Sistemas Operativos Multimedia

Práctica 0

Introducción a UNIX (I)

El sistema operativo es el software que controla la ejecución de los programas de aplicación y que actúa como interfaz ente el usuario y el hardware del computador. UNIX es el sistema operativo más utilizado en investigación científica, entre sus características más relevantes se pueden citar las siguientes:

- Multitarea. Puede ejecutar varios procesos o tareas simultáneamente, ya sea con uno o con varios procesadores.
- Multiplataforma. Está disponible para diversas arquitecturas.
- Multiusuario. Permite que varios usuarios trabajen simultáneamente. En UNIX cada usuario tiene creada una cuenta propia. Las cuentas de los usuarios son creadas por el superusuario (root), un usuario con privilegios especiales que se encarga de la administración del sistema
- Utiliza memoria virtual, en caso de falta de memoria RAM se utiliza el disco duro (swap)
- Gestiona los dispositivos de entrada/salidas como archivos.
- Permite compartir dispositivos en red.

El sistema UNIX que utilizaremos para las prácticas será el Linux. Linux se diseñó inicialmente como un clónico de UNIX distribuido libremente que funcionaba en máquinas PC con procesador 386 o superior, aunque en la actualidad funciona sobre otras plataformas como los procesadores Alpha, Sparc, máquinas de tipo MIPS y sobre PowerPC. Linux es ante todo un sistema UNIX rápido y completo, que puede instalarse fácilmente. Además, su difusión entre el gran público le permite evolucionar rápidamente.

CONEXIÓN AL SISTEMA

Todo usuario debe identificarse en el momento de la conexión al sistema. Para ello debe introducir, en primer lugar, su *identificador de usuario* (o *login-name*) en una terminal libre, como respuesta al mensaje de "*login:*" Y acreditar su personalidad mediante una *contraseña* (o *password*) que debe ser introducida cuando el sistema visualice el mensaje de "*password:*".

Aparte de utilizar la consola propia del computador en el que está instalado el sistema Unix, existen multitud de formas de conectarse a una máquina Unix. La más habitual es a través de *telnet*. *telnet* es un protocolo de comunicación que nos permite conectarnos a la máquina Unix para ejecutar comandos y obtener los resultados en nuestra pantalla. Lo primero que necesitamos es un *cliente telnet*, que es un programa que nos permite realizar la conexión e intercambiar la información entre nuestro ordenador y el ordenador Unix, utilizando el protocolo *telnet*. Habitualmente este programa se llama también telnet. Por ejemplo, en Windows, tenemos el programa c:\windows\telnet.exe, y Unix existe el comando *telnet*. De alguna forma debemos indicar con cuál ordenador queremos conectarnos, y eso lo hacemos especificando su nombre o su dirección IP. Esta identificación la pasamos como parámetro al programa *telnet*.

Cuando se inicia una sesión, el sistema arranca el shell de inicio el cual establece el entorno y permite interactuar con el usuario. El procedimiento de arranque del shell de inicio es el siguiente:

1. En primer lugar, el shell de inicio leerá los comandos del archivo `/etc/profile`, si es que existe. Este archivo es común a todos los usuarios del sistema y controla las opciones por defecto del sistema como por ejemplo la exportación de variables, la máscara cuando se crean nuevos archivos, tipos de terminales, mensajes de correo para indicar cuando un nuevo correo ha llegado, etc. El administrador del sistema es el único usuario que puede configurar este archivo.

2. A continuación, sí el archivo `.profile` existe en el directorio inicial del usuario (llamado directorio "home" del usuario), el shell de inicio leerá los comandos que contiene. Este archivo permite sobreescribir comandos y variables que se había especificado en el archivo `/etc/profile`. Además, permite personalizar el entorno de trabajo de cada usuario configurando por ejemplo, la apariencia del indicador del sistema, las variables de entorno, tipo de shell, etc.

CAMBIO DE CONTRASEÑA

Una de las primeras labores a realizar si todavía no tiene contraseña es introducir una. Para introducir o modificar una contraseña utilice la orden `passwd`: la orden `passwd` solicita, en primer lugar, al usuario la introducción de la contraseña actual; a continuación, la nueva contraseña y, para confirmar, la nueva contraseña otra vez. Si se produce algún error, `passwd` no modifica la contraseña.

DESCONEXIÓN DEL SISTEMA

Para cerrar una conexión con una máquina Unix basta con teclear "`exit <Ret>` " o "`^D <Ret>`". En cualquier caso, Unix presentará el mensaje de despedida `logout`. Y, a continuación, desaparecerá la ventana correspondiente a la conexión.

Una vez cerradas todas las conexiones con las otras máquinas (si había alguna) proceda a cerrar el entorno de ventanas. Para ello, pinche con el botón izquierdo sobre el botón Inicio y seleccione en el primer menú la opción `Salir`. Una vez haya salido del entorno de ventanas, para cerrar la sesión de trabajo escriba la orden `exit`.

NO SE DEBE APAGAR LA MAQUINA NI PULSAR EL BOTON DE RESET SINO PULSAR `CTRL-ALT-SUPR` PARA QUE EL SISTEMA SE DETENGA ORDENADAMENTE.

EL INTÉRPRETE DE ÓRDENES

Una vez se arranca la máquina Unix, se ejecuta el programa intérprete de órdenes, también conocido como *shell*. Existen diversas versiones de este programa: shell C (`csh`), shell Korn (`ksh`), shell Bourne (`bsh`), etc. Para las prácticas de *Sistemas Operativos* se utilizará el intérprete conocido como GNU Bourne Again Shell (`bash`), que es una variante del Bourne clásico. El intérprete de órdenes desplegará algún indicador o *prompt*. Este indicador denota que el intérprete espera una orden del usuario desde el teclado con el fin de que Unix los ejecute. La orden se ejecuta al pulsar `<Ret>.`

Ejemplos de órdenes sencillas y a la vez muy útiles son la orden `ls` que lista los archivos del directorio actual y la orden `ps` que lista los procesos de un usuario. Se ha de considerar que cada orden que se lance se convertirá en uno o más procesos que representan las unidades de ejecución del

sistema. Asimismo cada proceso tendrá asignado un descriptor de entrada estándar para obtener datos que procesar (por defecto es el teclado), un descriptor de salida estándar para devolver los resultados del procesamiento (por defecto es la pantalla) y un descriptor de salida de errores que pueden ocurrir en su ejecución (por defecto es la pantalla). Los descriptores de los procesos, como se verá más adelante, se pueden modifican con los caracteres especiales < (para entrada estándar), > (para salida estándar) y 2> (para salida de error).

```
$ ls <Ret>
$ ps <Ret>
```

Tenga en cuenta al teclear las órdenes que Unix distingue entre mayúsculas y minúsculas. La orden puede corregirse con la tecla de retroceso. El intérprete `bash` permite otras funciones avanzadas de edición de órdenes. Una de las más útiles es el "rodillo de órdenes" que permite recuperar órdenes ejecutadas anteriormente con los cursores ↑ y ↓.

CARACTERES DE CONTROL

Se denominan caracteres de control a aquellos que producen un efecto inmediato cuando se pulsan Los más importantes son:

- <Crtl> c: Termina o aborta la ejecución de la orden que se esté ejecutando
- <Crtl> s: Detiene la visualización en pantalla.
- <Crtl> q: Reanuda la visualización en pantalla
- <Crtl> d: Es utilizado por aquellos programas que aceptan datos desde teclado para indicar el final de los datos.

FORMATO DE LAS ÓRDENES

Muchas órdenes aceptan argumentos. Para Unix, el separador de argumentos es el espacio en blanco. La mayoría de órdenes asumen como opciones los argumentos cuyo primer carácter es el signo "–". Las opciones pueden expresarse por separado o combinadas

```
$ ll /etc/passwd /usr/lib
$ ls -l
$ ls -l /etc/passwd
$ ls -la
```

ÓRDENES BÁSICAS DE UNIX

Esta parte de la práctica tiene como finalidad conocer algunas de las órdenes básicas de Unix.

ORDEN MAN

Unix dispone de un manual en línea que permite consultar la sintaxis, la descripción y las opciones de cualquier orden sobre la propia terminal. Este manual se invoca con la orden `man`.

Así por ejemplo, podría haber obtenido la información correspondiente a la propia orden `man:`

```
$ man man
```

La información se proporciona paginada por pantallas. Al final de la pantalla la indicación `--More--` interroga si se desea avanzar a la siguiente página. Se puede contestar:

- **<Space>** (Barra espaciadora): Avanzar a siguiente página.
- **q** (quit): Abandonar
- **?** ó **h** (help). Para ver otros mandatos disponibles.

También existen otras posibilidades de obtener ayuda acerca de los mandatos del sistema: como `help`.

A PARTIR DE AHORA, RECUERDE UTILIZAR EL MANUAL CUANDO TENGA ALGUNA DUDA.

ORDEN DATE

Esta orden permite consultar la fecha y hora del sistema

El formato que utiliza date es el siguiente:

- día de la semana
- mes
- día del mes
- hora
- zona horaria
- año

`date` también sirve para modificar la fecha y hora, pero sólo el superusuario puede modificar estos valores. La fecha y hora son valores críticos para un sistema multiusuario. Muchos de los servicios del sistema dependen de que estos valores sean correctos. Por ello, tan sólo el superusuario puede modificarlos.

```
$ date
```

ORDEN WHO

Esta orden visualiza los usuarios conectados a una máquina. El formato que utiliza `who` es el siguiente

- Nombre del usuario: `login`
- Terminal de conexión: `tty???`
- Momento de la conexión

También puede utilizarse para conocer la propia identidad

```
$ who am i
```

OBTENCIÓN DEL DIRECTORIO ACTUAL (PWD)

Cuando entramos en el sistema a través de nuestro nombre de usuario (login) y nuestra palabra de paso (password), el sistema nos sitúa sobre nuestro directorio inicial. Para comprobarlo ejecutar la orden

```
$ pwd
```

Para saber sobre qué directorio estamos en un momento dado podemos utilizar la orden `pwd`. Si ejecutamos `pwd` inmediatamente después de entrar al sistema, lo que aparece es el camino completo de la situación de nuestro directorio dentro del sistema empezando por el directorio raíz "/". Del directorio raíz cuelgan todos los demás directorios del sistema.

LISTADO DEL CONTENIDO DE UN DIRECTORIO (LS)

La orden `ls` es una petición al sistema para mostrar el contenido de un directorio. La orden `ls` tiene diversas variantes (como la mayoría de las órdenes UNIX). Si tecleamos `ls` con la opción `-a` (all) nos aparecen además los archivos ocultos (aquellos cuyo nombre empieza por “.”).

Podemos combinar varias opciones a la vez. Por ejemplo las opciones `-a` y `-l` (long). Con esta combinación de opciones hemos conseguido obtener más información. En este listado nos aparecen todos los archivos. Hay dos archivos especiales que son el “.” y “..”. El archivo “.” hace referencia al directorio actual y el archivo “..” hace referencia al directorio padre.

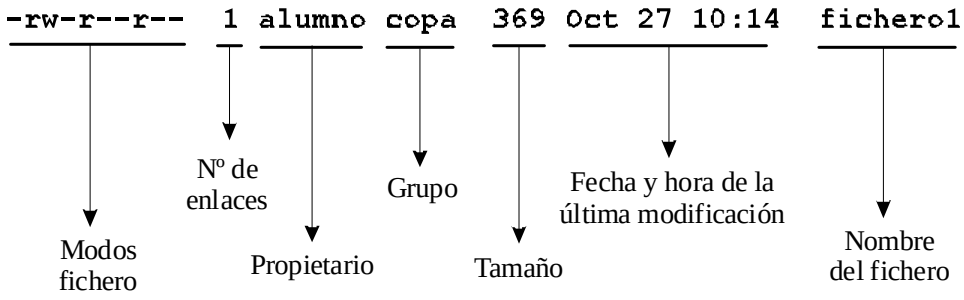
Si en un momento dado queremos saber qué archivos en un directorio son archivos ordinarios y cuáles son directorios, utilizamos la opción `–F`. Los archivos cuyo nombre acaba en `/` son directorios, los que acaban en “`*`” son ejecutables (código) y los que finalizan con @ son enlaces a otros archivos.

La distinción entre archivos ordinarios y directorios también se puede apreciar al ejecutar `ls –l` si observamos el primer carácter (empezando por la izquierda) de cada fila (archivo). Las entradas cuyo carácter es una “d” son directorios y los que aparece un “-“ son archivos ordinarios. Existe una

entrada especial “l” que hace referencia a un enlace (*link*). Un enlace es una referencia a un archivo que está físicamente en otro lugar.

SIGNIFICADO DE LOS CAMPOS EN EL LISTADO DE ARCHIVOS

Como se ha comentado anteriormente al efectuar el mandato ls -l aparecen una serie de entradas (filas), de tal forma que cada una de ellas hace referencia a un archivo. Una entrada típica consta de varios campos. El significado de cada uno de ellos es el siguiente:



- **Modos archivo (bits de protección):** El primer elemento (el de más a la izquierda) especifica el tipo de archivo. Los valores posibles son 'd', si hace referencia a un directorio, vacío('-'), si hace referencia a un archivo ordinario y 'l' si hace referencia a un enlace. El resto de los elementos son los llamados bits de protección. Están compuestos por tres secuencias contiguas de valores 'r', 'w' y 'x'. El significado de estas secuencias y sus valores lo estudiaremos más adelante.
- **Número de enlaces:** Para archivos, indica el número de enlaces físicos que se refieren a ese archivo. Para el caso de directorios este número coincide con el número de subdirectorios existentes en ese directorio más dos. Es decir, si el directorio no tiene subdirectorios, su número de enlaces es 2, si tiene un subdirectorio su número sería 3, y así sucesivamente.
- **Nombre propietario:** Indica el nombre del propietario del archivo.
- **Nombre grupo:** Un conjunto de usuarios puede formar parte de un grupo con una serie de características en común. Este campo hace referencia al grupo al que pertenece el usuario.
- **Tamaño archivo**: Muestra el tamaño de archivo en bytes.
- **Fecha y hora de la última modificación:** Hace referencia a la hora en que el archivo fue modificado por última vez. Si dicha fecha supera el medio año de antigüedad entonces éste también aparece en la fecha. Si queremos saber cuando se accedió por última vez utilizamos la opción `-u` combinada con `-l`.

La ordenación de los archivos por defecto es en forma alfabética ascendente. Si deseamos efectuar una ordenación por fechas podemos utilizar junto con los modificadores anteriores, el modificador `-t` (por defecto primero los más nuevos) y si además queremos invertir el orden (primero los más antiguos) añadimos el modificador `-r`.

Podemos hacer notar que la opción `-R` (mayúscula) es diferente de la anterior, ya que lista recursivamente un conjunto de directorios, bien a partir del directorio donde nos encontramos (pwd), o bien a partir del directorio que le pasemos como argumento. Comentar por último que el mandato `ls` tiene muchas más opciones que no son explicadas aquí. Para más información ejecutar `man ls`.

CAMBIO DE DIRECTORIO (CD)

Hemos comentado anteriormente que, por defecto, la secuencia de entrada en UNIX nos sitúa en nuestro directorio de trabajo. Pero al igual que otros sistemas operativos (p. ej. MS-DOS) podemos cambiar de directorio mediante el mandato `cd directorio`. Si no introducimos ningún nombre de directorio, el mandato `cd` sin argumentos vuelve a nuestro directorio de trabajo. Si queremos volver al directorio de nivel superior basta con utilizar `cd ..`

Así, podemos listar el contenido del directorio /usr o el directorio raíz del sistema (/).

```
$ ls /usr
```

El nombre de un archivo o de un directorio se puede referenciar de forma relativa o absoluta.

- **Forma relativa**: El nombre hace referencia a archivos o directorios desde el directorio en el que nos encontramos.
- **Forma absoluta**: El nombre hace referencia a todo el camino desde la raíz.

Si queremos consultar el contenido del archivo `passwd,` mediante el comando `cat`, podemos acceder a él a través de su camino absoluto:

```
$ cat /etc/passwd
```

o a través del relativo:

```
$ cd /etc ; cat passwd
```

REGLAS PARA NOMBRAR ARCHIVOS

Los nombres de los archivos están formados por caracteres. Su número varía entre los diferentes sistemas UNIX (en algunos hasta 14, y en otros hasta 255). Los caracteres válidos pueden ser cualesquiera en teoría. En la práctica hay algunos que debemos evitar:

'**;**', '<','>', '**$**', '|', '*****', '**?**'

ya que estos caracteres tienen un significado especial dentro de los mandatos UNIX.

Como regla general se trata de utilizar los caracteres alfabéticos, numéricos, el guión inferior (_) y el (`.`). Este último no se ha de utilizar como primer carácter del nombre de un archivo a no ser que queramos ocultarlo. UNIX oculta (a nivel de mandato `ls`) los nombres de archivo que comienzan con `.` excepto si utilizamos la orden `ls -a`. Ejemplo:

- Nombres correctos:

```
practica.c
mi_practica
practica3
```

- Nombres incorrectos:

```
practica*
>practica
prac|tica
```

CARACTERES COMODINES

A veces es interesante referenciar archivos que tengan en su nombre características comunes. “Todos los archivos que empiezan por la letra c...”. En UNIX esto se consigue utilizando caracteres especiales (llamados metacaracteres o comodines) que representan otras cosas:

- El carácter asterisco '*' representa a cualquier cadena de caracteres arbitraria incluyendo la cadena vacía.
- La interrogación '?' representa a cualquier carácter simple.
- Los corchetes '[' ']' pueden contener un grupo o rango de caracteres y corresponden a un carácter simple.
- Las llaves ‘{‘,’}’ deben contener diferentes alternativas, constituidas por un carácter o un grupo de caracteres, separadas todas ellas por comas. El shell utiliza todas las alternativas especificadas para formar una serie de nombres a partir del patrón donde aparezcan.

Por ejemplo, crea en tu directorio inicial (`home`) el directorio `so1` y dentro de él ejecutar los siguientes mandatos. Si no se conoce para qué sirve el comando `touch` se puede consultar al manual (`man`).

```
$ touch a2 archivo{1,2,3,4,5,12} c{1,2,3}
$ ls
$ ls a*
$ ls archivo?
$ ls c[1-3]
$ ls c[1,3]
$ ls c[13]
$ ls *2
```

¿Qué se está haciendo con cada comando?

EL METACARACTER *

El shell, no los mandatos, interpreta este carácter antes de ejecutar un mandato. Lo sustituye por los nombres de los archivos existentes en el directorio actual, y estos nombres son pasados como argumentos.

Para comprobarlo se puede utilizar el mandato `echo`. Este mandato envía a la salida estándar sus argumentos. Por ejemplo, pruébense, las siguientes líneas de mandato.

```
$ echo *
$ ls *
```

EVITANDO LA INTERPRETACIÓN DE LOS METACARACTERES

El shell no interpreta los caracteres encerrados entre comillas o antecedidos por la barra invertida (\), lo que permite escribir un mandato en varias líneas:

```
$ echo ’*’
$echo “Los archivos “ * “estan en el directorio actual.”
$echo “Los archivos \”*\” son “ * “y estan en el directorio actual.”
$ echo \
aquí \
hay \
cuatro \
argumentos
$ echo ’Se puede usar el intro
> dentro de comillas’
```

VISUALIZACIÓN DE ARCHIVOS

Debido a que UNIX es un sistema operativo de gran tamaño y con gran cantidad de mandatos existen múltiples órdenes de visualización del contenido de archivos.

ORDEN FILE

Antes de visualizar el contenido de un archivo es conveniente saber de qué tipo es. Para ello podemos utilizar la orden `file` seguida del nombre de archivo del que queremos averiguar su tipo.

Averiguemos cuál es el tipo de archivo del archivo de configuración del *shell* (intérprete de órdenes de UNIX) `/etc/profile`

```
$ file /etc/profile
/etc/profile:       ascii text
```

Puesto que el archivo `/etc/profile` es de texto podemos visualizarlo.

ORDEN CAT

La orden `cat` se utiliza para visualizar sobre la salida estándar el contenido de un archivo. Lógicamente el tipo de archivos a visualizar debe ser de texto, ya que si utilizamos la orden con un archivo ejecutable la salida sería ilegible.

Visualicemos el archivo de texto `/etc/profile`

```
$ cat /etc/profile
...
```

Podemos listar por ejemplo la lista de usuarios del sistema, que suelen encontrarse en el archivo `/etc/passwd`

```
$ cat /etc/passwd
```

```
...
```

Si el archivo no cabe en pantalla, como en este caso, podemos utilizar las órdenes <Crtl>-S (para detener la salida) y <Crtl>-Q (para reanudarla)

La orden `cat` permite listar varios archivos secuencialmente. Si tenemos dos archivos llamados `archivo1` y `archivo2`, la orden:

```
$ cat archivo1 archivo2
```

lista en primer lugar el archivo `archivo1` y a continuación `archivo2`.

Una aplicación muy útil de `cat` es concatenar archivos. Por ejemplo, si queremos concatenar los dos archivos anteriores en un nuevo archivo llamado `archivo3` bastaría con ejecutar:

```
$ cat archivo1 archivo2 > archivo3
```

ORDEN MORE

Una orden alternativa a `cat` es la orden `more` que da más control que la anterior, ya que automáticamente lista un archivo y cuando llena la terminal (lista tantas líneas como el tamaño de la terminal) se para, esperando que pulsemos la tecla espacio para reanudar la salida.

```
$ more /etc/termcap
```

Nos dice además el porcentaje de archivo que ya ha sido listado.

La orden `more` tiene varias opciones interesantes:

- Con el modificador `-n`, lista el archivo presentando de n en n líneas y no con el número de líneas que posee nuestra pantalla.
- Con el modificador `+n`, lista el archivo a partir de la línea `n`.

ORDEN TAIL

La orden `tail` permite visualizar el final de un archivo. Por defecto visualiza las 10 últimas líneas. Así por ejemplo:

```
$ tail /etc/profile
```

lista las últimas 10 líneas de nuestro archivo `/etc/profile`.

Si queremos listar por ejemplo las últimas 5 líneas

```
$ tail -5 /etc/profile
```

y si queremos visualizar a partir de la línea 2 entonces:

```
$ tail +2 /etc/profile
```

CREACIÓN Y BORRADO DE DIRECTORIOS

En este punto vamos a estudiar la creación y borrado de directorios, es decir, la estructuras de datos que contienen archivos.

CREACIÓN DE DIRECTORIOS (MKDIR)

Para crear un directorio es necesario utilizar la orden `mkdir` nombre(s) de directorio(s)

Si queremos crear un solo directorio:

```
$ mkdir prueba1
```

Si queremos crear varios directorios a la vez:

```
$ mkdir prueba2 prueba3
```

Podemos comprobar la creación haciendo un listado con `ls –l`:

```
$ ls -l
total 57
-rw-r--r--  1 alumno copa   38 Oct 27 10:20  a1
-rw-r--r--  1 alumno copa   41 Oct 27 10:20  a2
-rw-r--r--  1 alumno copa   44 Oct 27 10:20  c1
-rw-r--r--  1 alumno copa   47 Oct 27 10:20  c2
-rwxr-xr-x  1 alumno copa   50 Oct 27 10:26  c3
drwxr-xr-x  2 alumno copa  512 Oct 27 10:14  directorio1
-rw-r--r--  1 alumno copa  369 Oct 27 10:14  archivo1
-rw-r--r--  1 alumno copa  423 Oct 27 10:14  archivo2
drwxr-xr-x  2 alumno copa  512 Nov  7 22:03  prueba1
drwxr-xr-x  2 alumno copa  512 Nov  7 22:04  prueba2
drwxr-xr-x  2 alumno copa  512 Nov  7 22:04  prueba3
```

Si necesita crear un archivo puede hacerlo mediante la orden `touch`.

También podemos crear subdirectorios utilizando los caminos:

```
$ mkdir prueba1/prueba11 prueba2/prueba21 prueba3/prueba31
```

Para comprobar todos los niveles de subdirectorios que hemos creado, podemos utilizar la opción `ls -R` que lista recursivamente archivos y directorios:

```
$ ls -R
a1   c1   directorio1     archivo3        prueba2
a2   c2   archivo1        mensaje         prueba3
bin  c3   archivo2        prueba1

bin:

directorio1:

prueba1:
```

```
prueba11

prueba1/prueba11:

prueba2:
prueba21

prueba2/prueba21:

prueba3:
prueba31

prueba3/prueba31:
```

BORRADO DE DIRECTORIOS (RMDIR)

La orden `rmdir` elimina un directorio. Es necesario que dicho directorio esté vacío.

```
$ rmdir  prueba1/prueba11
```

COPIA, MOVIMIENTO, RENOMBRADO Y ELIMINACIÓN DE ARCHIVOS

ORDEN CP

Si queremos copiar un archivo utilizamos el mandato `cp`. Por ejemplo, suponer que queremos copiar el archivo de los caracteres ASCII que hemos utilizado anteriormente. El primer argumento del mandato es el archivo origen y el segundo el destino. El archivo destino es físicamente diferente del origen. Ejemplo:

```
$ cp /etc/profile miprofile
```

Esto copia el archivo `/etc/profile` a nuestro directorio y con el nombre `miprofile`. Esto es equivalente a utilizar el mandato:

```
$ cp /etc/profile ./miprofile
```

Recordemos que . es nuestro directorio actual.

También podemos efectuar la copia a un directorio concreto

```
$ cp /etc/profile prueba3
```

introduce el archivo en el directorio `prueba3`.

La orden `cp` también copia directorios. Lógicamente querremos copiar tanto un directorio como su contenido. Para esto utilizamos el modificador recursivo `-R`.

```
$ cp -R prueba3 prueba4
```

ORDEN MV

El cometido de la orden `mv` es mover archivos entre diferentes directorios. Si se usa sobre el mismo directorio el efecto obtenido consiste en cambiar el nombre al archivo.

Ejemplos:

```
$ mv miprofile nuevo_profile
```

Cambia el nombre del archivo `miprofile` a `nuevo_profile`. Mientras ...

```
$ mv nuevo_profile prueba4
```

coloca el archivo `nuevo_profile` en el directorio `prueba4`.

Para comprobarlo utilizar el mandato `ls prueba4` que devolverá el contenido del directorio `prueba4`. La orden `mv` ha cambiado el archivo de sitio (ha movido el archivo). Si ejecutamos la orden `ls` directamente podremos observar que el archivo `nuevo_profile` ha desaparecido del directorio en el que se encontraba.

Si el archivo destino al que copiamos o movemos ya existe y no tiene permisos de escritura entonces el sistema nos pide confirmación. Los permisos del archivo copiado o movido son los mismos que los del archivo original. Estudiaremos los permisos más adelante en esta práctica.

Para más información sobre ambos mandatos consultar el manual.

BORRADO DE ARCHIVOS (RM)

La orden `rm` suprime un archivo de un directorio. Si queremos borrar el archivo que habíamos creado anteriormente...

```
$ rm profile
```

OPCIONES

Vale la pena resaltar algunas de las opciones que admite la orden `rm`:

- **`-i`** : Opción interactiva. Solicita la confirmación del usuario antes de proceder al borrado.

```
$ rm -i  archivo1
rm: remove archivo1?
```

- **`-r`** : Opción recursiva. Borra recursivamente todos los directorios y subdirectorios del nivel que estamos y de los niveles inferiores. ¡OJO! ESTA ORDEN ES MUY PELIGROSA.

PROPIEDAD Y PROTECCIÓN

Puesto que el sistema operativo UNIX es de tipo multiusuario, hemos de manejar los conceptos de propiedad y protección, es decir, a quién pertenece un determinado archivo y cuáles son los privilegios de acceso para un determinado archivo respectivamente.

Ejemplo: Creemos un archivo de la siguiente forma:

```
$ ls -l > hola
```

Este mandato utiliza el concepto de la redirección (que trataremos más adelante). Si mostramos el contenido de este archivo podremos observar que contiene un listado del directorio en el que nos encontramos:

```
$ more hola
total 57
-rw-r--r--  1 alumno copa   38 Oct 27 10:20  a1
-rw-r--r--  1 alumno copa   41 Oct 27 10:20  a2
-rw-r--r--  1 alumno copa   44 Oct 27 10:20  c1
-rw-r--r--  1 alumno copa   47 Oct 27 10:20  c2
-rwxr-xr-x  1 alumno copa   50 Oct 27 10:26  c3
drwxr-xr-x  2 alumno copa  512 Oct 27 10:14  directorio1
-rw-r--r--  1 alumno copa  369 Oct 27 10:14  archivo1
-rw-r--r--  1 alumno copa  423 Oct 27 10:14  archivo2
-rw-r-----  1 alumno copa    0 Oct 27 13:11  hola
drwxr-xr-x  2 alumno copa  512 Nov  7 22:03  prueba1
drwxr-xr-x  2 alumno copa  512 Nov  7 22:04  prueba2
drwxr-xr-x  2 alumno copa  512 Nov  7 22:04  prueba3
```

Podemos observar que el propietario de este archivo es `alumno`, que pertenece al grupo `copa`, y que sus bits de acceso están de la siguiente forma : `rw-r-----.` ¿Que indica todo esto?

Para cada archivo del sistema hay tres clases de usuarios que pueden tener acceso en los siguientes modos:

- **Propietario**: Todos los archivos creados en UNIX tienen su propietario. Habitualmente la persona que lo creó. El propietario de un archivo puede asignarle diversos privilegios de acceso. Para cambiar a un archivo de propietario se utiliza la orden `chown` (que habitualmente sólo usa el administrador del sistema).
- **Grupo**: Varios usuarios pueden tener alguna característica común (p. ej. trabajar en un mismo proyecto).
- **Público**: El resto de usuarios del sistema (exceptuando al propietario y al grupo)

Todos los archivos del sistema tienen tres tipos de permisos que describen qué tipo de operaciones se pueden efectuar con ese archivo:

- **Lectura (r)**: Un usuario que tiene permiso de lectura sobre un archivo puede leerlo. Un usuario que tiene permiso de lectura sobre un directorio puede averiguar qué contenidos hay en él sólo con el mandato `ls`. Si quiere leer el contenido de algún archivo dentro de ese directorio depende de los permisos de ese archivo.

- **Escritura (w):** Un usuario que tiene permiso escritura sobre un archivo puede cambiar el contenido de dicho archivo. Un usuario que tiene permiso de escritura sobre un directorio puede crear y borrar archivos sobre él (si además tiene el permiso de ejecución).
- **Ejecución (x)**: Un usuario que tiene permiso de ejecución sobre un archivo puede ejecutarlo. Aunque el permiso de ejecución se puede aplicar a cualquier archivo, sólo tiene sentido si éste es un ejecutable. Un usuario que tenga permiso de ejecución sobre un directorio puede acceder a él, copiar archivos a ese directorio (si tiene permiso de escritura) y copiar archivos de él (si tiene permiso de lectura).

Para cada uno de los posibles modos de usuario comentados anteriormente (propietario, grupo y público) hay tres tipos de privilegio posibles (lectura, escritura y ejecución). Esto nos da un total de nueve modos posibles que normalmente se escriben como: `rwxrwxrwx`.

Ejemplo:
Nuestro archivo hola tiene los siguientes privilegios:

```
$ ls -l hola
-rw-r-----  1 alumno copa  58 Oct 27 13:11 hola
```

Para:

- propietario: escritura y lectura.
- grupo: lectura.
- resto: nada.

ORDEN CHMOD

Para poder cambiar estos permisos hemos de ser los propietarios del archivo (o administradores del sistema) y utilizar el mandato `chmod`.

La sintaxis es :

```
chmod modo_protección archivo(s)
```

Para especificar el modo existen diversas formas. Vamos a utilizar en primer lugar la más conocida y que se basa en la representación binaria. Se trata de representar cada uno de los 9 permisos mediante unos o ceros en función de si un permiso está activado o no.

Ejemplos:
`rw-r-----` significa 110 100 000 que en modo octal sería 640

Se puede observar que para convertir de binario a octal basta con agrupar los bits de tres en tres y convertir a decimal.

Supongamos que nosotros somos los únicos que deseamos poder leer y escribir sobre nuestro archivo `hola` y que el resto de usuarios sólo puedan leerlo, (una opción bastante lógica). Los permisos quedarían como:

`rw-r--r--`. Es decir 110 100 100 que en modo octal sería 644

Para cambiar los permisos del archivo habría que escribir lo siguiente:

```
$ chmod 644 hola
```

Confirmemos que hemos cambiado el permiso de forma correcta:

```
$ ls -l hola
-rw-r--r--  1 alumno copa    58 Oct 27 13:11 hola
```

Es conveniente que protejamos nuestro directorio para evitar problemas. Si queremos hacerlo:

1. Nos situamos sobre él (`cd`)
2. Nos colocamos en un nivel superior (`cd ..`)
3. Ejecutamos el mandato `chmod` con los privilegios que deseamos.
4. Confirmamos la corrección de los cambios.

Básicamente podemos restringir nuestro directorio a los demás usuarios tanto como deseemos; la opción más restrictiva supone que nosotros tenemos todos los permisos y el resto de usuarios (y los de nuestro grupo) no pueden leer ni escribir (ni ejecutar) sobre nuestro directorio:

Esto es `rwx------` 111 000 000 ó 700 en octal

```
$ cd
$ cd ..
$ ls -ld alumno
drwxr-xr-x  2 alumno copa    512 Oct 17 21:53 alumno
$ chmod 700 alumno
$ ls -ld alumno
drwx------  2 alumno copa    512 Oct 17 21:53 alumno
$ cd
```

Existe otra forma de referenciar los permisos. Es mediante el llamado modo ``simbólico''. Es algo más complejo que el anterior. Los diferentes modificadores son:

- `u` permisos de usuario (propietario)
- `g` permisos de grupo
- `o` permisos de otros (público)
- `a` permisos de todos (usuario, grupo y otros)
- `=` asigna un permiso (inicializando el resto)
- `+` añade un permiso (a los permisos actuales)
- `-` elimina un permiso (de los permisos actuales)

Los tipos de permisos son los mismos que en el caso anterior: `r`, `w` y `x`

Si partimos de :

```
$ ls -l hola
-rw-r--r--  1 alumno copa     58 Oct 23 18:26 hola
$ chmod +x hola
$ ls -l hola
-rwxr-xr-x  1 alumno copa     58 Oct 23 18:41 hola
```

añade a (todos) los permisos iniciales, el permiso de escritura para todos, y es equivalente a `chmod a+x hola`; pero diferente de

```
$ chmod a=x hola
$ ls -l hola
---x--x—x  1 alumno copa      58 Oct 23 18:41 hola
```

ya que coloca todos los permisos de escritura pero inicializa los permisos que hubiera anteriormente.

Para volver al modo de partida que era `-rw-r--r--` podemos utilizar el modo binario

```
$ chmod 644 hola
```

o el modo simbólico

```
$ chmod +r,u+w hola
```

Hay que hacer notar que los argumentos del modo de protección de `chmod` tienen que estar juntos (sin espacios). Si hay varios argumentos, deben ir separados por comas.

ORDEN UMASK

Cuando se crea un archivo o un directorio se le dan unos permisos por defecto. Estos permisos pueden ser cambiados por el mandato `umask`. Este mandato sirve para especificar la máscara que determinará los permisos reales que van a otorgarse a los archivos creados a partir del momento en que se invoca el mandato `umask`.

El cálculo de la máscara de `umask` se efectúa de la siguiente forma:

```
   666    (valor de referencia)
 -644     (valor requerido)
-----------------------------------------
   022    (valor del argumento de umask para obtener los permisos reales)
```

Este mandato sólo tiene valor durante la sesión actual.

Si introducimos el mandato `umask` sin argumentos, nos devuelve el valor actual.

```
$ umask
022
```

ORDEN SU

Puede que en algunos casos un mismo usuario disponga de varias cuentas en un sistema Unix (es decir, que disponga de varios identificadores y contraseñas, cada uno con privilegios distintos y que deberá utilizar según lo que pretenda hacer con el sistema). Para permitir que un usuario ya conectado pueda cambiar de cuenta y adoptar así la identidad asociada a la otra cuenta, Unix facilita la orden `su`. Si esta orden es utilizada sin argumentos se entenderá que pretendemos adoptar la identidad del superusuario. Si no queremos hacer eso deberemos facilitar el login de la otra cuenta. Hecho esto, `su` nos pedirá la contraseña y en caso de darla correctamente, se arrancará un nuevo shell asociado con el UID y GID del usuario que acabamos de dar. Para volver a la situación original habrá que cerrar el shell generado, utilizando `exit`.

Utilice `su` para adoptar la identidad del otro miembro de su grupo de prácticas (Sustituya el identificador facilitado en el ejemplo por el que corresponda):

```
$ su alu2
Password:
```

Utilice la página de manual para ver las opciones complementarias de `su`.

LOCALIZAR ARCHIVOS

En puntos anteriores, se han descrito órdenes para manejar archivos. Ahora se trata de utilizar mandatos que permitan localizar su situación en el sistema de archivos. Con la orden `find` se pueden explorar partes del sistema de archivos, buscando aquellos que coincidan con un determinado nombre o tipo. Su sintaxis consiste en:

`find <directorio_búsqueda> <opciones de búsqueda> <acciones>`

Por ejemplo el mandato

```
$ find prueba1 –name archivo1 -print
....
```

buscará a partir del directorio `prueba1` todos aquellos archivos que coincidan con el nombre `archivo1` y los imprimirá por pantalla con el nombre de ruta obtenido. Con `find` se pueden utilizar metacaracteres para realizar búsqueda de archivos cuyo nombre exacto no se conoce, por ejemplo

```
$ find . –name “f*” -print
....
```

busca en el directorio actual los archivos que comiencen por la letra “f”. También pueden utilizarse otros criterios de búsqueda como la opción `–type d` que permite buscar directorios o `–user` para limitar a un determinado usuario la búsqueda. Para mayor información, se puede recurrir al manual (`man`).

Por ultimo las acciones a realizar, pueden ser además de `-print`, `-exec <cmd>` que permite aplicar el mandato `cmd` a los archivos que sean localizados o `–ok <cmd>` que antes de aplicar el mandato `cmd`, pide conformidad al usuario. Por ejemplo:

```
$ find . –name “f*” –exec rm {} \;
....
```

que borra todos los archivos que comiencen por la letra “f” a partir del directorio actual. El argumento { } que acompaña a la orden rm indica que ésta se aplicará a los archivos objeto de la búsqueda.

REFERENCIAS

BIBLIOGRAFÍA

- Syed Mansoor Sarwar: El Libro de Unix, Addison Wesley, 2002.
- F. Maciá Pérez y A. Soriano Payá: El Shell Korn. Manual de Usuario y Programador, Servicio de Publicaciones de la Universidad de Alicante, 1999.
- J. Tackett y D. Gunter: Linux 4ª Edición, Prentice may, 2000.

WEB

Existe una gran candidad de manuales y tutoriales de Unix y Linux en Internet que puedes localizar a partir de cualquier buscador (ej. www.google.com).

AUTOEVALUACIÓN

1. Borrar todo el contenido del directorio inicial (home).
2. Crear en el directorio inicial los siguientes archivos: `prueba1.txt, prueba2.txt, prueba3.txt, prueba4.txt, so1.asig, ec.asig, ib.asig.`
3. Visualizar la fecha del sistema.
4. Crear en el directorio inicial los siguientes directorios: `privado` (lectura, escritura y ejecución sólo el propietario) y `publico` (lectura, escritura y ejecución para el propietario, para el grupo nada y lectura y ejecución para el resto).
5. Cambiar al directorio `publico`.
6. Copiar los archivos de extensión `asig` al directorio `publico`.
7. Con una sola orden, copiar los archivos `prueba2.txt, prueba3.txt, prueba4.txt` al directorio `privado`.
8. Visualizar en qué directorio te encuentras actualmente.
9. Borrar los archivos `prueba2.txt, prueba3.txt, prueba4.txt` del directorio inicial.
10. Concatenar los archivos `prueba1.txt` con `prueba3.txt` y guardarlo en el directorio `publico`.
11. Mover los archivos del directorio `privado` al directorio `publico`.
12. Cambiar al directorio inicial.
13. Modificar los permisos del archivo `so1.asig` del directorio inicial para que lo pueda leer y escribir todo el mundo.
14. Visualizar el nombre de los archivos que empiezan por `s` de todo nuestro directorio inicial (incluyendo subdirectorios).
15. Listar el contenido del directorio `publico` ordenado por fecha de forma ascendente.
16. Cambiar al directorio `privado`.
17. Establecer los permisos de los archivos por defecto a lectura, escritura y ejecución para todo el mundo y sólo de escritura para el propietario.
18. Listar el contenido del directorio inicial ordenado por nombre de forma descendente.
19. Crear el archivo `¿contenido?` con el contenido completo del directorio inicial (archivos y directorios).
20. Mostrar el contenido del archivo `¿contenido?`.