5. INTRODUCCIÓN AL LINUX

Un **shell** es un entorno para la relación con el sistema basados en modo texto. Este modo de interactividad se basa en la ejecución de una serie de comandos, que son interpretados por un programa o *shell*. Linux dispone de varios de estos programas o *shells,* los más habituales son tcsh, bash, bsh y ksh. Si Linux se ha arrancado en modo texto, el sistema arranca de directamente el *shell* y queda a la espera de introducción de nuevos comandos.

5.1. INTRODUCCIÓN

Hay una serie de nociones básicas que hay que tener en cuenta a la hora de introducir los comandos. En primer lugar citaremos las siguientes:

- los comandos hay que teclearlos exactamente.
- las letras mayúsculas y minúsculas se consideran como diferentes.
- en su forma más habitual (los *shells* bsh, bash, ksh, …), el sistema operativo utiliza un signo de ***$*** como *prompt* para indicar que está preparado para aceptar comandos, aunque este carácter puede ser fácilmente sustituido por otro u otros elegidos por el usuario. En el caso de que el usuario acceda como administrador (usuario *root*) este signo se sustituye por el carácter **#.**
- cuando sea necesario introducir el nombre de un fichero o directorio como argumento a un comando, Linux permite escribir las primeras letras del mismo y realiza un autorrellenado al presionar la tecla del tabulador (*Tab*). Si no puede distinguir entre diversos casos rellenará hasta el punto en el que se diferencien. Por ejemplo, supongamos una carpeta con los siguientes directorios:

  Programas
  Documentos_proyecto
  Documentos_privados

  Al escribir **cd Pr<tab>**, Linux rellenará el resto del contenido hasta escribir **cd Programas**. Por el contrario, al escribir **cd D<tab>** se escribirá **cd Documentos**_

Otra nociópn básica es el directorio personal, es un directorio, con un determinado nombre asignado a un usuario. Los directorios personales habitualmente son subdirectorios del directorio **/usr/users.** Generalmente el nombre del directorio de cada usuario coincide con su *nombre de usuario*, aunque puede no ser así, y varios usuarios pueden estar trabajando en el mismo directorio. Cada usuario de Linux puede crear una estructura en árbol de subdirectorios y archivos tan compleja como desee bajo su directorio personal, pero normalmente nunca fuera de él.
Así, si por ejemplo si somos el usuario *jorge,* posiblemente nuestro directorio personal sea **/usr/users/jorge**. Dentro de este directorio podemos crear un árbol de directorios para organizar nuestros propios ficheros y documentos.

5.2 ALGUNOS COMANDOS SENCILLOS DE LINUX

Para efectuar el cambio o la introducción de un **password** o *contraseña* se utiliza el comando **passwd**. El proceso a seguir es el siguiente:

passwd
(current) UNIX password: (se teclea la contraseña actual; no aparece en pantalla)
New UNIX password: (se teclea la nueva contraseña; no aparece en pantalla)
Retype new UNIX password: (se teclea de nuevo la nueva contraseña comprobando que se ha tecleado bien. Si no coincide no se produce el cambio).

A continuación se describen algunos comandos sencillos del sistema.

date
Muestra por pantalla el día y la hora.

cal 1949
Muestra el calendario del año 1949.
cal 05 1949
Muestra el calendario de mayo de 1949.

who
Indica qué usuarios están conectados al sistema en ese momento, en qué terminal están y desde qué hora llevan conectados al sistema.

whoami
Indica cuál es la terminal y la sesión en la que se está trabajando.

man <comando>
Linux posee manuales disponibles desde dentro del propio sistema operativo. Esta orden permite acceder a la información correspondiente al comando <*comando*>. Por ejemplo, con **man who** aparecerá por pantalla, y de forma formateada por páginas, la explicación del comando **who**. Se puede navegar a través de estas páginas con los cursores del teclado, y presionar **q** para salir.

clear
Este comando limpia la consola.

5.3. COMANDOS MÁS UTILIZADOS

5.3.1. LISTADO DEL CONTENIDO DE DIRECTORIOS: COMANDO LS

Una de las acciones más habituales a la hora de trabajar es mostrar el contenido de un directorio. Gráficamente, esto es muy sencillo y existen numerosas

herramientas gráficas para explorar los contenidos de un directorio. No obstante, el *shell* incluye un programa o comando con este mismo fin: **ls**.

Este comando muestra los nombres de los ficheros y subdirectorios contenidos en el directorio en el que se está (o en el directorio *dir* si se ejecuta **ls <dir>**). Sólo se obtienen los nombres de los ficheros, sin ninguna otra información.
Para ver todas las opciones que permite este comando se puede consultar el manual con **man ls**. No obstante, a continuación se indican algunos de los usos más frecuentes.

- **ls -a** muestra todos los ficheros, incluyendo algunos que ordinariamente están ocultos para el usuario (aquellos que comienzan por un punto). Se recuerda que el fichero punto . indica el directorio actual y el doble punto .. el directorio padre que contiene al actual.

- **ls -l** es la opción de listado detallado. Muestra toda la información de cada fichero, incluyendo protecciones, tamaño, fecha de creación, del último cambio introducido, ...

- **ls -c** muestra los archivos ordenados por día y hora de creación.

- **ls -t** muestra los archivos ordenados por día y hora de modificación.

- **ls -r** muestra el directorio y lo ordena en orden inverso.

- **ls -l fichero** muestra toda la información relativa a *fichero*.

- **ls --color** muestra el contenido del directorio coloreado.

Las opciones anteriores pueden combinarse. Por ejemplo, **ls -cr** muestra el contenido del directorio ordenado inversamente por fechas.
El comando **ls** admite los caracteres de sustitución o metacaracteres (***) y (***?***). El carácter ***** representa cualquier conjunto o secuencia de caracteres. El carácter ***?*** representa cualquier carácter, pero sólo uno.
Por ejemplo:

- ls *.gif muestra todos los nombres de ficheros que acaben en *.gif,* por ejemplo *dib1.gif, a.gif,* etc.

- ls file? muestra todos los ficheros cuyos nombres empiecen por *file* y tengan un nombre de cinco caracteres, por ejemplo: *file1, file2, filea,* etc.

5.3.2. CREACIÓN DE DIRECTORIOS. COMANDO MKDIR

El comando **mkdir** (*make directory*) permite a cada usuario crear un nuevo directorio. Por ejemplo, **mkdir subdir1** crea un directorio de nombre *subdir1* como subdirectorio del directorio actual.

5.3.3. BORRADO DE SUBDIRECTORIOS. COMANDO RMDIR

Este comando borra uno o más directorios del sistema (*remove directory*), siempre que estos subdirectorios estén vacíos. Por ejemplo, **rmdir subdir1** elimina el directorio de nombre *subdir1* (si está vacío).

5.3.4. CAMBIO DE DIRECTORIO. COMANDO CD

Este comando permite cambiar de directorio actual. Podemos hacer el cambio utilizando una *ruta absoluta* (es decir, empezando la ruta por el directorio raíz /). Por ejemplo, **cd /home/pedro** pasa del directorio actual de trabajo al nuevo directorio **/home/pedro**, que será desde ahora nuestro nuevo directorio de trabajo. Otra forma de hacer el cambio es indicando una ruta relativa (relativa al directorio actual de trabajo). Así, **cd dir1** Nos traslada al subdirectorio *dir1* (que deberá existir como subdirectorio del directorio actual).

Con **cd ..** retrocedemos un nivel en la jerarquía de directorios. Por ejemplo, si estamos en */home/pedro* y usamos este comando, pasaremos al escalafón inmediatamente superior de la jerarquía de directorios, en este caso a */home*.
Nota*: al contrario que en MS-DOS, en Linux no existe la forma* ***cd..*** *sin espacios entre* ***cd*** *y los dos puntos.*

El comando **cd** nos sitúa nuevamente en el directorio personal del usuario (por ejemplo, */home/luis*).

5.3.5. SITUACIÓN ACTUAL. COMANDO PWD

El comando **pwd** (*print working directory*) visualiza o imprime la ruta absoluta del directorio actual de trabajo. Este comando es uno de los pocos que no tiene opciones.

5.3.6. COPIA DE FICHEROS. COMANDO CP

Este comando tiene la siguiente forma: **cp file1 file2**. Lo que hace es una copia del fichero *file1* a la que llama *file2*. Si *file2* no existía, lo crea con los mismos atributos de *file1*. Si *file2* ya existía antes de hacer la copia, su contenido queda destruido y es sustituido por el de *file1*. El fichero
file2 estará en el mismo directorio que *file1,* a menos que demos una ruta distinta como en **cp file3 /home/pedro/copia_file3**. Así, tanto *file1* como *file2* indican el

nombre de un archivo, que pueden incluir la ruta al mismo si alguno de ellos no se encuentra en el directorio actual.
Otra posibilidad es **cp file1 file2 namedir** que hace copias de *file1* y *file2* en el directorio *namedir*.

5.3.7. TRASLADO Y CAMBIO DE NOMBRE DE FICHEROS. COMANDO MV

Este comando tiene una forma similar al anterior, **mv file1 file2.** El comando **mv** realiza la misma función que el anterior (*cp*) pero lo que hace es mover el fichero, no copiarlo. Es decir, se destruye el fichero original y sólo queda el fichero destino. En definitiva traslada el contenido de *file1* a *file2*; a efectos del usuario lo que se hace es cambiar el nombre de *file1*, pasándose a llamar *file2*. Podemos indicar una ruta para mover le fichero a otro lugar del árbol de directorios.
De igual forma, **mv file1 file2 namedir** traslada uno o más ficheros (*file1*, *file2*,...) al directorio *namedir,* conservándoles el nombre.
El comando **mv namedir1 namedir2** cambia el nombre del subdirectorio *namedir1* por *namedir2*.
Es decir, permite cambiar de nombre (mover) ficheros y directorios.

5.3.8. ENLACES A FICHEROS. COMANDO LN

En Linux, un mismo fichero puede ser accesible desde varios sitios y tener distintos nombres o alias. No se trata de tener varias copias de un mismo fichero, sólo existe un fichero pero hay varios enlaces o accesos al mismo. A veces, esto es práctico para poder acceder a un mismo fichero desde más de un directorio. En Linux, esto recibe el nombre de enlaces múltiples a un fichero. El ahorro de espacio de disco es importante al poder compartir un fichero más de un usuario. Estos enlaces son muy prácticos a la hora de utilizar ficheros que pertenecen a directorios distintos. Gracias a los enlaces se puede acceder a muchos ficheros desde un mismo directorio, sin necesidad de copiar en ese directorio todos esos ficheros.
Para crear enlaces se utiliza la orden **ln file1 file2.** A partir de este momento el fichero *file1* tiene dos nombres: *file1* y *file2*. A diferencia de los comandos **cp** y **mv**, este comando toma más precauciones, ya que advierte previamente si el nombre *file2* está ocupado, y en este caso no se ejecuta.
Por ejemplo, al ejecutar **ln panacea subdir/panacea,** el fichero *panacea* tendrá el mismo nombre, y además estará accesible desde dos sitios distintos: en el directorio actual y en el subdirectorio *subdir* (en ambos sitios accesible mediante el nombre *panacea*).
Los ficheros enlazados a otro se borran como los ficheros normales. Si se borra el fichero original permanece su contenido en los ficheros enganchados. El contenido del fichero sólo se borra cuando se borran todos los enlaces al fichero.

5.3.9. BORRADO DE FICHEROS. COMANDO RM

Este comando tiene las formas siguientes:

• **rm file1 file2** elimina uno o más ficheros (en este caso *file1* y *file2*) de un directorio en el cual tengamos permiso de escritura. Con este comando resulta facilísimo borrar ficheros inútiles, y desgraciadamente, también los útiles.
Por eso es conveniente y casi imprescindible emplear lo opción ***-i***, como se indica en el siguiente caso.

• **rm -i file1 file2** pedirá confirmación para borrar cada fichero de la lista. Se recomienda usar siempre este comando con esta opción para evitar el borrado de ficheros útiles.
Por ejemplo, si se teclea **rm -i superfluo** aparecerá en pantalla el aviso siguiente:

remove superfluo?

y habrá que contestar ***y*** (yes) o ***n*** (not). En este comando se pueden utilizar los caracteres de sustitución mencionados anteriormente (***** y ***?***). Así, puedo poner **rm fich*** para borrar todos los ficheros contenidos en el directorio actual que comiencen por *fich*.
Con **rm *** se borrarán todos los ficheros del directorio actual, mientras que **rm -i *** realiza una labor similar, pero con previa confirmación.

5.3.10. CARACTERÍSTICAS DE UN FICHERO. COMANDO FILE

Este comando realiza una serie de comprobaciones en un fichero para tratar de clasificarlo. El formato de este comando es **file fich**. Tras su ejecución, este comando muestra el tipo del fichero e información del mismo.

5.3.11. CAMBIO DE MODO DE LOS FICHEROS COMANDOS CHMOD, CHOWN Y CHGRP

Los permisos de cada fichero se pueden ver con el comando **ls -l**. Para cambiar los permisos de un fichero se emplea el comando **chmod**, que tiene el formato **chmod [quien] oper permiso files.**

• *quien* indica a quien afecta el permiso que se desea cambiar. Es una combinación cualquiera de las letras ***u*** para el usuario, ***g*** para el grupo del usuario, ***o*** para los otros usuarios y ***a*** para todos los anteriores. Si no se especifica este campo, el sistema supone ***a***.

• *oper* indica la operación que se desea hacer con el permiso. Para dar un permiso se pondrá un **+**, y para quitarlo se pondrá un **-**.

• *permiso* indica el permiso que se quiere dar o quitar. Será una combinación cualquiera de las letras *r,w,x,s*.

- *files* son los nombres de los ficheros que son afectados por este nuevo cambio de permisos.

Por ejemplo, para quitar el permiso de lectura a los usuarios de un fichero el comando es:

chmod a -r fichero.txt

Los permisos de lectura, escritura y ejecución tienen un significado diferente cuando se aplican a directorios y no a ficheros normales. En el caso de los directorios el permiso ***r*** significa la posibilidad de ver el contenido del directorio con el comando **ls**; el permiso ***w*** da la posibilidad de crear y borrar ficheros en ese directorio, y el permiso ***x*** autoriza a buscar y utilizar un fichero concreto.

Por otra parte, el comando **chown** se emplea para cambiar de propietario (***ch****ange* ***own****er*) a un determinado conjunto de ficheros. Este comando sólo lo puede emplear el actual propietario de los mismos. Los nombres de propietario que admite Linux son los nombres de *usuario*, que están almacenados en el fichero **/etc/passwd**. La forma general de uso del comando **chown** es la siguiente:

chown newowner file1 file2 ...

Análogamente, el grupo al que pertenece un fichero puede ser cambiado con el comando **chgrp**, que tiene una forma general similar a la de **chown**, (**chgrp newgroup file1 file2**...)
Los grupos de usuarios están almacenados en el fichero **/etc/group**.

5.4. ESPACIO OCUPADO EN EL DISCO COMANDOS DU, DF Y QUOTA

El comando **du** permite conocer el espacio ocupado en el disco por un determinado directorio y todos los subdirectorios que cuelgan de él. Para usarlo basta simplemente colocarse en el directorio adecuado y teclear **du**. Este comando muestra el espacio de disco utilizado (en bloques). Para obtener la información en bytes se debe emplear el comando con la opción *–h* (**du –h**).
El comando **df**, por el contrario, informa del espacio usado por las particiones del sistema que se encuentren montadas.
El comando **quota**, muestra el disco usado y los límites dispuestos.

5.5. VISUALIZACIÓN DE FICHEROS

5.5.1. VISUALIZACIÓN SIN FORMATO DE UN FICHERO. COMANDO CAT

Este comando permite visualizar el contenido de uno o más ficheros de forma no formateada. También permite copiar uno o más ficheros como apéndice de otro ya existente. Algunas formas de utilizar este comando son las siguientes:

- **cat filename** saca por pantalla el contenido del fichero *filename*.

- **cat file1 file2** saca por pantalla, secuencialmente y según el orden especificado, el contenido de los ficheros indicados.

- **cat file1 file2 > file3**, el contenido de los ficheros *file1* y *file2* es almacenado en *file3*.

- **cat file1 file2 >> file3**, el contenido de los ficheros *file1* y *file2* se añade (*append*) al final de *file3*.

- **cat >file1** acepta lo que se introduce por el teclado y lo almacena en *file1* (se crea *file1*). Para terminar se emplea **<ctrl>d.**

5.5.2. COMANDO HEAD

El comando **head -7 filename** escribe las 7 primeras líneas del fichero *filename.*

5.5.3. VISUALIZACIÓN DE FICHEROS CON FORMATO. COMANDO PR

El comando **pr**, a diferencia de **cat**, imprime por consola el contenido de los ficheros de una manera formateada, por columnas, controlando el tamaño de página y poniendo cabeceras al comienzo de las mismas. Está muy en relación con el comando **lp** de salida por impresora. Las formas más importantes que admite son las siguientes:

- **pr file** produce una salida estándar del fichero *file* de 66 líneas por página, con un encabezamiento de 5 líneas (2 en blanco, una de identificación y otras 2 líneas en blanco).

- **pr -ln file** produce una salida de *n* líneas por página (útil cuando el tamaño de papel de impresora, por ejemplo, tiene un número de líneas distinto de 66).

- **pr -p file** hace una pausa para presentar la página, hasta que se pulsa **<return>** para continuar

- **pr -t file** suprime las 5 líneas del encabezamiento y las del final de página para el fichero *file*.

- **pr -wn file** ajusta la anchura de de la línea del fichero *file* a *n* posiciones.

- **pr -d file** lista el fichero *file* con espaciado doble.

- **pr -h `caracteres` file** donde el argumento o cadena de caracteres `*caracteres*` se convierten en la cabecera del listado de *file*.

- **pr +n file** imprime el fichero *file* a partir de la página *n*.

Además de los ejemplos anteriores, se pueden combinar varias opciones en un mismo comando como por ejemplo en **pr -dt file**.
La salida de este comando es por la consola, pero puede redireccionarse a otro fichero. Si ejecutamos la orden **pr file1 > file2** se crea un fichero nuevo llamado *file2* que es idéntico a *file1*, pero con formato por páginas y columnas.

5.5.4. VISUALIZACIÓN DE FICHEROS PANTALLA A PANTALLA. COMANDOS MORE Y LESS

Estos comandos permiten visualizar un fichero pantalla a pantalla. Con el comando **more**, el número de líneas por pantalla es de 23 líneas de texto y una última línea de mensajes, donde aparecerá la palabra *more*. Cuando se pulsa la barra espaciadora (el espacio en blanco), se visualizará la siguiente pantalla de contenido. Para salir de este comando (terminar la visualización) se pulsa **q** o **<ctrl>d**.
Para ejecutar este comando sobre un fichero *file* podemos ejecutar la orden **more file**.
El comando **less** es muy similar al anterior pero permite el desplazamiento a lo largo del texto empleando las teclas de cursores, pudiendo desplazarse hacia arriba o abajo de un fichero.

5.6. BÚSQUEDA EN FICHEROS. COMANDOS GREP, FGREP Y EGREP

El comando **grep** localiza una palabra, clave o frase en un conjunto de directorios, indicando en cuáles de ellos la ha encontrado. Este comando rastrea fichero por fichero, por turno, imprimiendo aquellas líneas que contienen el conjunto de caracteres buscado. Si el conjunto de caracteres a buscar está compuesto por dos o más palabras separadas por un espacio, se colocará el conjunto de caracteres entre comillas simples (').
Su formato es el siguiente:

grep 'conjuntocaracteres' file1 file2 file3

siendo *'conjuntocaracteres'* la secuencia de caracteres a buscar, y *file1*, *file2*, y *file3* los ficheros donde se debe buscar.

Veamos un nuevo ejemplo:

grep 'TRIANGULARIZACION MATRIZ' matrix.f scaling.f

Este comando buscará la cadena *TRIANGULARIZACION MATRIZ* entre las líneas de los ficheros *matrix.f* y *scaling.f.* Este comando permite seleccionar, entre todas las

líneas de uno o más ficheros, aquellas que contienen un motivo que satisface una expresión regular determinada.

grep [-opcion] expresión_regular [referencia...]

Las opciones principales son:

- *c*, lo único que se hace es escribir el número de las líneas que satisfacen la condición.
- *i*, no se distinguen mayúsculas y minúsculas.
- *l*, se escriben los nombres de los ficheros que contienen líneas buscadas.
- *n,* cada línea es precedida por su número en el fichero.
- *s,* no se vuelcan los mensajes que indican que un fichero no se puede abrir.
- *v,* se muestran sólo las líneas que no satisfacen el criterio de selección.

A continuación se muestra una serie de ejemplos.

grep ‘^d’ text → líneas que comienzan por *d*.
grep ‘^[^d]’ text → líneas que no comienzan por *d*.
grep -v ‘^C’ file1 > file2 → quita las líneas de *file1* que comienzan por *C* y lo copia en *file2*.

5.7. BÚSQUEDA AVANZADA EN FICHEROS. EXPRESIONES REGULARES

A veces se desea encontrar las líneas de un fichero que contienen una palabras o palabras determinadas. Cuando el texto que se desea encontrar es único, lo que hay que hacer es ponerlo tal cual en la sección del comando que define la búsqueda, por ejemplo

grep "PATATAS" Lista_de_la_compra.txt

Sin embargo, en otras ocasiones el texto que se desea buscar no es único, es decir, no está unívocamente determinado como en el ejemplo anterior, sino que debe cumplir unas ciertas condiciones, como la de estar escrito con mayúsculas, comenzar por determinado carácter, estar a principio o final de línea, etc. Este problema se puede resolver en muchos comandos de ***Linux*** por medio de las expresiones regulares que se van a presentar a continuación.

Las *expresiones_regulares* son una forma de describir patrones para la búsqueda de unas determinadas líneas dentro de uno o más ficheros ASCII. Se trata pues de encontrar las líneas cuyo contenido cumple ciertas condiciones, que se definen en la *expresión_regular*.

5.7.1. CARACTERES ESPECIALES

En una *expresión_regular* se pueden utilizar algunos caracteres que tienen un significado especial. Son los siguientes:

[comienzo de la definición de un conjunto de caracteres
. un carácter cualquiera, excepto el <eol>

*	un conjunto de caracteres cualesquiera, excepto el primer carácter de una expresión o inmediatamente después de la secuencia \(
]	terminación de la definición de un conjunto de caracteres
-	sirve para definir el conjunto de caracteres que van del que le precede al que le sigue. Si va detrás del [o delante del], no es especial
^	comienzo de línea, si está al comienzo de la expresión
^	conjunto complementario (el que no cumple la condición), si está justo después del [que abre la definición de un conjunto fin de línea, cuando está al final de una expresión
\	quita el significado especial a un carácter, excepto en la definición de un conjunto de caracteres.

5.7.2. EXPRESIONES REGULARES DE UN SOLO CARÁCTER

Se trata de buscar palabras o conjuntos de un solo carácter, que cumple ciertas condiciones. A continuación se presentan algunos ejemplos:

*	representa el carácter *
.	cualquier carácter, excepto el <eol>
[a-f]	un carácter cualquiera entre la a y la f
[A-Z]	cualquier letra mayúscula
[^a-d]	cualquier carácter que no sea una letra entre la a y la d
[:clase:]	donde *clase* puede ser: *digit* (cifra del 0 al 9), *xdigit* (cifra hexadecimal), *alpha* (letra cualquiera), *upper* (letra mayúscula), *lower* (letra minúscula), *alnum* (letra o dígito cualquiera), *space* (un espacio en blanco), *cntrl* (carácter de control), *punct* (un carácter de puntuación) y *print* (carácter imprimible).

5.7.3. EXPRESIONES REGULARES GENERALES

Se pueden formar de acuerdo con las siguientes reglas:

- Una expresión regular de un sólo carácter, por ejemplo, [a-z] da cualquier letra minúscula.
- Una expresión regular de un sólo carácter, seguida del carácter *, representando entonces todas las palabras de longitud positiva o nula que se pueden construir con los caracteres aceptados por la una expresiones regulares de un sólo carácter, por ejemplo, [a-z]* da cualquier palabra escrita con minúsculas.
- Concatenando (poniendo una a continuación de la otra) dos expresiones regulares construidas previamente, por ejemplo, [a-z][A-Z] da cualquier palabra de dos letras, de las cuales la primera es minúscula y la segunda mayúscula.
- Una expresión_regular definida en la forma \(expresiones regulares\) representa la propia expresiones regulares (es decir, definida ella sola), pero define una forma de referirse luego a esa expresiones regulares. En efecto las expresiones regulares definidas de esta forma quedan

afectadas por un número del 1 al 9, y es posible luego hacer referencia a una expresiones regulares por medio del número que le corresponde, en la forma \número. Si \número va seguido de un *, esa subexpresión puede repetirse un número cualquiera de veces. Por ejemplo, ^\(.*\)\1\1 da al comienzo de la línea, un campo formado por un carácter cualquiera que se repite las veces que sea, volviendo a aparecer dos veces mas antes de que se acabe la línea. Esta expresión detectaría las líneas que contienen palabras (o conjuntos de palabras) triples.

- Una expresión regular de un sólo carácter seguida de \(*num*\) representa *num* apariciones consecutivas de alguno de los caracteres aceptados por la expresiones regulares de un sólo carácter, donde *num* debe ser un valor entero. Si va seguida de \(*num*,\) representa un número mínimo de *num* apariciones consecutivas. Si va seguida de \(*num1*, *num2*\) representa un mínimo de *num1* y un máximo de *num2* apariciones consecutivas. Los números *num1* y *num2* deben ser enteros comprendidos entre 0 y 255.
- Toda expresión que comienza con ^ indica que los caracteres buscados deben estar a comienzo de la línea, por ejemplo, ^[a-z]* selecciona las líneas que sólo contienen letras minúsculas

5.8. COMPRESIÓN. COMANDOS TAR Y GZIP

Tanto el comando **tar** como **gzip** son ampliamente empleados para la difusión de programas y ficheros en Linux. El primero de ellos sirve para agrupar varios ficheros en un solo fichero, mientras que el segundo sirve para comprimir un fichero. En conjunto estos dos programas actúan de forma muy similar a programas como *Winzip* para Windows.

Para crear un nuevo archivo se emplea el comando:

tar –cvf nombre_archivo.tar fichero1 fichero2 …

donde *fichero1*, *fichero2,* … son los ficheros que se van a añadir al archivo resultante de ejecutar el comando **tar**. Si se desea extraer los ficheros de un archivo empaquetado con tar, se emplea el comando:

tar –xpvf nombre_archivo.tar fichero1 …

Al contrario que **tar** (que agrupa varios ficheros en uno), **gzip** comprime un único fichero con lo que la información se mantiene pero se reduce el tamaño del mismo. El uso de **gzip** es muy sencillo: **gzip fichero**.

Este comando comprime *fichero* y crea un fichero con nombre *fichero.gz*.

Si lo que se desea es descomprimir un fichero previamente comprimido, se emplea entonces la orden **gzip –d fichero.gz**, con lo que se recupera el fichero inicial. Como se ha comentado al principio, es típico emplear **tar** y **gzip** de forma consecutiva, para obtener ficheros con extensión **tar.gz** o **tgz** que contienen realmente varios ficheros de forma comprimida (similar a un fichero **zip** de Windows). El comando **tar** incluye la opción *z* para estos ficheros comprimidos, de forma que para extraer los ficheros que contiene un fichero llamado *fichero.tar.gz*, ejecuto la orden siguiente:

tar –zxf fichero.tar.gz

5.9. REDIRECCIONES Y TUBERÍAS

5.9.1. REDIRECCIONES

Los comandos de Linux tienen una entrada estándar (*stdin*, identificada con el número 0) y dos salidas estándar (*stdout*, identificada con el número 1 para la salida normal del comando, y *stderr* identificada con el número 2 para la salida de los mensajes de error que se puedan producir en su ejecución).
Por defecto tanto la entrada como las salidas estándar de los comandos son la propia terminal, a no ser que por la propia naturaleza del comando se den en él los nombres de algunos ficheros que hagan el papel de entrada y de salida.
Por ejemplo, en el comando **cp file1 file2**, *file1* es la entrada y *file2* es la salida, es decir, aquí no intervienen las entradas y salidas estándar. Sin embargo, cuando utilizamos, por ejemplo, el comando **ls**, la salida de este comando se dirige hacia el terminal. Si queremos que la salida de este comando se dirija a un fichero llamado *file*, podemos hacerlo escribiendo **ls >file**. El operaor (**>**) es uno de los llamados operadores de redirección y dirige la salida estándar hacia el fichero indicado a continuación; si este fichero no existe, se crea en ese momento.
Otros operadores de redirección son el operador (**<**) que redirige la entrada estándar desde un determinado fichero, y el operador (**>>**) que redirige la salida estándar hacia otro fichero, pero en modo *append* (añadiendo dicha salida al final
del fichero, sin sobrescribir el contenido original).
Por ejemplo, si cada vez que entramos en el sistema ejecutamos el comando **date >> archivo**, tendremos un fichero llamado *archivo* que va a contener la información sobre todas las veces que hemos entrado en el sistema.
Otro ejemplo, para añadir el fichero *file2* al final de *file1* y llamar al resultado *file3*, podemos hacer **cat file1 file2 >file3**. Si quisiéramos que el fichero resultante fuera el propio *file1*, entonces haremos **cat file2 >>file1**.
Un ejemplo en redirección a la entrada podría ser el siguiente:
mail juan <carta
que envía al usuario *juan* el contenido del fichero *carta*.

5.9.2. TUBERÍAS

Siguiendo con los ejemplos anteriores, si quisiéramos enviar a *juan* una lista de nuestros ficheros podríamos utilizar los comandos:
ls >fichero
mail juan <fichero
rm fichero

Es decir que hemos conectado la salida estándar de **ls** con la entrada estándar de **mail**, a través de un fichero transitorio *fichero*. Para estos casos, Linux permite hacer esta operación directamente, sin pasar por el fichero de

almacenamiento transitorio. Esto se hace mediante el uso del concepto de tubería (*pipe*), que consiste en "empalmar" la salida estándar de un comando con la entrada estándar de otro.
Para el ejemplo anterior esto se hace en la forma **ls | mail juan.**
Con el operador de tubería (|) se pueden empalmar tantos comandos como se desee.

5.9.3. BIFURCACIÓN O T (COMANDO TEE)

A veces interesa que la salida de un comando, además de redirigirse a un determinado fichero, se bifurque también hacia la terminal con el objeto de observar inmediatamente el resultado. Esto se consigue con el operador **tee**, que podría emplearse de la siguiente forma:
ls | tee file
donde la salida de **ls** se bifurca hacia el terminal y también hacia *file*.
Si quisiéramos que la salida de este comando se añadiera al final del fichero *file*, deberíamos utilizar la opción ***–a***, es decir, **ls | tee -a file**

5.9.4. REDIRECCIÓN DE LA SALIDA DE ERRORES

Los mensajes de error se dirigen a la salida número 2, que normalmente es también el terminal. A veces, por ejemplo, cuando se quiere ejecutar un comando en segundo plano o *background* (ejecutar un comando en *background* es lanzar su ejecución y recuperar el control de la terminal sin esperar a que termine, lo
cual se hace añadiendo el carácter ***&*** al final del comando, lo vemos en la siguiente sección), interesa evitar que los mensajes de error aparezcan en la pantalla, pues en ella habremos empezado a hacer otra cosa y no queremos que nos salga nada.
Supongamos, por ejemplo, que queremos compilar y enlazar en background un conjunto de ficheros, dirigiendo los listados de los mensajes de salida a un fichero llamado *listados*, y los mensajes de error a un fichero llamado *errores*. Esto lo podemos hacer de la siguiente manera:
gcc prueba.c 2>errores
con lo que la salida 2 (errores) se redirige hacia el fichero *errores*. Para redirigir la salida estándar de errores al mismo fichero que la salida estándar se emplea un comando como el siguiente:
program <datos.d >resultados.r 2>&1

5.10. EJECUCIÓN DE PROGRAMAS

5.10.1. EJECUCIÓN EN BACKGROUND (&), KILL, NICE Y NOHUP

Para ejecutar un programa en background, es decir, recuperando inmediatamente el control del terminal tras el lanzamiento a ejecución, basta añadir el carácter ***&*** al final del comando de ejecución:
program <datos.d >resultados.r &
Con esto, tendremos control inmediatamente del terminal. Al ejecutar esta orden, aparecerá en el terminal un número que es el número de proceso o *pid* del proceso que acaba de lanzarse como consecuencia de la ejecución en de este programa. Para detener definitivamente dicha ejecución (no se puede detener temporalmente) se puede utilizar el comando **kill**, con el formato **kill pid_proceso**, donde *pid_proceso* es el *pid* o identificador o número de proceso del proceso que queremos detener. La ejecución de un programa en background no impide que aparezcan en la pantalla los mensajes de error que se produzcan (a no ser que se haya redirigido la salida de errores), y que el programa se pare cuando se salga del sistema. Para que el programa continúe ejecutándose aun cuando nosotros hayamos terminado la sesión, hay que utilizar el comando **nohup** al lanzar el programa, **nohup program*.***

Si no se utilizan redirecciones, todas las salidas del programa se dirigen a un fichero llamado *nohup.out*. Cuando se utiliza **nohup** el ordenador entiende que el usuario no tiene prisa y automáticamente disminuye la prioridad de la ejecución del proceso. Existe un comando, **nice**, que permite realizar ejecuciones con baja prioridad, es decir, se le indica al ordenador que puede ejecutar de forma más lenta esta aplicación si existen otras que sean más urgentes. Este comando se utiliza de las formas:
nice program &
nice nohup program &

Para darle al programa la prioridad mínima habría que utilizar el comando **nice -19 program &**, ya que el -19 indica la mínima prioridad.

5.10.2. COMANDO TIME

El comando **time**, precediendo a cualquier otro comando, suministra información acerca de:
- el tiempo total empleado en la ejecución,
- el tiempo de CPU utilizado en la ejecución,
- el tiempo de CPU consumido en utilizar recursos del sistema.

Por ejemplo, para saber el tiempo utilizado en la compilación y enlazado del programa *prueba.c,* utilizamos el comando **time gcc prueba.c**

5.10.3. COMANDO TOP

Linux incluye una aplicación, que se invoca con el comando **top**, cuya finalidad es permitir la manipulación de procesos (programas en actual ejecución) de manera interactiva. Esta aplicación muestra una lista de los procesos que se están ejecutando. Los principales comandos que permite **top** son:

- *u*, que muestra los procesos que pertenecen a un determinado usuario.

- *k*, equivalente al comando **kill** para detener o matar un proceso.

- *h*, que muestra la ayuda del programa.