Introducción a Unix

Primer contacto con Unix

Hace algunos años, comprar un ordenador de un determinado fabricante implicaba forzosamente tener que utilizar el sistema operativo que el propio fabricante había desarrollado para sus equipos. Afortunadamente, Unix ha terminado con la hegemonía de los sistemas propietarios. Hoy en día, Unix funciona sobre cualquier tipo de ordenador, independientemente de su marca y modelo. De hecho, muchos fabricantes han abandonado incluso sus vetustos sistemas propietarios.
Cuesta creer que este sistema operativo universal fuese ideado hace ya un cuarto de siglo por una sola persona, Ken Thompson de Bell Laboratories. Al haberse desligado su empresa de un ambicioso proyecto que nunca llegó a funcionar, Ken decidió ocupar su tiempo en diseñar un sistema operativo orientado fundamentalmente al desarrollo de programas. El resultado de aquella idea ha rebasado todo lo imaginable. Las siguientes líneas inician un curso sobre Unix. En ellas aprenderás nociones básicas sobre el uso de este sistema.

Usuarios y contraseñas

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

Desconexión del sistema

Para cerrar una conexión con una máquina Unix basta con teclear **`exit`** o **`Ctrl-D`**. En cualquier caso, Unix presentará el mensaje de despedida logout. Y, a continuación, desaparecerá la ventana correspondiente a la conexión.

El intérprete de comandos o shell

Una vez realizada una conexión a una máquina Unix ejecuta el programa intérprete de comandos, también conocido como shell. Existen diversas versiones de este programa. Para las prácticas se utilizará el intérprete conocido como **`bash`**. El intérprete de comandos desplegará algún indicador o prompt. Este indicador denota que el intérprete espera un comando del usuario desde el teclado con el fin de que Unix los ejecute. La orden se ejecuta al pulsar **`<ENTER>`**.
Ejemplos de comandos sencillos y a la vez muy útiles son el comando **`ls`** que lista los archivos del directorio actual y el comando **`ps`** que lista los procesos de un usuario. Se ha de considerar que cada comando 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, un descriptor de salida estándar para devolver los resultados del procesamiento y un descriptor de salida de errores que pueden ocurrir en su ejecución (salida de diagnóstico).

```
$ ls
$ ps
```

Tenga en cuenta al teclear los comandos, Unix distingue entre mayúsculas y minúsculas. El comando 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 ***historial*** que permite recuperar órdenes ejecutadas anteriormente con los cursores ▼ y ▲. Otras opciones interesantes son: **Ctrl–a** para ir al principio de la linea, **Ctrl–e** para ir al final, **Ctrl–d** para borrar el carácter sobre le que se encuentra el cursor y **Ctrl–r** para buscar un comando en el historial con la cadena que se especifique.

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 los comandos

Muchos comandos aceptan argumentos. Para Unix, el separador de argumentos es el espacio en blanco.
La mayoría de los comandos 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
```

Algunos comandos básicos de Unix

Esta parte de la práctica tiene como finalidad conocer algunos de los comandos básicos de Unix.

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 el comando **man**.
Así por ejemplo, podría haber obtenido la información correspondiente al propio comando **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**: abandonar
- **?** ó **h**: para ver otros comandos disponibles.

También existen otras dos posibilidades de obtener ayuda acerca de los comandos del sistema: **help** y **apropos**.
A PARTIR DE AHORA, RECUERDE UTILIZAR EL MANUAL CUANDO TENGA ALGUNA DUDA.

DATE

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
```

WHO

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
```

PWD

Cuando entramos en el sistema a través de nuestro nombre de usuario y password, el sistema nos sitúa sobre nuestro directorio. 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.

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 los archivos (uno por cada fila de información, excepto la primera), la ocupación en sectores de disco (la primera fila "total ???"). Hay dos archivos especiales que son el "." y "..". El archivo "." hace referencia al directorio actual y el ".." hace referencia al directorio padre.
Si en un momento dado queremos saber qué archivos en un directorio son ordinarios y cuáles son directorios, utilizamos la opción **–F**. Los archivos cuyo nombre acaba en **/** son directorios y los que acaban en ***** son ejecutables (código).
La distinción entre archivos ordinarios y directorios también se puede apreciar 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*** (ele) que hace referencia a un enlace (link). Un enlace es una referencia a un archivo que está físicamente en otro lugar (similar a un acceso directo de windows).
Creemos un enlace del archivo que en UNIX da el primer mensaje que sale a todos los usuarios, **/etc/motd** al entrar en su terminal.

```
$ ln –s /etc/motd mensaje
$ ls –l
....
lrwxr–xr–x 1 alumno copa 9 Oct 27 10:29 mensaje –> /etc/motd
```

El archivo **mensaje** es un enlace a **/etc/motd**
Un enlace no es más que una entrada de directorio que apunta a un archivo o directorio ya existente.
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, o bien a partir del directorio que le pasemos como argumento. Comentar por último que el comando **ls** tiene muchas más opciones que no son explicadas aquí. Para más información ejecutar **man ls**.

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 podemos cambiar de directorio mediante el comando **cd directorio**. Si no introducimos ningún nombre de directorio, el comando **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.

Ejemplo:

Si queremos consultar el contenido del archivo **passwd** 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:

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

Estos caracteres tienen un significado especial dentro de los comandos 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 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.

Ejemplos:

Vamos a utilizar la orden **ls**, aunque en principio los comodines se pueden aplicar a cualquier orden. Por ejemplo, pruébense los siguientes comandos sobre un directorio vacío:

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

Todas estas opciones pueden ser combinadas entre sí. El alumno debe probar diferentes combinaciones y observar los resultados.

El metacaracter *

El shell, no los comandos, interpreta este carácter antes de ejecutar un comando. 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 comando **echo**. Este comando envía a la salida estándar sus argumentos. Por ejemplo, pruébense, las siguientes líneas de comando

```
$ 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 comando en varias líneas:

```
$ echo ’*’
$echo “Els arxius “ * “estan en el directori actual.”
$echo “Els arxius \”*\” son “ * “i estan en el directori 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 comandos existen múltiples órdenes de visualización del contenido de archivos. En este punto vamos a practicar las más importantes.
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. De cualquier forma si queremos saber el tipo de un archivo podemos utilizar el comando **file** seguido del nombre de archivo del que queremos averiguar su tipo.

Ejemplo

Visualicemos el archivo de configuración del shell **/etc/profile**

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

Puesto que el archivo **/etc/profile** es de texto podemos visualizarlo:

```
$ 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 **fichero1** y **fichero2**, la orden:

```
$ cat fichero1 fichero2
```

lista en primer lugar el archivo **fichero1** y a continuación **fichero2**.
Una aplicación muy útil de **cat** es concatenar archivos. Por ejemplo, si queremos concatenar los dos archivos anteriores en un nuevo archivo llamado **fichero3** bastaría con ejecutar:

```
$ cat fichero1 fichero2 > fichero3
```

Un comando alternativo a **cat** es **more** que da más control que la anterior, ya que automáticamente lista un archivo y cuando llena la terminal se para, esperando que pulsemos la tecla espacio para reanudar la salida.

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

Además indica 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.

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
```

MKDIR

Para crear un directorio es necesario utilizar el comando **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 fichero1
–rw–r––r–– 1 alumno copa 423 Oct 27 10:14 fichero2
lrwxr–xr–x 1 alumno copa 9 Oct 27 10:29 mensaje –> /etc/motd
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 el comando **touch** (utilice **man**).
También podemos crear subdirectorios de la siguiente forma:

```
$ 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 fichero3 prueba2
a2 c2 fichero1 mensaje prueba3
bin c3 fichero2 prueba1
bin:
directorio1:
prueba1:
prueba11
prueba1/prueba11:
```

```
prueba2:
prueba21
prueba2/prueba21:
prueba3:
prueba31
prueba3/prueba31:
```

RMDIR

El comando **rmdir** elimina un directorio. Es necesario que dicho directorio esté vacío.

```
$ rmdir prueba1/prueba11
```

CP

Si queremos copiar un archivo utilizamos el comando **cp**. Por ejemplo, si queremos copiar el archivo de los caracteres ASCII que hemos utilizado anteriormente. El primer argumento del comando es el archivo origen y el segundo el destino. El archivo destino es físicamente diferente del origen.

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

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

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

Recordemos que . (punto) es nuestro directorio actual.
También podemos efectuar la copia a un directorio concreto

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

introduce el archivo en el directorio prueba3.
El comando **cp** también copia directorios. Para esto utilizamos el modificador recursivo **–R**.

```
$ cp –R prueba3 prueba4
```

MV

La finalidad del comando **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 que, si prueba4 fuese un directorio existente en el directorio actual, el siguiente comando

```
$ mv nuevo_profile prueba4
```

colocaría el archivo **nuevo_profile** en el directorio prueba4.
Para comprobarlo utilizar el comando **ls prueba4** que devolverá el contenido del directorio **prueba4**. La orden **mv** ha cambiado el archivo de lugar. Si ejecutamos el

comando **ls** directamente podremos observar que el archivo **nuevo_profile** ha desaparecido del directorio en el que se encontraba.
Los permisos del archivo copiado o movido son los mismos que los del archivo original.

RM

El comando **rm** suprime un archivo de un directorio. Si queremos borrar el archivo que habíamos creado anteriormente...

```
$ rm profile
```

OPCIONES

- ✓ **–i**: opción interactiva. Solicita la confirmación del usuario antes de proceder al borrado.

```
$ rm –i fichero1
rm: remove fichero1?
```

- ✓ **–r**: opción recursiva. Borra recursivamente todos los directorios y subdirectorios del nivel que estamos y de los niveles inferiores.

FIND

En puntos anteriores, se han descrito comandos para manejar archivos. Ahora se trata de utilizar comandos para ubicar archivos en el sistema. Con el comando **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 comando

```
$ find prueba1 –name fichero1 –print
....
```

buscará a partir del directorio prueba1 todos aquellos archivos que coincidan con el nombre **fichero1** 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.
Por ultimo las acciones a realizar, pueden ser además de **–print**, **–exec <cmd>** que permite aplicar el comando **cmd** a los archivos que sean localizados o **–ok <cmd>** que antes de aplicar el comando **cmd**, pide conformidad al usuario.

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

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.