Práctica

2

GESTIÓN Y UTILIZACIÓN DE REDES LOCALES

Curso 2001/2002

Introducción a Linux

Introducción

Linux es un sistema operativo totalmente compatible con **Unix**. Tradicionalmente los sistemas operativos Unix han sido desarrollados por el fabricante del equipo sobre el que debía ejecutarse el sistema operativo. De esta forma, **SUN Microsystems** tenía su propia versión de Unix, **Hewlett Packard** disponía de la suya propia, etc. Obviamente, estos fabricantes sacaban provecho de su sistema operativo, y por tanto, el código fuente no estaba disponible para los usuarios. Por otra parte, si una empresa necesitaba el sistema operativo Unix, debía comprarlo a uno de los fabricantes, así como el ordenador para el cual estaba diseñado ese sistema.

Afortunadamente, en los últimos años ha aparecido **Linux**, un sistema operativo totalmente compatible con Unix y que funciona sobre ordenadores **PC compatibles**. Además, Linux es un sistema abierto que cualquiera puede conseguir en **Internet** y del cual está disponible el código fuente. Por tanto, las mejores prestaciones de los sistemas Unix están ahora al alcance de cualquiera, y de forma totalmente gratuita. Esto, unido a la gran solidez de Linux hace que cada vez sean más las empresas que lo utilizan en sus sistemas, reemplazando a otros sistemas operativos como pueden ser **Windows 9x**, **Windows NT**, o **Unix**.

Linux (Unix) es un sistema **multiusuario**, es decir, varios usuarios pueden trabajar al mismo tiempo en el mismo equipo. Cada usuario tiene una cuenta propia, creada por el *superusuario* (root) del sistema, que es un usuario con privilegios especiales que se encarga de la administración del sistema.

Para poder entrar en un sistema Linux es necesario identificarse. Para ello debe introducirse el identificador de usuario (*login*) y la correspondiente contraseña (*password*).

Primeros pasos

Cuando se entra en un sistema Linux, el directorio en el que nos encontramos depende del nombre de usuario, o *login*, que hayamos empleado. Por lo general nos encontraremos en un directorio del estilo "**`/home/login`**", donde **`login`** es el nombre de usuario con el que hayamos accedido al sistema. En la figura de la derecha se puede observar el árbol de directorios que habitualmente vamos a encontrar en un sistema Linux. El directorio del cual cuelgan todos los demás se llama **directorio raíz**. El directorio **`home`** alberga los directorios de cada uno de los usuarios (**`gyurl00`** .. **`gyurl11`** en la figura).

pwd: Obtención del directorio actual

Podemos saber el directorio en el que nos encontramos con la orden **`pwd`** (escrita en minúsculas, pues en Linux las mayúsculas y las minúsculas no son intercambiables). Si ejecutamos **`pwd`** inmediatamente después de entrar en el sistema, lo que aparece es la ruta completa de nuestro directorio personal, empezando en el directorio raíz.

ls: Lista el contenido de un directorio

Para ver el contenido de un directorio debemos emplear la orden **`ls`**. Esta orden nos lista de forma condensada el contenido del directorio. Como en la mayoría de las órdenes de Linux, en **`ls`** existen diversas opciones. Por ejemplo, si queremos obtener un listado más detallado del contenido del directorio debemos emplear

la opción **l** (*long*). La sintaxis sería la siguiente: **ls -l**. El resultado que obtenemos tras ejecutar la orden **ls -l** será del estilo de:

```
drwxr-xr-x  16 gyurl01   users  31744 feb 18 15:33 .
drwxr-xr-x  16 gyurl01   users  31744 feb 18 15:33 ..
drwxr-xr-x  16 gyurl01   users  31744 feb 18 15:33 directorio_1
-rwxr-xr-x   8 gyurl01   users  18125 sep 20 18:19 fichero_1
-rwxr-xr-x   8 gyurl01   users   2622 jan 13 10:23 fichero_2
```

En el listado nos aparecen los ficheros del directorio (uno por cada fila de información). Hay dos ficheros especiales que son "." y "..". El fichero "." hace referencia al directorio actual y el fichero ".." hace referencia la directorio del cual cuelga el directorio actual (llamado directorio padre). El significado de los datos contenidos en cada fila es el siguiente:

- La columna de más a la derecha indica el **nombre del fichero o directorio**. Se puede observar que el listado que nos ha proporcionado este comando está ordenado por orden alfabético.
- Las tres columnas contiguas indican la **fecha y hora de la última modificación** del fichero. En el caso del directorio **directorio_1**, fue modificado por última vez el 18 de febrero, a las 15:33 horas.
- El número que tenemos a la izquierda de la fecha de última modificación es el **tamaño del fichero** o directorio. En el caso del fichero **fichero_1**, éste ocupa 18125 bytes.
- A continuación, siguiendo de derecha a izquierda, tenemos el **grupo y el propietario del fichero**. El propietario es, en este caso, el propio **gyurl01**, como era de esperar. El grupo al que pertenece **gyurl01** es **users**. En Linux, al igual que en el resto de sistemas Unix, los usuarios del sistema están organizados por grupos. Esta organización por grupos es útil a la hora de compartir información entre los distintos usuarios, como veremos a continuación. Esta estructura por grupos también está relacionada con la seguridad del sistema.
- Finalmente, la primera columna, que contiene "**drwxr-xr-x**", indica los **permisos asociados al fichero** o directorio en cuestión. La primera "**d**" significa que es un directorio. El resto de letras, "**rwxr-xr-x**", hacen referencia a los permisos para ese fichero, tema relacionado con la **seguridad**. Linux, al igual que el resto de sistemas Unix, mantiene un firme control de acceso para cada uno de los ficheros y directorios del sistema. Este control de seguridad se ha diseñado a tres niveles: lo que cualquier usuario del sistema puede hacer con uno de tus ficheros, lo que cualquier usuario que pertenece a tu mismo grupo puede hacer con ese fichero, y también lo que tú mismo puedes hacer con tu propio fichero.
  De la cadena "**rwxr-xr-x**", las tres primeras letras "**rwx**" indican que el propietario del fichero puede leer el fichero ("**r**" *read*), puede modificar el fichero ("**w**" *write*), y también puede ejecutarlo ("**x**" *execute*). En el caso de un directorio, puesto que no tiene sentido ejecutarlo, la "**x**" significa permiso para entrar en ese directorio.
  Las tres siguientes letras de la cadena de permisos, "**r-x**", se refieren a lo que otros usuarios de tu mismo grupo pueden hacer con tu fichero. En el caso de **directorio_1** sólo pueden leerlo y pasar dentro del directorio. Las tres últimas letras de los permisos indican lo que el resto de usuarios del sistema pueden hacer con ese fichero. En este caso concreto pueden leer y ejecutar los ficheros mostrados arriba.

Volviendo a la orden **ls**, podemos utilizar este comando para obtener un listado del contenido del directorio ordenado por fechas en vez de ordenado alfabéticamente. Para ello, debemos utilizar el comando "**ls -lt**". El parámetro "**l**" indica que queremos un listado "largo", mientras que el parámetro "**t**" indica que lo ordene temporalmente.

Otra opción interesante de la orden **ls** es la opcion "**a**". Esta opción se combina, generalmente, con la opción "**l**" (la orden sería **ls -la**). La opción "**a**" (*all*) hace incluir en el listado los ficheros ocultos (aquellos cuyo nombre empieza por ".").

Si utilizamos la orden "**ls -l**" seguida del nombre de un directorio, la orden nos proporciona el contenido del directorio. En ocasiones podemos no querer ver el contenido del directorio sino el propio directorio. Para forzar a que así sea debemos usar la opción "**d**". Por ejemplo, si tecleamos "**ls -ld Desktop**" obtendremos información del propio directorio y no de los ficheros que contiene.

A veces resulta interesante referirse a ficheros que tengan características comunes en su nombre. Por ejemplo, podemos necesitar listar todos los ficheros que comiencen por **ab** seguido de cualquier cosa. Otras veces quizá sea necesario listar los ficheros que empiecen por cualquier cosa y acaben por "**.c**", por ejemplo. En Linux esto se puede conseguir utilizando un carácter especial, "*****", que representa una cadena de caracteres cualesquiera. Por ejemplo, podemos listar todos los ficheros que contienen una "**y**" en su nombre. La orden a teclear sería "**ls -ld *y***".

cd: Cambio de directorio

Podemos movernos por el sistema de ficheros con la orden "**cd**". Si ejecutamos la orden "**cd ..**" (el espacio entre **cd** y los dos puntos es necesario) pasaremos al directorio anterior (o directorio padre). Por otra parte, podemos pasar a cualquier otro directorio para el que tengamos permisos de acceso con la orden "**cd nombre_directorio**".

Trata de acceder al directorio de algún compañero, y también muévete por el árbol de directorios del sistema. Si en algún momento se te deniega el acceso, comprueba que realmente no tienes permiso para entrar a ese directorio. Un directorio interesante es **/usr/bin**, donde se encuentra gran parte de los programas ejecutables del sistema.

En los sistemas Unix (Linux entre ellos) hay un directorio llamado **tmp** que cuelga directamente de la raíz del árbol de directorios (**/tmp**). En este directorio todos los usuarios tienen permiso de escritura. Se trata de una especie de cajón de sastre, donde todos los usuarios pueden almacenar de forma temporal sus propios ficheros. Es habitual que el administrador del sistema borre periódicamente el contenido de este directorio, por lo que no debemos utilizarlo para almacenar ficheros de forma indefinida. Además de los diferentes usuarios, el sistema operativo también usa este directorio para almacenar temporalmente algunos de sus ficheros.

Después del paseo por el árbol de directorios del sistema, podemos volver a nuestro propio directorio con el comando "**cd**". Podemos emplear esta orden de tres formas diferentes cuando queremos volver a nuestro directorio:

- La primera es tecleando la ruta completa de nuestro directorio personal. En este caso el comando sería del estilo "**cd /home/login**", donde **login** es vuestro nombre de usuario.
- Otra forma más sencilla es usando una variable del entorno, que apunta a nuestro directorio personal. Esta variable se llama **HOME**. Podemos ver su contenido tecleando "**echo $HOME**" (el símbolo de dólar es necesario). Para volver a nuestro directorio daríamos la orden "**cd $HOME**".
- La tercera forma de volver a nuestro directorio es la más sencilla. Tecleando simplemente "**cd**" el sistema operativo se da por enterado de que queremos volver a nuestro directorio personal (o, como se diría habitualmente, a nuestro *home*).

Entorno gráfico

A diferencia de Windows 9x, que es un sistema operativo gráfico, la base sobre la que está diseñado Linux no es gráfica[1]. De esta forma, esta primera toma de contacto con Linux la

[1] El hecho de que Linux no esté desarrollado sobre un interfaz gráfico lo hace más robusto frente a posibles errores del interfaz.

hemos hecho en la pantalla a la que nos ha llevado el sistema cuando hemos entrado en él. Esta pantalla (o terminal) trabaja en modo texto. Este tipo de terminales son muy sencillas, pero tienen el inconveniente de no poder proveer los servicios necesarios para ejecutar aplicaciones gráficas, como **Netscape** (prueba a ejecutar la orden **netscape** y observa lo que sucede).

A pesar de no estar diseñado sobre un interfaz gráfico, Linux también incorpora un entorno gráfico similar al de Windows 9x. Para entrar en dicho entorno hay que teclear la orden **startx**. Una vez dentro del entorno podemos ejecutar tanto aplicaciones orientadas a texto (como hemos hecho en la sección anterior) como aplicaciones gráficas. Para poder seguir con la práctica es necesario que abras una ventana de *shell*, pinchando en el icono remarcado en la figura siguiente.

La ventana de *shell*, como puede comprobarse, es similar a la terminal de texto utilizada anteriormente, aunque con algunas mejoras. Ahora las posibilidades son mayores. Podemos cambiar el tamaño de letra si pinchamos con el botón derecho en la ventana para acceder al menú de propiedades. También podemos hacer más grande la ventana, para tener más área de trabajo. De esta forma, al listar el contenido de un directorio podemos retener más líneas en la ventana. En caso de que algunas líneas hayan desaparecido por la parte superior, con el *scroll* que hay en la parte derecha podemos acceder a ellas. También podemos hacer *scroll* con la combinación de teclas "**SHIFT+Re Pag**" y "**SHIFT+Av Pag**". Otra posibilidad es la de copiar y pegar texto. Con el botón izquierdo del ratón podemos seleccionar texto en nuestra ventana de *shell*. Para pegarlo debemos pulsar el botón del centro (en caso de ratones con dos botones se deben pulsar los dos botones al mismo tiempo). Esta capacidad de copiar y pegar texto es especialmente útil cuando tenemos que teclear el nombre de un fichero y este nombre es muy largo; copiando y pegando no tenemos ningún problema.

Gestión de ficheros y directorios

mkdir: Creación de directorios

Los directorios en Linux se crean con la orden **mkdir**. A continuación de la orden debemos teclear el nombre del directorio (o directorios) a crear. Por ejemplo, podríamos crear un directorio en nuestro *home* de nombre dir_1 con la orden **mkdir dir_1**. Para crear varios directorios, con nombres dir_1, dir_2 y dir_3, el comando sería **mkdir dir_1 dir_2 dir_3**.

También se puede utilizar esta orden para crear directorios que cuelguen de otros directorios en los cuales no estamos actualmente dentro. Por ejemplo, si estamos en nuestro *home* y queremos crear un nuevo directorio dentro del directorio **/tmp**, la orden sería **mkdir /tmp/dir_1**. Prueba a crear un directorio en **/usr**. ¿Qué sucede? ¿Por qué?

rmdir: borrado de directorios

Para borrar un directorio se utiliza la orden **rmdir directorio**. Es necesario que el directorio esté vacío. Si no lo estuviera, el sistema nos devolvería un mensaje de error informándonos.

cp: Copia de archivos

Podemos copiar archivos en Linux utilizando la orden **cp**. El primer argumento es el fichero origen y el segundo argumento es el fichero destino. Así por ejemplo, para copiar el fichero **hosts** del directorio **/etc** a nuestro *home* la orden sería: **cp /etc/hosts $HOME**.

La orden cp también permite copiar directorios (con su contenido). Para ello debemos utilizar la opción **r**. La sintaxis de la orden sería: **cp -r dir_existente dir_nuevo**.

mv: Moviendo archivos

La orden **mv** mueve archivos de un directorio a otro. También puede mover directorios completos en el árbol de directorios. Si se utiliza para mover un fichero dentro del mismo directorio, el efecto que se consigue es cambiarle el nombre. Así, por ejemplo **mv fich1 fich2** cambia el nombre de **fich1** a **fich2**.

rm: Borrado de archivos

La orden **rm** se utiliza, principalmente, para borrar ficheros. También se puede combinar con la opción **r** para borrar un directorio y todo lo que contiene, aunque este uso resulta peligroso, por la cantidad de información que podemos borrar. Hay que tener en cuenta que una vez borrado un fichero (o un directorio) no hay forma de recuperarlo.

Prueba a borrar alguno de los ficheros que hay en el directorio **/bin**, ¿Qué sucede?.

chmod: Protección de ficheros

Hemos comentado en la descripción de la orden **ls** que los ficheros y directorios en un sistema Linux tienen un propietario. También tienen una serie de permisos asociados con ellos Se ha mencionado que de la cadena "**rwxrwxrwx**" asociada a cada fichero y directorio, las tres primeras letras indican lo que el propietario del fichero puede hacer con él, las tres segundas letras se refieren a lo que otros usuarios del mismo grupo que el propietario pueden hacer con ese fichero, y las tres últimas letras se aplican al resto de usuarios del sistema. Esto constituye el sistema de permisos (y por lo tanto de protección) de este archivo.

Los permisos asociados con cada fichero y directorio pueden ser modificados utilizando la orden **chmod**. Su sintaxis es **chmod permisos fichero**. Para especificar los permisos hay varias formas. Quizá la más sencilla sea la que se basa en la representación binaria. Se trata de representar cada uno de los tres grupos de tres permisos como un número en binario mediante unos y ceros en función de si un permiso está activado o no. Por ejemplo, los permisos "**rwxr-xr--**" se podrían dividir en los tres grupos correspondientes (propietario, grupo, todos), dando lugar a los siguientes grupos: "**rwx** ", "**r-x**", "**r--**". Estos permisos, representados en binario dan lugar a 111, 101 y 100, que en decimal dan lugar a la combinación 754. Así pues, para cambiar los permisos de un fichero a "**rw-r-----**" la orden correspondiente sería **chmod 640 fichero**. Obviamente, para poder cambiarle los permisos de acceso a un fichero o directorio, debemos ser los propietarios de él.

Prueba a cambiarle los permisos al directorio **Desktop** de tu *home*, de forma que los nuevos permisos sean "**---------**", o sea, **000**. ¿Qué sucede si ahora intentas entrar en él con la orden "**cd Desktop**"?

Edición de ficheros

Linux permite, al igual que todos los sistemas operativos, editar el contenido de los ficheros de texto ascii. Podemos utilizar diversos editores de texto. Algunos de ellos poseen una interfaz gráfica, como **xemacs**, **nedit** u otros.

Otros editores de texto están especialmente pensados para ser utilizados en una terminal de texto. Quizá el más extendido sea el editor "**vi**". Este editor es tan viejo como los mismos sistemas Unix, y en consecuencia es un tanto arcaico e incómodo de usar, una vez acostumbrados a editores del estilo de Word. No obstante, en la época en la que se inventó fue toda una revolución. La utilidad que tiene hoy en día es que ha llegado a ser un

estándar, estando presente en todos los sistemas Unix. Por tanto, conocerlo nos permite acceder de forma sencilla a muchos recursos de una gran variedad de sistemas.

La forma de invocar esta orden es "**`vi nombre_fichero`**". Invoca al editor usando el nombre de fichero que quieras.

El editor vi tiene dos modos de funcionamiento: el **modo comando** y el **modo edición**. Cuando entramos en el editor, éste se pone en modo comando, a la espera de que le demos alguna orden. Las órdenes más comunes son:

- **i:** pone el editor en modo edición, en el que podemos introducir texto
- **x:** borra el carácter sobre el que se encuentra el cursor
- **dd:** borra la línea sobre la que se encuentra el cursor
- **u:** deshace la última acción
- **yy:** copia al buffer interno la línea sobre la que se encuentra el cursor
- **p:** pega el buffer interno en el lugar donde se encuentra el cursor
- **:x :** sale del editor, grabando el fichero editado
- **:w :** graba el contenido del fichero que estamos editando
- **:q! :** sale del editor sin grabar el fichero editado
- **:r nombre_fichero :** lee un fichero y lo pega en el lugar donde está el cursor

Para pasar de modo edición a modo comando debemos pulsar la tecla de escape (**`Esc`**). Por otra parte, para movernos por el fichero podemos usar las flechas del teclado.

Edita un fichero, de al menos 40 líneas, donde cada línea contenga un número, que debe coincidir con el número de línea. Este fichero nos será útil en la siguiente sección.

Visualización de ficheros

more: Mostrando ficheros página a página

A menudo resulta necesario simplemente ver el contenido de un archivo, sin llegar a editarlo. La orden **`more`** nos permite precisamente esto. Dado un fichero de texto llamado **`fichero`**, podemos ver su contenido utilizando la orden **`more fichero`**. Por otra parte, esta orden permite ver de forma cómoda ficheros extensos, pues en el caso de que el fichero ocupe más de una pantalla, esta orden se para tras presentar una pantalla completa, esperando a que le indiquemos que queremos continuar. Prueba a mostrar el fichero que has editado en la sección anterior.

Por otra parte, esta orden se utiliza a menudo combinada con la orden **`ls`** para visualizar el contenido de directorios con muchos archivos. Efectivamente, en el caso de directorios extensos, al usar la orden **`ls`** (**`ls -l`**) es muy probable que la respuesta proporcionada por la orden se escape por la parte superior de la pantalla. Combinando la orden **`ls`** con la orden **`more`** se puede evitar esto, obligando al sistema a que se pare cada vez que se rellene una pantalla y espere nuestra indicación antes de seguir mostrando más ficheros. La sintaxis de la orden es: **`ls -l | more`**. La barra vertical, también llamada tubería o *pipe*, y obtenida con la combinación de teclas **`Alt_Gr + 1`**, envía la salida de la orden **`ls`** a la orden **`more`**, de forma que esta última la visualice de pantalla en pantalla.

less: Lo contrario de more

La orden less es similar a la orden more, pero resulta más potente, pues no sólo visualiza la información pantalla a pantalla, sino que permite movernos por ella hacia adelante, hacia atrás y hacia la derecha e izquierda (en caso de que las líneas sean más largas que el ancho de pantalla) usando las teclas de cursor. Por otra parte, permite hacer búsquedas dentro de la información presentada, utilizando el comando interno "**`/`**".

La sintaxis de la orden es la misma que en el caso anterior: **`less fichero`**. Visualiza el fichero creado en la sección anterior. Trata de moverte por él y de buscar el carácter "**`1`**", utilizando el comando "**`/`**".

tail: Viendo el final del fichero

En ocasiones puede resultar interesante ver únicamente las últimas líneas de un fichero. La orden **`tail`** permite hacerlo, pudiendo incluso especificar cuántas de las últimas líneas se quiere ver. Por ejemplo, la orden **`tail +25 fichero`** mostraría las últimas 25 líneas del fichero. Utiliza esta orden aplicándola al fichero creado anteriormente.

cat: Concatenar ficheros

La orden **`cat`** permite concatenar varios ficheros de texto en uno solo. Su sintaxis es: **`cat fich1 fich2 fich3 fichN > fichResultado`**. Esta orden parte de varios ficheros (**`fich1 fich2 fich3 fichN`**) y los concatena uno detrás de otro, guardando el resultado en **`fichResultado`**. El símbolo "**`>`**" redirecciona la salida de la orden **`cat`**, que iría destinada a la pantalla, a un fichero. Este símbolo de redirección se puede usar con cualquier otra orden de Linux.

La orden **`cat`** permite concatenar varios ficheros, aunque generalmente no se usa con este fin, sino que habitualmente se usa para presentar en pantalla el contenido de un único fichero. Así, por ejemplo, la orden **`cat fich1`** volcaría el contenido del fichero **`fich1`** en pantalla. Si el fichero en cuestión no tiene una longitud excesiva, la orden **`cat`** resulta más cómoda de usar que las órdenes **`more`** o **`less`**.

grep: Búsqueda dentro de un fichero

El comando **`grep`** permite buscar en un fichero de texto una cadena de caracteres determinada, especificada como argumento de la orden. Se puede utilizar para extraer información de los archivos, buscar líneas en las que haya una cadena de caracteres concreta o para, incluso, localizar archivos que contengan una palabra en especial.

La sintaxis de esta orden es: **`grep expresion_a_buscar fichero`**. Si la cadena a buscar contiene más de una palabra (es una frase, por ejemplo), entonces hay que encerrar la expresión de búsqueda entre comillas.

Las opciones más interesantes de esta orden son:

- **`-i:`** la búsqueda no diferencia entre mayúsculas o minúsculas.
- **`-n:`** muestra el número de línea donde se ha encontrado la coincidencia.
- **`-l:`** muestra los nombres de fichero que contienen el patrón de búsqueda pero no las líneas.

¿Qué obtienes con la orden "**`grep login .*`**"?

Localización de ficheros

locate: Búsqueda de ficheros en bases de datos

En Linux existen órdenes para localizar un fichero determinado dentro de la estructura de directorios del sistema. Uno de estos comandos es **`locate`**. Su sintaxis es **`locate fichero`**, donde **`fichero`** es el nombre del fichero buscado. La salida de esta orden es una lista de ficheros en cuyo nombre está incluido el nombre de fichero buscado.

El funcionamiento de la orden **`locate`** se basa en una búsqueda en la base de datos que el sistema mantiene con la localización de todos los ficheros del disco duro. Habitualmente esta base de datos se actualiza automáticamente todos los días. No obstante,

si por algún motivo la actualización de esta base de datos dejara de hacerse, obviamente la búsqueda se realizaría entre unos datos viejos. En cualquier caso el sistema informa de esta eventualidad si ocurriera.

Prueba a buscar el fichero **ps2pdf**. ¿Qué obtines?¿Obtienes algo si buscas el fichero que has creado en el editor **vi**?¿Por qué?

find: Búsqueda de ficheros en un árbol de directorios

find es otra orden para buscar ficheros dentro del sistema de ficheros de nuestro equipo. A diferencia de **locate**, **find** no basa su búsqueda en la consulta a una base de datos, sino que realiza una búsqueda exhaustiva por los directorios del sistema que hayamos especificado. Por tanto, los resultados proporcionados por **find** siempre están actualizados. El inconveniente es que la búsqueda es más lenta.

La sintaxis es: **find directorio_inicial opciones_de_búsqueda acciones**, donde **directorio_inicial** es el directorio a partir del cual empieza la búsqueda, **opciones_de_búsqueda** especifican las condiciones que deben cumplir los ficheros que se buscan, y **acciones** indica qué hacer con los ficheros que cumplen las con las opciones de búsqueda.

La opción de búsqueda más habitual es **-name nombre_fichero**. Esta opción busca aquellos ficheros cuyo nombre coincide con **nombre_fichero**. Por otra parte, la acción más habitual es **-print**, que muestra los ficheros encontrados.

Prueba a buscar los mismos ficheros que has buscado con la orden locate de la sección anterior. Como directorio inicial usa el directorio raíz (**/**). ¿Obtienes los mismos resultados?

Gestión de procesos

ps: Informe del estado de los procesos

Los sistemas Linux son sistemas multitarea, es decir, pueden realizar más de una tarea simultáneamente. Cada una de estas tareas, o procesos, consumirán más o menos recursos, entre ellos memoria y procesador. Para ver la lista de procesos que están en ejecución en el sistema podemos usar la orden "**ps**". Si usamos esta orden sin ningún parámetro, nos devolverá la lista de procesos que nosotros mismos tenemos en marcha. Con los parámetros "**aux**" nos dará una lista completa de todos los procesos que el sistema está ejecutando.

En esta lista, la columna **PID** indica el identificativo del proceso (Process IDentifier). La columna **size** indica el tamaño del proceso en kilobytes. Puesto que el sistema seguramente tendrá memoria virtual, la columna **RSS** nos indica cuánta memoria RAM está consumiendo este proceso, o dicho de otra forma, qué cantidad de la memoria total del proceso está residente en memoria principal.

Las columnas **START** y **TIME** indican cuando se inició el proceso y cuanto tiempo lleva en ejecución.

Utiliza la orden **ps** combinada con la orden **grep** para ver qué procesos está ejectuando el usuario **root** (utiliza para ello la barra vertical llamada tubería o *pipe*).

top: Procesos ordenados por consumo de CPU

Podemos ver una lista de los procesos ordenada por consumo de CPU, que se actualiza en tiempo real. Para ello debemos usar la orden "**top**". Para salir de este comando debemos pulsar la tecla "**q**".

kill: Matando procesos

Esta orden se usa principalmente para matar a un proceso que está en ejecución (que obviamente debe ser nuestro, puesto que no podemos matar procesos de otros usuarios). Para usarla debemos conocer el **PID** del proceso en cuestión (esto lo podemos averiguar con la orden **ps**). La sintaxis de esta orden, cuando se usa para matar procesos, es: **kill -9 pid**.

Prueba a matar, de tus procesos, alguno que se llame **bash**.

Órdenes relacionadas con la red

ping: Probando la ruta a un destino

La orden **ping** sirve para comprobar que existe una ruta entre nuestra máquina y un destino dado, y además que el destino está operativo. Esta orden se vió con detalle en la práctica primera.

traceroute: Conociendo la ruta a un destino

La orden **traceroute** sirve para conocer la ruta entre nuestra máquina y un destino dado. Esta orden es similar a la orden **tracert** vista en la práctica primera.

finger: Información sobre usuarios remotos

En Internet hay conectadas muchas máquinas que utilizan el sistema operativo Linux (o Unix). Como ya se ha visto, este tipo de máquinas admiten varios usuarios diferentes. Para obtener información acerca de estos usuarios de máquinas remotas podemos usar la orden **finger**. Su sintaxis es **finger usuario@maquina**. Por ejemplo, podemos preguntar sobre el usuario **root** a la máquina **pleione.cc.upv.es**, o a la máquina **aldebaran.cc.upv.es**. Si no se especifica un nombre de usuario, entonces la orden nos da información acerca de todos los usuarios conectados en ese momento.

Dado que la orden **finger** proporciona infomación personal sobre los usuarios de una máquina (si tiene o no correo, si está o no conectado, cuándo fue la última vez que se conectó, etc), es habitual que los administradores de las máquinas remotas no permitan que se saque este tipo de información de ellas, y por tanto deshabilitan este servicio. Prueba a usar **finger** contra la máquina **www.uvp.es**.

Otras órdenes comúnmente usadas

man: Manual On-Line

Linux (y Unix) proporciona un sistema de ayuda *on-line*, o manual, el cual podemos consultar para averiguar cómo usar una orden concreta. Este manual describe, para las diferentes órdenes del sistema, su sintaxis y sus opciones, proporcionando una descripción detallada de las mismas. Para consultar el manual hay que teclear "**man comando**", donde **comando** es la orden acerca de la cual buscamos ayuda. Para salir del manual hay que pulsar la tecla "**q**". La tecla "**h**" nos proporciona una lista de otros posibles mandatos cuando estamos utilizando el manual.

Este manual a menudo nos proporciona tanta información que es difícil encontrar de forma rápida lo que buscamos. Si queremos una ayuda más concisa, la mayoría de comandos son capaces de proporcionar una breve ayuda acerca de sí mismos. La opción que hay que emplear generalmente es "**--help**" (hay que poner los dos guiones, no es una

errata). Así, si tecleamos "**ls --help**", obtendremos una breve pero útil ayuda del comando **ls**.

date: Consulta de la fecha y hora

Esta órden nos proporciona la fecha y hora del sistema. **date** también sirve para modificar la hora del sistema. Prueba a modificar la fecha y hora del sistema. Para conocer la sintaxis exacta consulta la ayuda (bien con **man date**, bien con **date --help**).

who: ¿Quién?

Puesto que en un sistema Linux puede haber varios usuarios al mismo tiempo, es posible pedirle al sistema una relación de los usuarios que en estos momentos están conectados a él. Para ello debemos usar la orden "**who**". Esta orden nos devuelve la lista de usuarios conectados y también cuándo se han conectado. Hay una versión más potente de esta orden, llamada "**w**", que además nos dice desde dónde está conectado cada usuario, cuánto tiempo ha pasado desde la última vez que pulsó una tecla, cuánta CPU está consumiendo, etc.

Salir de Linux

Para salir de nuestra sesión en Linux debemos emplear la orden **exit**. Con esta orden cerramos nuestra sesión de trabajo y dejamos el puesto libre para que otro usuario se conecte a la máquina Linux. Si estamos en el entorno gráfico, entonces debemos salir del entorno antes de teclear la orden **exit**.

Si además de salir de nuestra sesión queremos apagar el ordenador, **NUNCA** debemos hacerlo pulsando el botón de encendido (o el de reset), sino que debemos pulsar la combinación de teclas **CTRL+ALT+SUPR**. De esta forma el sistema se apagará de forma ordenada.