ls -l` se muestra al comienzo de cada línea un conjunto de diez caracteres asociado a cada fichero:

- El primer carácter indica el tipo de fichero: **`-`** si se trata de un fichero normal, **`d`** si se trata de un directorio, y **`l`** si es un enlace.
- Los restantes 9 caracteres indican por grupos de 3 los permisos asociados al dueño o usuario del fichero, al grupo al que pertenece el fichero (sin incluir el dueño) y al resto de los usuarios respectivamente. En UNIX los usuarios pertenecen a grupos y para ver a qué grupo se pertenece existe la orden groups. Los ficheros pertenecen a un usuario y a un grupo. Para ver el usuario y el grupo al que pertenecen los ficheros, usar `ls -lg`.

De estos 3 caracteres el primero representa el permiso de lectura, puede ser `r` o `-`, el segundo representa el permiso de escritura, y puede ser `w` o `-`, y el tercero representa el permiso de ejecución (si se trata de un fichero) o de búsqueda (si se trata de un directorio), y puede valer `x` o `–`.

El carácter `r` indica permiso de lectura, w el de escritura y x el de ejecución. Si se trata de un directorio, r indica permiso de ver los nombres de los ficheros, w para crear y borrar ficheros (es necesario además el permiso x) y x buscar o acceder a través del directorio. El carácter - (guión) indica que se carece del permiso correspondiente.

Por ejemplo, si los permisos de un fichero son `rwxr–xr––`, significa que el dueño tiene permiso de lectura, escritura y ejecución (`rwx`), los usuarios que pertenecen al mismo grupo al que pertenece el fichero tienen permiso de lectura y ejecución pero no de escritura (`r–x`), y el resto de los usuarios sólo tienen permiso de lectura (`r––`).

2.3.1 Orden `chmod`

Cambia los permisos de un fichero o directorio. El formato de este comando es:

```
chmod [opciones] cambio_de_permiso nombre
```

El cambio de permisos puede ser de dos formas:

- Especificando u (usuario o dueño), g (grupo) u o (otros, no es el usuario ni tampoco pertenece a su grupo), un signo + o - para dar o quitar permiso y la letra r, w o x:

  `u+x` da al usuario permiso de ejecución si es un fichero, o de búsqueda si es un directorio.

  `g+r` da al grupo permiso de lectura.

- Un número de 3 dígitos en octal. El primer dígito indica los permisos del usuario, el segundo dígito los permisos del grupo y el tercer dígito los permisos para los demás. Cada dígito se calcula sumando 4 si se quiere dar permiso de lectura, 2 si se quiere dar permiso de escritura y 1 si se quiere dar permiso de ejecución:

  `700` da todos los permisos al usuario (4+2+1) y ningún permiso al grupo ni a los demás (recomendable para el directorio de inicio de sesión).

  `644` da permiso de lectura y escritura al usuario (4+2+0), y de lectura al grupo y a los demás (4+0+0).

Pruebe con los siguientes ejemplos:

`chmod g–x bin`	quita el permiso de búsqueda al grupo para el directorio `bin`
`chmod g+x bin`	da al grupo el permiso quitado con el ejemplo anterior

2.4 Operaciones sobre ficheros.

2.4.1 Orden `cp`

Copia ficheros y directorios. Presenta tres formatos:

```
cp nombre_fichero nombre_fichero_copia
cp nombre_1 nombre_2 ...  directorio_destino
cp –r directorio_origen directorio_destino
```

En el primer caso, se crea un nuevo fichero que es una copia del original.

En el segundo, el comando permite una serie de argumentos (no sólo dos como en el primer caso), debiendo ser el último de ellos un directorio. El resultado de la ejecución del comando es que todos los ficheros especificados en los argumentos (menos el último) se copian en el directorio especificado en el último argumento.

Opciones:

`-r`	Copia directorios. Se copia en el directorio de destino el directorio especificado como origen, incluyendo toda la estructura de directorios que esté por debajo y los ficheros que contienen. Las copias tienen los mismos nombres que los originales. Si se va a copiar un directorio en otro directorio, el de destino puede no existir, y en ese caso no se copia el directorio sino lo que contiene. Cuidado con esta opción ya que no se debe dar como directorio de destino el nombre de un subdirectorio del directorio a copiar.

Para realizar los siguientes ejemplos, copie antes que nada el fichero `ordenar.txt` (que está en el subdirectorio `prac_meta` dentro del directorio correspondiente a la primera práctica) en el directorio de trabajo con el nombre `p1`:

```
cp /usuarios/primero/fp_lab/practicas/p01/prac_meta/ordenar.txt p1
```

Realice ahora los siguiente ejemplos:

`cp p1 p2`	copia el fichero `p1` en `p2` (se crea el fichero `p2` que es una copia de `p1`)
`cp p1 p3`	copia el fichero `p1` en `p3`
`mkdir prac`	crea el subdirectorio `prac`
`cp p1 p2 p3 prac`	copia los ficheros `p1`, `p2` y `p3` en el subdirectorio `prac` (se crean los ficheros `prac/p1`, `prac/p2` y `prac/p3`)
`mkdir prac/subprac`	
`ls -R prac`	muestra recursivamente el directorio `prac` y toda la estructura de directorios que está por debajo (incluyendo los ficheros)
`mkdir prac1`	
`cp -r prac prac1`	copia el directorio `prac` y todo lo que está por debajo en `prac1`
`ls -R prac1`	
`cp -r prac prac2`	copia el contenido del directorio `prac` (pero no el propio directorio `prac`) con todo lo que está por debajo en `prac2` (se crea `prac2` ya que no existía).
`ls -R prac2`	

2.4.2 Orden `mv`

Cambia de nombre o ubicación un fichero. El formato del comando es:

```
mv nom_fichero nuevo_nom_fichero
mv nom_fichero1 nom_fichero2 ... directorio
```

No se crea ningún fichero, simplemente cambia de nombre. El nuevo nombre puede hacer que el fichero cambie de directorio. Si el último argumento es un directorio se cambian los ficheros a ese directorio (es como si se "movieran"):

`mv p3 p4`	El fichero `p3` se llama ahora `p4`.
`mv p4 prac`	El fichero `p4` está ahora en el subdirectorio `prac` (su nuevo nombre es `prac/p4`).

2.4.3 Orden `rm`

Borra ficheros (o directorios, si se utiliza la opción `-r`). Para borrar hay que tener permiso de escritura en el directorio. El formato del comando es:

```
rm [opciones] nombre_fichero_o_directorio
```

Opciones:

`-i`	pregunta si se quiere borrar o no. Esta opción está activa por defecto. Si se quiere borrar el fichero se debe responder a la pregunta con `yes` o `y`.
`-r`	borra recursivamente los ficheros del directorio y el propio directorio.

Realice los siguientes ejemplos:

`chmod u-w .`	quita el permiso de escritura al usuario en el directorio de trabajo. El punto significa directorio de trabajo, y debe estar separado por al menos un espacio en blanco
`ls -ld`	muestra los permisos del directorio de trabajo (el usuario no tiene permiso `w`)
`rm p2`	intenta borrar el fichero `p2`
`chmod u+w`	da permiso de escritura en el directorio de trabajo
`rm p2`	borra el fichero `p2`
`rm -r prac prac1 prac2`	borra los directorios `prac`, `prac1` y `prac2`, su contenido y todo lo que está por debajo

2.5 Comandos para Visualización de Ficheros

2.5.1 Orden `cat`

Muestra el contenido de uno o varios ficheros:

```
cat [opciones] fichero1 [fichero2 ..]
```

Opciones:

`-n`	precede cada línea con un número de línea

Ejecute los siguientes comandos y compruebe los resultados:

```
cat p1
cat p1 p3
cat -n p1
```

2.5.2 Orden `more`

Es un filtro para ver el contenido de un fichero fácilmente. La diferencia con respecto al anterior consiste en que sólo presenta lo que cabe en la ventana, y es necesaria una acción por parte del usuario para continuar. Su formato es:

```
more [opciones] fichero
```

Una vez ejecutado el comando, es posible realizar las siguientes acciones:

`espacio`	avanza una página.
`b`	retrocede una página.
`INTRO`	avanza una línea.
`q`	termina.
`h`	ayuda.

Ejemplo:

`more p1`	
`man cp`	Al consultar el manual la información se visualiza con more (aunque la orden b a veces no funciona).

2.5.3 Orden `tail`

Muestra parte del contenido de un fichero. El fomato del comando es:

```
tail [opciones] fichero
```

Opciones:

`+n`	muestra a partir de la línea `n`.
`-n`	muestra las últimas `n` líneas.

Si no se pone opción muestra las últimas 10 líneas

Compruebe el funcionamiento del comando realizando los siguientes ejemplos:

`tail p1`	muestra las 10 últimas líneas del fichero `p1`
`tail +2 p1`	muestra desde la segunda línea del fichero `p1`
`tail -3 p1`	muestra las tres últimas líneas del fichero `p1`

2.6 Comandos informativos

2.6.1 Orden `quota`

Consulta los límites de cuota del usuario. Su formato es el siguiente:

```
quota [-v]
```

El espacio en las cuentas de usuario está limitado. La ejecución de este comando presenta la utilización del disco por parte del usuario, y los límites máximos del mismo. Cuando se usa sin opciones muestra un mensaje si el usuario ha sobrepasado su cuota. En el ejemplo mostrado a continuación el espacio en disco para el usuario es de 6000 bloques (normalmente 1 bloque se corresponde con 1024 octetos) y el límite es de 7500. El límite de la cuota puede ser sobrepasado temporalmente hasta la cantidad fijada en la columna `limit`:

```
quota -v
Disk quotas for jesus (uid 1006):
Filesystem usage quota limit timeleft files quota limit timelef
/alumnos   217   6000  7500  0        0     0     0     0
```

2.6.2 Orden `wc`

Cuenta líneas, palabras y caracteres de los ficheros. El primer número indica el número de líneas, el segundo el de palabras y el tercero el de caracteres:

```
wc [opciones] fichero
```

Opciones:

`-c`	muestra sólo el número de caracteres
`-l`	muestra sólo el número de líneas (letra ele)
`-w`	muestra sólo el número de palabras

Ejemplos:

`wc p1`	Muestra el número de líneas, palabras y caracteres del fichero `p1`.
`wc -l p1`	Muestra sólo el número de líneas del fichero `p1`.

2.6.3 Orden `who`

Muestra información acerca de los usuarios conectados. Su formato es:

```
who
```

Si se da la orden:

```
who am i
```

da información del propio usuario.

2.6.4 Orden `du`

Muestra el espacio en disco (normalmente en Kbytes) ocupado por los ficheros y/o directorios dados como parámetros. Se considera el espacio ocupado por un directorio al espacio ocupado por todos los ficheros y, recursivamente, directorios dentro de éste. El formato del comando es:

```
du [opciones] [nombre]
```

.Si no se especifica `nombre` se supone `'.'` (el directorio de trabajo).

Opciones:

`-a`	muestra el tamaño de cada fichero
`-s`	sólo muestra el resumen

Si no se especifica ninguna opción, sólo hace un resumen por directorios, sin especificar lo que ocupa cada fichero.

Ejemplos:

`du`	muestra información del directorio de trabajo
`du practica`	muestra información del directorio practica y de sus subdirectorios
`du –a practica`	muestra información del directorio practica y de sus subdirectorios (especificando lo que ocupa cada fichero)

2.7 Otros Comandos

2.7.1 Orden `sort`

Muestra las líneas de un fichero ordenadas alfabéticamente:

```
sort [opciones] fichero
```

Opciones:

`–r` muestra en orden inverso.

Ejemplo:

`sort p1`	muestra por pantalla el fichero p1 ordenado alfabéticamente
`sort`	al no especificar nombre de fichero, la orden sort irá aceptando lo que se escriba y cuando se pulse ^D (pulsar simultáneamente la tecla `Control` y la tecla `D)` mostrará lo escrito pero ya ordenado.

2.7.2 Orden `echo`

Escribe sus argumentos en la salida estándar. Es útil para comprobar el uso correcto de los metacaracteres del *shell* (se verá en el siguiente apartado de esta práctica).

Ejemplo:

`echo hola a todos`	escribe en pantalla `hola a todos`

2.7.3 Orden `diff`

Busca diferencias entre dos ficheros. Su formato es:

```
diff fichero_origen destino
```

Si `destino` es un fichero, compara los dos ficheros indicados. Si es un directorio, busca en `destino` el fichero de nombre `fichero_origen` y los compara.

La salida de la orden muestra las diferencias encontradas entre ambos ficheros. Cuando el comando encuentra diferencias en líneas consecutivas del fichero, las agrupa.

Al presentar las diferencias, las referencias al primer fichero aparecen precedidas por < (fichero de la izquierda), y las correspondientes al segundo por > (fichero de la derecha).

A continuación se presentan los tres casos típicos que pueden aparecer al ejecutar este comando:

- Si la línea (o líneas) sólo aparece en el primero de los ficheros:
  ```
  10d10
  < como podemos ver
  ```
- Si la línea (o líneas) sólo aparece en el segundo fichero:
  ```
  25a25
  > existen diferencias
  ```
- Si la línea aparece en los dos ficheros, pero es diferente:
  ```
  12c13
  < La línea es distinta
  ---
  > la línea es distinta
  ```

Los números y letras que anteceden a cada cambio especifican qué comandos habría que ejecutar en el editor de línea `sed` para obtener el segundo fichero a partir del primero. Puesto que dicho editor no se utiliza en el laboratorio, pueden ignorarse, sirviendo únicamente para separa cada grupo de cambios.

3 Funciones Especiales del Intérprete de Comandos

El intérprete de comandos, o *shell*, es el procesador de órdenes. Es un programa que lee la orden que se le da por teclado, la analiza y llama al programa del sistema operativo que realmente la ejecuta. El *shell* por defecto para los alumnos es el `csh` (*C-shell*). Para más información sobre sus características (además de las vistas en esta práctica) ejecutar:

```
man csh
```

3.1 Metacaracteres

Hay caracteres que para el *shell* tienen un significado especial, los metacaracteres. Con los metacaracteres se pueden construir expresiones que el *shell* sustituye por nombres de ficheros que ya existan y que estén de acuerdo con la expresión. Los metacaracteres que se van a estudiar son:

`*`	sustituye cualquier carácter o caracteres (excepto los que empiezan por punto).
`?`	sustituye un sólo carácter cualquiera.
`[...]`	sustituye cualquier carácter dentro del corchete. También se pueden utilizar rangos con el guión (ver ejemplos).

Para que el *shell* no haga estas sustituciones con los metacaracteres se pueden poner entre `''` (comillas simples inclinadas a la derecha: se encuentran sobre la tecla a la derecha de la tecla del cero) o anteponiendo el carácter \ si sólo es un carácter.

Ejemplos (para probarlos en el próximo ejercicio):

`echo d*`	muestra en pantalla el nombre de todos los ficheros del directorio actual que empiezan por `d`

`echo dat0[12345].*`	muestra en pantalla el nombre de todos los ficheros del directorio actual que se llaman `dat0` seguido de 1,2,3,4 ó 5, seguido de un punto y con cualquier extensión
`echo \*`	muestra un `*` en pantalla
`echo *`	muestra el nombre de todos los ficheros del directorio actual excepto los que empiezan por punto (comprobarlo con `ls -a`)
`echo dat0?.?`	muestra el nombre de todos los ficheros del directorio actual que se llaman `dat0` seguido de un carácter cualquiera, un punto y otro carácter cualquiera
`echo dat[01][1-37-9].*`	muestra el nombre de todos los ficheros del directorio actual que se llaman `dat` seguido de un 0 ó 1, seguido de 1,2,3,7,8 ó 9, seguido de punto y con cualquier extensión
`echo 'dat0?.?'`	muestra en pantalla `dat0?.?`

Aunque en los ejemplos sólo se utiliza `echo`, los metacaracteres del *shell* se utilizan con cualquier orden. Cuando se prueben estos ejemplos en el ejercicio, probarlos también con `ls` en lugar de `echo`.

3.2 Caracteres Ordinarios y Caracteres de Control

Los caracteres que se escriben en el teclado son enviados al sistema, el cual normalmente los devuelve al terminal para que aparezcan en pantalla. Este proceso se llama eco y funciona usualmente con los caracteres ordinarios excepto cuando se escribe la palabra clave.

Existen algunos caracteres con un significado especial, son los caracteres de control. El más importante es el asociado a la tecla **`INTRO`** que señala el fin de una línea de entrada, y el sistema le hace eco moviendo el cursor al principio de la siguiente línea de la pantalla. La tecla INTRO debe ser pulsada para que el sistema interprete los caracteres que se acaban de teclear.

Otros caracteres de control no tienen una tecla propia sino que deben ser escritos manteniendo pulsada la tecla `CONTROL` y pulsando simultáneamente otra tecla, por lo general una letra. Por ejemplo, pulsar la tecla `CONTROL` y a la vez la tecla `S` se indica con `^S`. Algunos de estos caracteres son los siguientes:

`^S`	Detiene el scroll de la pantalla (desplazamiento hacia arriba de la información cuando no cabe en una pantalla).
`^Q`	Reanuda el scroll de la pantalla.
`^D`	Envía un signo de fin de fichero desde el teclado. Si se da cuando el sistema espera una orden es como si se diera la orden de fin de sesión.

Ejemplos:

- Ejecutar la orden `who` y detener la salida a la pantalla. Volver a reanudarlo
- Pulsar `^S`. Escribir `who` y pulsar `INTRO`. Escribir `ls` y pulsar `INTRO`. Pulsar `^Q`.

3.3 Redireccionamiento de la Entrada/Salida

Hay muchas órdenes que toman los datos de entrada de la entrada estándar (teclado) y muestran sus datos de salida por la salida estándar (pantalla). También pueden dar mensajes de errores o diagnóstico, que van a la salida de errores (normalmente la pantalla).

El *shell* puede redirigir la E/S estándar a ficheros. De esta forma se puede conseguir que una orden tome la entrada de un fichero en vez de teclado y que la salida se lleve a un fichero en vez de a la pantalla.

Este redireccionamiento lo hace el *shell* sin que la orden tenga que detectar lo que está pasando. Los comandos de redirección pueden variar según el *shell* que estemos usando. En el caso de `csh`:

`orden >  nomf`	Se crea un fichero de nombre `nomf`, y a él va la salida estándar. Si `nomf` ya existía se pierde.
`orden >> nomf`	Si existía se añade al final.
`orden >& nomf`	Se crea un fichero de nombre `nomf`, y se redirecciona a él la salida estándar y la salida de errores.
`orden >>& nomf`	Si existía se añade al final.
`orden < nomf`	La entrada estándar es leída del fichero nomf.

Si desea redireccionar por separado la salida estándar y la salida de errores debe redireccionar primero la salida estándar, y redireccionar la salida de errores del conjunto.

```
( orden > out.txt ) >& err.txt
```

Ejecute los siguientes ejemplos y observe el resultado:

`who > usuarios`	guarda en el fichero `usuarios` la salida de `who`
`sort < usuarios`	ordena el fichero `usuarios`. Muestra la salida por pantalla
`sort < usuarios > ordenados`	guarda en el fichero `ordenados` el contenido de usuarios una vez ordenado
`ls > temp`	guarda en el fichero `temp` el listado de todos los ficheros del directorio actual
`wc < temp`	cuenta el número de líneas, de palabras y caracteres del fichero `temp`
`cat < usuarios`	muestra por pantalla el contenido del fichero `usuarios`. Es equivalente a `cat usuarios`
`du > informe`	guarda en el fichero `informe` el tamaño de los ficheros del directorio actual
`cd`	para situarnos en el directorio de inicio

`du .. > kbytes`	guarda en el fichero `kbytes` el tamaño de los ficheros del directorio padre
`du .. >&  kbytes2`	guarda en el fichero `kbytes2` el tamaño de los ficheros del directorio padre y todos los mensajes de error

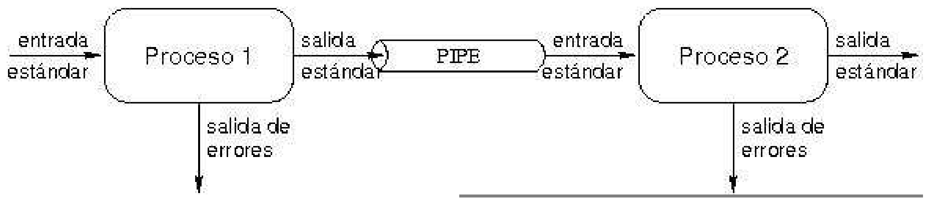
4 Procesos

Un proceso (o tarea) es la ejecución de un programa. Un programa se puede ejecutar simultáneamente por uno o varios usuarios, existiendo así varios procesos y un solo programa.

Cada proceso tiene una serie de atributos que el sistema operativo mantiene en una tabla. El más importante es el **`PID`** (Process IDentification) que es un número entero que identifica de forma única al proceso.

4.1 Intercomunicación entre Procesos

Una forma de comunicación entre procesos es mediante un PIPE o tubería, que conecta la salida de un proceso con la entrada de otro. Se utiliza el símbolo **|**. La siguiente figura recoge este concepto:



Esto permite realizar una serie de comandos en "cascada", en la que cada uno toma como entrada la salida del comando que le precede en la línea de comandos.

Los programas se ejecutan a la vez, y es el sistema operativo el que se encarga de pasar la información de uno a otro.

Se puede usar para cualquier programa que use la E/S estándar, por ejemplo las órdenes vistas anteriormente.

Si se quiere conectar también la salida de errores, no sólo la salida estándar, se usa el símbolo **|&** en vez de **|**.

No confundir el pipe con el '`;`'. Cuando se utiliza el '`;`' se ejecuta un proceso, luego otro, y así sucesivamente, sin haber ninguna intercomunicación entre ellos. Los paréntesis se usan para agrupar órdenes.

Ejemplos:

`who │ sort`	la salida de la orden `who` se utiliza como entrada para la orden `sort`. Se muestra por la salida estándar los nombres de los usuarios de forma ordenada
`ls │ wc -w`	la salida de la orden `ls` se utiliza como entrada para la orden `wc`. Se cuentan los ficheros y

	directorios del directorio actual
`cd`	cambia al directorio de inicio de sesión
`du .. | wc -l`	cuenta el número de líneas que la orden `du` envía a la salida estándar (los errores siguen saliendo por pantalla)
`du .. |& wc -l`	cuenta el número de líneas que la orden `du` envía a la salida estándar y a la salida de errores (los errores ya no salen por pantalla)
`date ; who ; ps`	ejecuta primero `date`, luego `who` y luego `ps`.
`(date ; who) > u1.dat`	se almacena en `u1.dat` la salida de `date` y de `who`.

4.2 Control de Trabajos

Un trabajo es el resultado de una petición al intérprete de comandos. El trabajo puede estar formado por un único proceso, o por un conjunto de ellos. Por ejemplo, el trabajo siguiente:

```
ls -lR
```

sólo involucra un proceso, mientras que este otro:

```
ls -lR /usr/lib | wc -l
```

(que da una idea aproximada del número de ficheros del directorio `/usr/lib` y todos sus subdirectorios) se ha creado un trabajo con dos procesos.

Un trabajo se puede encontrar en uno de tres modos de ejecución:

- en *foreground* (de forma interactiva): es el estado normal de un proceso. El terminal está controlado por el proceso. No se puede hacer otra cosa hasta que no termine de ejecutarse.
- en *background*: el terminal queda libre para seguir trabajando en otras cosas (es útil si se está ejecutando un proceso que tarda bastante).
- parado, .lo que significa que el trabajo deja de ejecutarse, a la espera de que se le especifique un nuevo cambio modo de ejecución.

Por defecto, todos los trabajos se ejecutan en *foreground*. Para ejecutar una orden en *background* se utiliza el símbolo **&** al final de la línea. Existen comandos, que se verán más adelante, para cambiar el estado de ejecución de los trabajos.

Un trabajo queda totalmente determinado por su identificador. La identificación del trabajo es un número entero, asignado por el intérprete de comandos cuando se lanzó la orden. No debe confundirse el identificador de trabajo con los diferentes PID de los procesos que componen el trabajo. Para referirnos a un trabajo lo haremos anteponiendo el carácter **%** al identificador de trabajo. Así, para referirnos al trabajo de identificador 3, escribiremos:

```
%3
```

Si un proceso que está en *background* intenta leer de la entrada estándar, éste se queda parado. Sin embargo sí puede escribir en la salida estándar, pudiendo interferir con la salida del proceso que está ejecutándose en primer plano. Para evitar esto hay que dar la siguiente orden:

```
stty tostop
```

La orden anterior le indica al sistema operativo que pare todo proceso en *background* que intente escribir en la salida estándar. Para anular el efecto de la orden anterior habrá que ejecutar:

```
stty –tostop
```

Cuando un trabajo en *background* termina, el sistema envía un mensaje al terminal indicándolo ("`Done`"). En el mensaje aparece entre corchetes el índice del trabajo terminado y la orden que lo puso en marcha. Por ejemplo:

`ls –lR /usr > b0.dat &`	se crea un trabajo en *background* con un proceso. Se crea el fichero `b0.dat` con el listado del directorio `/usr` y todos sus subdirectorios
`ps –aux | wc –l &`	se crea un trabajo en *backgound* con 2 procesos. Cuenta el número de procesos que hay en este momento en el sistema

4.3 Órdenes de información sobre Procesos y Trabajos

4.3.1 Orden `ps`

Muestra información de los procesos.

Opciones:

`a`	muestra los procesos de todos los usuarios
`x`	muestra todos los procesos aunque no estén asociados a terminal (ver fin de sesión con trabajos en background)
`u`	muestra más información.

La información que ofrece este comando es la siguiente:

`PID`	número de identificación del proceso
`TT`	terminal lógico asociado
`S`	estado del proceso. Puede ser `O` (en ejecución), `R` (la espera del procesador), `S` (inactivo a la espera de que se produzca un evento) o `T` (parado).
`TIME`	tiempo de CPU consumido por el proceso
`COMMAND`	nombre de la orden.

Ejecute `ps` en el terminal, dándole diferentes opciones, y compruebe los resultados.

4.3.2 Orden `jobs`

Muestra los trabajos que están en *background* o que están parados.

En esta orden (que da una idea aproximada del número de ficheros del directorio `/usr/lib` y todos sus subdirectorios) se ha creado un trabajo con dos procesos. Para observar la diferencia entre este comando y el anterior, ejecute ambos:

`ps`	muestra que hay dos procesos (además de `ps`)
`jobs`	muestra que hay un trabajo

Estas dos últimas órdenes hay que ejecutarlas antes de que el trabajo termine, y para hacerlo más rápido se puede utilizar el punto y coma:

```
ls -lR /usr/lib | wc -l & ; ps ; jobs
```

4.4 Órdenes de Cambio de Estado de un Trabajo

4.4.1 Orden `^C`

Destruye el trabajo que se encontraba en estado *foreground*. El comando se ejecuta pulsando simultáneamente las teclas `Control` y `C`.

Por ejemplo, ejecute la orden `who` y destruya el proceso antes de que termine de mostrar los usuarios pulsando `^C`.

4.4.2 Orden `^Z`

Pasa un trabajo del estado foreground al estado detenido. El comando se ejecuta pulsando simultáneamente las teclas `Control` y `Z`.

4.4.3 Orden `bg`

Pasa un trabajo de parado a *background*. Su formato es:

```
bg identificación_de_trabajo
```

4.4.4 Orden `fg`

Pasa un trabajo de parado o en *background* a *foreground*:

```
fg identificación_de_trabajo
```

4.4.5 Orden `kill`

Envía la señal de terminación a un proceso o trabajo del usuario (lo destruye). El formato de la orden es:

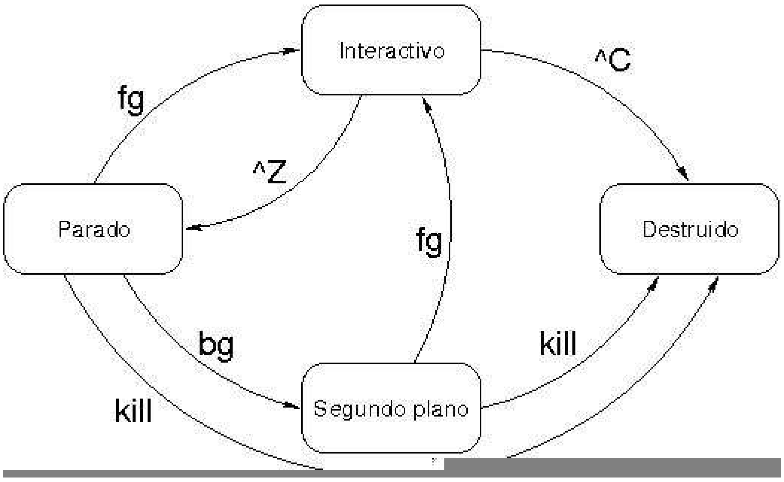
```
kill [opciones] identificación_de_proceso_o_trabajo
```

Opciones:

-9	Para procesos que ignoran la señal de terminación de trabajo.

4.4.6 Resumen

La siguiente figura resume los diferentes estados en los que se puede encontrar un trabajo, y las órdenes necesarias para pasarlo de un estado a otro.



5 *Ejercicios*

5.1 Ejercicio 1

Dada la siguiente estructura de directorios:

```
                UsuarioX
                    ¦
                 pruebas
   +----------+------------------+---------------+
trabajo    fuentes              exe          pruebas
        +------------+    +-------------+   +--------+
        c        pascal   c         pascal  c    pascal
+-------+------+
prog1   prog2  prog3
```

- Crearla dentro del directorio de inicio de sesión cambiando cada vez al directorio donde se va a crear el subdirectorio.
- Borrarla desde el directorio de inicio de sesión, sin cambiar de directorio.
- Crear hasta el directorio `fuentes`, y desde allí crear el resto de la estructura.
- Copiar los ficheros `ordenar.txt` y `datos.txt` del directorio `prac_meta` (que se encuentra en la estructura de directorios correspondientes a las prácticas), en el directorio `fuentes`.
- Hacer que el directorio de trabajo sea el directorio padre de `fuentes`.
- Quitar todos los permisos del directorio `fuentes` para el usuario.
- Intentar listar el contenido del directorio `fuentes`.
- Dar permiso de lectura al directorio `fuentes` para el usuario. Listar el contenido de este directorio.
- Intentar borrar el directorio `pascal`.
- Dar permiso de escritura y ejecución al directorio `fuentes` para el usuario. Borrar el directorio `pascal`.

- Desde el directorio `prog1` borrar todo lo que sea posible, y el resto desde el directorio de inicio.

5.2 Ejercicio 2

- Crear un directorio llamado `prac_fich`.
- Cambiar a este directorio (hacer que ese directorio sea el directorio de trabajo).
- Copiar el fichero `ordenar.txt` que está en el directorio prac_meta,
- Mostrar el contenido de este fichero.
- Contar las palabras, líneas y caracteres del fichero.
- Mostrar ordenadas alfabéticamente las líneas del fichero.
- Copiar el fichero en `dat1.1` y en `dat1.3`.
- Quitar el permiso de escritura para el usuario al directorio `prac_fich`.
- Intentar borrar el fichero `dat1.3`.
- Restaurar el permiso de escritura en `prac_fich` y volver a intentar.
- Crear en el directorio `prac_fich` el subdirectorio `subprac_fich`.
- Mover todos los ficheros de `prac_fich` a `subprac_fich`.
- Borrar con una sola orden el subdirectorio `prac_fich` completo.

5.3 Ejercicio 3

- Copiar el directorio `prac_meta` completo en el directorio de inicio de sesión. Este directorio ya existe en la estructura de directorios y se debe copiar con una sola orden (no hace falta crear previamente el directorio) desde el directorio de inicio de sesión. Una vez realizada la copia, en el directorio de inicio de sesión debe aparecer un directorio de nombre `prac_meta` que contenga una copia de los ficheros que se encuentran en el directorio indicado.
- Listar de `prac_meta` todos los ficheros que tengan de extensión un solo carácter.
- Listar de `prac_meta` todos los ficheros que empiecen por `dat`.
- Ver resultado de los ejemplos.
- Crear en `prac_meta` el subdirectorio `subprac_meta`.
- Pensar cuál es el resultado de la siguiente orden:
  ```
  mv dat0?.1 datos0?.1
  ```
  Ejecutarla y comprobar el resultado.
- Copiar en `subprac_meta` todos los ficheros `dat01.1` hasta `dat09.1`.
- Borrar de `prac_meta` todos los ficheros cuya extensión sea `1`.

5.4 Ejercicio 4

- Mostrar los procesos del usuario.
- Mostrar los procesos de todos los usuarios.
- Mostrar los procesos de todos los usuarios aunque no estén asociados a un terminal.
- Imprimir en pantalla y escribir en el fichero `usuarios.dat` todos los usuarios conectados a la máquina, en una sola orden.
- Copiar el fichero `bucle_infinito` (que se encuentra en el subdirectorio `bucle` dentro del árbol de directorios de las prácticas) en el directorio de trabajo. Este fichero es ejecutable y contiene únicamente un bucle infinito (que no para nunca), y que no escribe nada en la salida estándar:
  - Ejecutar el programa en *background*.
  - Destruirlo.
  - Ejecutar de nuevo este programa pero esta vez en *foreground*.
  - Pararlo.
  - Pasarlo de parado a *background*.
  - Volverlo a *foreground* y por último destruirlo.