TEMA 11: SISTEMA OPERATIVO LINUX. COMANDOS. REDIRECCIONAMIENTO. FILTROS. TUBERÍAS

TEMA 11: SISTEMA OPERATIVO LINUX. COMANDOS. REDIRECCIONAMIENTO. FILTROS. TUBERÍAS

11.1 FORMATO DE LAS ÓRDENES EN LINUX

Una vez iniciado el sistema y abierta una terminal, Los usuarios teclean comandos en el prompt del shell. El prompt por defecto es el signo de dólares ($) para un usuario cualquiera y el símbolo # para el superusuario o root.

La sintaxis de la línea de comandos, formato de una orden o comando en Linux, es:

orden [-modificador] [parámetros] [&] [;]

Donde:

- **orden** es el programa o comando que el usuario va a ejecutar. Algunas veces el comando es representativo de la función, por ejemplo el comando ls es una contracción de la primera y tercera letras de la palabra "list".
- **-modificadores:** los modificadores permiten modificar el comportamiento de la orden. Siempre van precedidos de un guión – y pueden agruparse varios de ellos. por ejemplo el comando ls lista el contenido de un directorio, mientras que el comando ls -l da una lista Larga del directorio y ls -C proporciona la salida en Columnas.
- **parámetros** son normalmente nombre de ficheros, directorios, usuarios...
- **&** indica al shell que podemos ejecutar la orden en segundo plano pudiendo así lanzar mas tareas mientras esta se ejecuta.
- **;** con este signo podemos poner varias órdenes en la misma línea.

Para iniciar la ejecución del comando, después de la sintaxis anterior se debe presionar la tecla <ENTER>, entonces el shell interpreta y ejecuta la línea de comandos.

11.2 OBTENER AYUDA EN LINUX

Existen múltiples y variadas formas de obtener ayuda en un sistema Linux. A continuación se describen algunas de ellas:

- El comando **man** formatea y despliega un manual bastante amplio acerca de comandos, formatos de ficheros de configuración, llamados al sistema, etc.
  - Ejemplo: $ **man chmod**
- Muchos comandos poseen una opción para mostrar una ayuda breve acerca de su utilización. Esta opción usualmente es **-h**, **--help** ó -?.
  - Ejemplo: $ **ls --help**

- El programa **info** despliega información acerca de comandos, en ocasiones más amplia que la que brinda man. Las páginas de info poseen una estructura arbórea (nodos), a través de las cuales se puede navegar con ayuda de comandos especiales.
    - Ejemplo: $ **info ls**
- El comando **whatis** realiza una búsqueda en las secciones del man y muestra la descripción abreviada del comando en cada una de las secciones que aparezca. Este comando trabaja con una base de datos que se actualiza periódicamente y se crea con el comando makewhatis.
    - Ejemplo: $ **whatis shadow**
- Otra ayuda muy útil consiste en escribir las primeras letras de un comando y pulsar dos veces el tabulador. Linux nos mostrara un listado de todos los comandos que comienzan con las letras que hemos escrito.

11.3 METACARACTERES EN LINUX

La utilidad de los metacaracteres es la de indicar al shell que se quede con los archivos que se ajustan a determinados patrones. Los metacaracteres del Shell son:

***** Sustituye a cualquier carácter o conjunto de caracteres.

? Sustituye un solo carácter.

[..] El uso de corchetes permite hacer referencia a un solo carácter, las posibilidades son:

- hacer referencia a un solo carácter pero con la obligatoriedad de estar comprendido en los valores listados entre corchetes.
- hacer referencia a un rango de valores separados por guión.

Se pueden mezclar entre ellos, ejemplo:

$ ls ed? [11-9] *

nos mostraría todos los ficheros cuyo nombre de fichero verifique:

1. Sus dos primeros caracteres son “ed”.
2. El tercer carácter puede ser cualquiera.
3. El cuarto carácter es un número comprendido entre 11 y 9.
4. El resto de caracteres pueden ser cualesquiera.

11.4 COMANDOS PARA MANIPULAR DIRECTORIOS Y FICHEROS

Comando ls

El comando ls permite listar el contenido de un directorio.

Sintaxis: ls [opciones] [directorio | fichero]

Algunas opciones:

-l : muestra la salida en formato largo, mostrando los permisos, nº de enlaces, propietario, grupo, tamaño, tiempo de modificacion y el nombre de fichero o directorio ordenados por su nombre.

-R: lista recursivamente un directorio.

-a : lista además los ficheros ocultos (sus nombres comienzan con punto).

-h : muestra el tamaño de los ficheros en forma más legible (Ej.: 16M, 4k, etc.)

-t: clasifica por tiempo de modificación.

-u: clasifica por tiempo de último acceso.

-r: invierte el orden de clasificación.

Ejemplos:

$ ls /home/juanito

$ ls -l /home/juanito

$ ls -hl /home/juanito

$ ls –al /home/juanito

Un grupo de opciones que se suele utilizar bastante es ls –la

Comando cd

El comando cd se utiliza para cambiarnos de directorio actual.

Sintaxis: cd [directorio]

Ejemplos:

$ cd /home/pepe

$ cd **cambia al directorio home del usuario.**

$ cd ~ **cambia al directorio home del usuario.**

Comando mkdir

El comando mkdir se utiliza para crear directorios.

Ejemplos:

$ mkdir /home/alumno

$ mkdir –p /home/alumno/som/linux **se crean los directorios intermedios si es necesario**

Comando rm

El comando rm se utiliza para borrar directorios y ficheros. Si vamos a borrar un directorio este debe estar vacío. Si el directorio a borrar no estuviera vacío habría que utilizar el modificador –r, ahora se comentará:

Sintaxis:

rm [opciones] <ficheros | directorios>

Algunas opciones:

-r : borra recursivamente un directorio.

-i : ejecuta el comando de forma interactiva.

Ejemplo:

$ rm -ri /home/pepe

Comando pwd

El comando pwd indica el camino absoluto del directorio en el cual nos encontramos actualmente.

Ejemplo:

$ **pwd** **nos devuelve algo como /home/pepe/som/practicas**

Comando mv

El comando mv mueve un fichero hacia otro, o varios ficheros hacia un directorio. Este permite a su vez renombrar ficheros o directorios.

Sintaxis: mv [opciones] <fuente> <destino>

mv [opciones] <ficheros> <directorio>

Algunas opciones:

-i : ejecuta el comando de forma interactiva, o sea, pregunta ante de sobrescribir el destino si existiera.

-u : actualiza (upgrade) el destino con el fuente solo si este es más reciente.

Ejemplos:

$ mv /home/alumno/som /home/alumno/sistemas	**renombra el directorio som.**
$ mv -i /home/alumno/*.txt /home/alumno/sistemas	**mueve los ficheros terminados en .txt al directorio sistemas**

Comando touch

Este comando permite actualizar la fecha de un archivo a la fecha actual y si el archivo no existe, lo crea vacío.

Sintaxis:

touch <nombre fichero>

Ejemplo:

$ touch /home/alumno/archivo1.txt	**Crea un archivo en blanco llamado archivo1.txt**

Comando cp

El comando cp permite copiar un fichero en otro, o varios ficheros en un directorio.

Sintaxis:

cp [opciones] <fuente> <destino>

Algunas opciones:

-R : copia recursivamente un directorio.

-i : utiliza una forma interactiva (pregunta antes de sobrescribir el destino).

Ejemplos:

$ cp /home/alumno/sistemas/*.txt /home/juanito	**copia los ficheros con extensión txt al directorio home/juanito**

Comando stat

El comando stat muestra las características de un fichero. Por ejemplo: su nombre, permisos, tamaño en bytes, número del i-nodo que lo representa, las fechas de modificación y acceso, el tipo, el dueño, el grupo, etc.

Ejemplo**:**

$ stat /home/alumno/sistemas/archivo1.txt

Comandos more y less

Los comandos more y less paginan (dividen en páginas) uno o varios ficheros y los muestran en la terminal (pantalla). Se diferencian en las facilidades que brindan. Por ejemplo more es más restrictivo en cuanto al movimiento dentro del texto, mientras que less no limita este aspecto pues acepta el empleo de todas las teclas de movimiento tradicionales. Cuando se alcanza el final del último fichero a paginar, more termina automáticamente, no así less. También more muestra sucesivamente el porcentaje del fichero visto hasta el momento. Tanto less como more proveen una serie de comandos para moverse con facilidad dentro del texto paginado.

Ejemplos:

$ less /etc/passwd

$ more /etc/passwd

El man, para dar formato a su salida, utiliza por defecto el paginador less. Existen además los comando zless y zmore que permiten paginar con less y more respectivamente, a los ficheros comprimidos sin necesidad de descomprimirlos previamente.

Comandos tail y head

Los comandos tail y head muestran respectivamente el final y el comienzo (10 líneas por defecto) de uno o varios ficheros. De no especificarse al menos un fichero toman la entrada estándar.

Sintaxis:

tail [opciones] [ficheros]

head [opciones] [ficheros]

Algunas opciones:

-<n> imprime las n últimas (primeras) líneas en lugar de las diez establecidas por defecto.

Ejemplos:

$ tail /etc/passwd

$ tail –2 /etc/passwd

$ head /etc/passwd

$ head –2 /etc/passwd

Comando ln

La orden ln se utiliza para permitir que un mismo archivo aparezca en el sistema de archivos bajo dos nombres diferentes, pero con una única copia. Con ln no se hace una copia del archivo origen, solamente se crea otro nombre de archivo que hace referencia al mismo archivo físico. Esto permite que una única copia de un archivo aparezca en varios directorios con distintos nombres. De este modo se puede compartir información de forma cómoda. Si en un momento eliminamos alguno de los archivos que hacen referencia a la misma copia física, sólo eliminaremos el nombre, pero no la copia real. Ésta sólo será definitivamente suprimida si eliminamos todos sus vínculos. El número de enlaces de un archivo lo indica el segundo campo de la información que obtenemos con la orden ls –l.

Sintaxis:

ln prog1 prog2 creamos un enlace fuerte al prog1 denominado prog2. A partir de este momento prog1 y prog2 son dos archivos diferentes que contienen la misma información y una única copia en el disco.

A este tipo de enlaces se les conoce con el nombre de enlaces fuertes o **hard links**. En los enlaces fuertes ambos archivos tienen el mismo número de i-nodo, de manera que accediendo a uno u otro accedemos al mismo archivo físico. Cualquier cambio realizado en el primero de ellos se manifestará en el segundo y viceversa. El problema de este tipo de enlaces es que no sirven para archivos que se encuentren en sistemas de archivos diferentes. Los enlaces duros tampoco son aplicables a directorios.

Para solventar esto podemos hacer uso de los enlaces simbólicos o **soft links**, que tienen una funcionalidad similar a los enlaces duros y pueden utilizarse en archivos que se encuentren en diferentes sistemas de archivos, así como enlazar directorios.

Sintaxis:

ln –s prog1 prog2 hemos creado un enlace a prog1 apuntado por prog2.

Debemos tener cuidado con lo enlaces simbólicos, ya que si eliminamos el archivo que actúa como destino del enlace, el archivo que lo enlazaba seguirá existiendo y apuntará a un archivo no existente.

11.5 REDIRECCIÓN DE ENTRADA, SALIDA Y ERRORES

La redirección de entrada-salida es una de las características más relevantes y versátiles de los sistemas operativos UNIX y Linux. Vamos a tratar a continuación este tema, describiendo previamente una serie de conceptos necesarios para entender fácilmente la redirección.

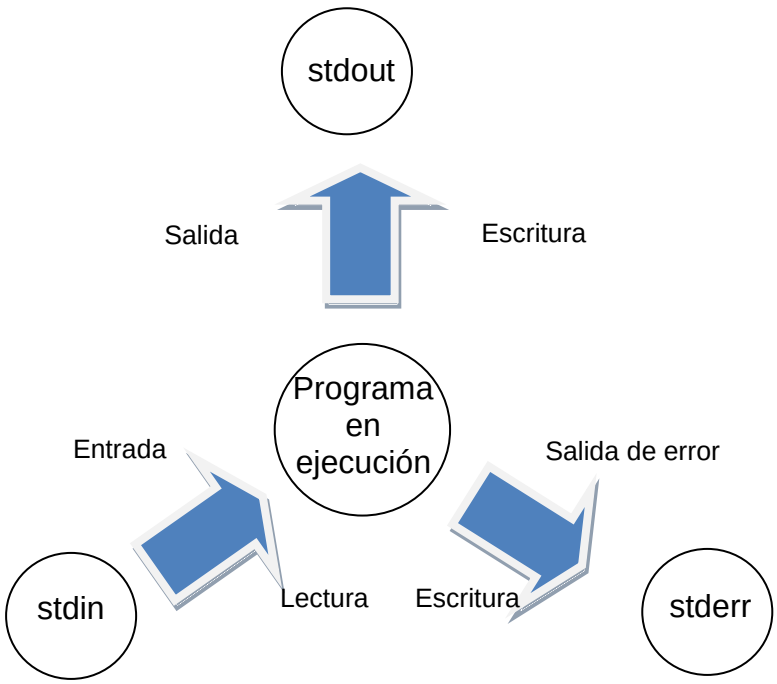
Cada vez que se inicia un intérprete de órdenes, se abren automáticamente tres archivos. Abrir un archivo implica que el núcleo o Kernel del sistema operativo habilitará las estructuras necesarias para poder trabajar con dicho archivo. Cuando se abre un archivo, el sistema operativo devuelve un número entero, denominado descriptor de archivo, que será utilizado por los programas para manipular dicho archivo (leer datos en él, escribir datos en él, etc.). Estos archivos representados en la figura que aparece a continuación y son los siguientes:

El **archivo estándar de entrada**, conocido como **stdin**, cuyo **descriptor es el 0**, y se identifica con el **teclado**. Es por dónde generalmente los programas leen su entrada.

El **archivo estándar de salida**, conocido como **stdout**, cuyo **descriptor es el 1**, y se identifica con la **pantalla**. Es a dónde los programas envían generalmente sus resultados.

El **archivo estándar de errores**, conocido como **stderr**, cuyo **descriptor es el 2**, y que se idéntica como el anterior con la pantalla. Es a dónde los programas envían sus salidas de error.

Nota: No debe extrañarnos el hecho de que tanto el teclado como la pantalla sean tratados por UNIX y Linux como archivos ordinarios, como ya hemos estudiado en el primer tema dedicado a Linux, esta es una de las características más relevantes de estos sistemas operativos, aplicable a todos los dispositivos físicos.



11.5.1 REDIRECCIÓN DE LA ENTRADA

Cualquier orden que lea su entrada de stdin puede ser avisada para que tome dicha entrada de otro archivo. Esto se hace utilizando el carácter <. La redirección de la entrada no produce ningún cambio en el archivo de entrada.

Ejemplo:

```
$ ls <prueba
```

prueba es un fichero que he creado y cuyo contenido es /home/noelia. El comando ls, como ya sabemos, lista el contenido del directorio que le indiquemos, en este caso va a recibir la entrada del fichero prueba y lo va a mostrar por la salida estándar que en este caso es la pantalla. Como lo que he grabado en el fichero prueba es /home/noelia, lo que vamos a conseguir es mostrar por pantalla el contenido de mi directorio personal.

Cuando, como en el ejemplo anterior, el intérprete de órdenes, detecta el símbolo **<** en la línea de comandos, sabe que tiene que realizar una redirección de la entrada. Como consecuencia de esto lo que hace el intérprete de comandos es cerrar el archivo de entrada estándar, cuyo descriptor es el 0, y en su lugar va a colocar el archivo prueba, al que va a asignar el descriptor 0. Es decir, aunque la orden cat siempre lea del archivo cuyo descriptor es el 0, unas veces el 0 va a corresponderse con el teclado y otras con cualquier otro archivo. Realmente es el shell o interprete de comandos el que engaña al comando ls, que no se entera de la redirección producida.

11.5.2 REDIRECCIÓN DE LA SALIDA

Se puede redireccionar la salida de cualquier orden a un determinado archivo en vez de hacerlo por la salida estándar stdout. Para realizar una redirección de salida se utiliza el símbolo **>**. Si el archivo al que redireccionamos no existe, el shell lo creará automáticamente; si, por el contrario ya existía, entonces se sobrescribirá la información, machacando el contenido original del archivo. Si por cualquier causa lo que queremos es añadir información a un archivo sin destruir su contenido, deberemos utilizar para la redirección el símbolo anterior doble, **>>**

Ejemplo:

```
$ date >prueba
```

El comando date mostraría por pantalla la fecha y hora actuales, de este modo esa información no se mostrará por pantalla, en lugar de eso será almacenada en el fichero prueba.

Ocurre algo similar a lo que ocurre con la redirección de la entrada, en este caso cuando el shell detecta el símbolo de redirección de salida (ya sea > o >>), cierra el archivo cuyo descriptor es el 1, la pantalla y le asigna ese descriptor al archivo prueba. Todo ocurre sin que el comando date se entere de nada. Es el intérprete el que se encarga de todo el proceso.

Si ahora queremos añadir más información al archivo son destruir el contenido existente podremos hacer lo siguiente:

```
$ ls –l /home/noelia >>prueba
```

Si ahora editáramos el contenido del fichero prueba, a continuación de la información que ya almacenaba, que era la fecha y hora actuales, aparece el listado en formato largo del contenido de mi directorio de trabajo.

11.5.3 REDIRECCIÓN DE ERRORES

La mayoría de las órdenes en Unix/Linux producen diagnósticos para ver si algo va mal en su ejecución. Si al ejecutar una orden esta genera un error, ese error se mostrará en la salida estándar de errores, stderr, que como antes explicamos es la pantalla. Podremos redireccionar su salida a otro archivo empleando los símbolos 2> o 2>>, dependiendo de si lo que queremos es crear o añadir datos al archivo respectivamente.

Para comprender mejor el proceso vamos a ir ilustrándolo con un ejemplo:

Vamos a ejecutar la orden cp sin argumentos:

```
$ cp
cp: faltan argumentos (ficheros)
Pruebe ‘cp – help’ para más información
$
```

Vemos cómo por pantalla se han visualizado los mensajes de error generados por cp. Es lógico el error, puesto que cp necesita como mínimo dos argumentos para poderse ejecutar correctamente. Si queremos que estos mensajes de error no salgan por pantalla, a simple vista, una forma de hacerlo podría ser la siguiente:

```
$ cp >basura
cp: faltan argumentos (ficheros)
Pruebe ‘cp – help’ para más información
$
```

sin embargo, vemos cómo los mensajes de error siguen saliendo por el terminal y no son redireccionados al archivo que hemos llamado basura. La razón es que la salida de error de cp no va dirigida a stdout (archivo con descriptor 1), sino a stderr (archivo con descriptor 2), que son archivos diferentes, aunque coincidan con el mismo dispositivo físico de salida. Para indicarle al Shell que lo que queremos es redireccionar la salida de error a otro archivo, debemos utilizar como hemos indicado anteriormente, 2>

```
$ cp 2>basura
```

Si ahora visualizamos el fichero basura, veremos que contiene el mensaje de error generado por la orden cp al no recibir ningún argumento:

```
$ cat basura
cp: faltan argumentos (ficheros)
Pruebe 'cp – help' para más información
$
```

Si lo único que deseamos es evitarnos estos mensajes de error, pero no nos interesa crear archivos basura, deberemos enviar dichos mensajes a un archivo de dispositivo denominado /dev/null. El archivo /dev/null es un pozo sin fondo donde podemos enviar toda la información no deseada sin tener que preocuparnos de borrar su contenido.

```
$ cp 2>/dev/null
```

Las operaciones realizadas por el intérprete de órdenes en la redirección de errores son completamente equivalentes a las realizadas en una redirección de salida. La única diferencia es que ahora trabajará con el descriptor 2 en lugar de hacerlo con el descriptor número 1.

11.6 CONCEPTO DE FILTRO

Cualquier comando que lea su entrada de la entrada estándar (stdin) y escriba su salida en la salida estándar (stdout) se denomina filtro. Vamos a estudiar los comandos denominados filtros, más importantes, aunque existen muchos más:

11.6.1 COMANDOS FILTRO

Comando cat

Con esta orden se puede visualizar en pantalla el contenido de un archivo. Si a cat no le pasamos como argumento ningún archivo de texto, entonces leerá caracteres de la entrada estándar (teclado) y cada vez que pulsemos enter visualizará en pantalla lo tecleado, esto se repetirá hasta que pulsemos Ctrl-d o Ctrl-z, momento en el que deja de ejecutarse el comando.

Ejemplo 1:

$cat

hola	**lo que recibe del teclado**
hola	**lo que devuelve a la pantalla**

CTRL+ D fin de cat

Si al comando cat le pasamos como argumento un fichero, mostrará por pantalla el contenido de dicho fichero:

Ejemplo 2:

$ cat /etc/passwd	**muestra por pantalla el contenido del fichero /etc/passwd**

Utilizando redireccionamiento podremos utilizar este comando para introducir información en un fichero, ya sea desde teclado o desde otro fichero.

Ejemplo 3:

$ cat >prueba	**Escribe el texto que desees y para acabar ctrl+z o d**
$ cat prueba	**Veras en pantalla el texto que mandaste al archivo prueba**

Ejemplo 4:

$ cat archivo_1 >archivo2	**Envía el contenido del archivo_1 al archivo_2 (archivo_1 sigue manteniendo su contenido)**
$ cat archivo2	**Verás en pantalla que el contenido del archivo_2 es el mismo que el del archivo_1**

Otra utilidad que posee el comando cat es la de concatenar ficheros:

Ejemplo5:

Previamente créate dos archivos, uno llámalo archivo_3 y al otro archivo_4

$ cat archivo_3 archivo_4 >archivo_5

$ cat archive_5	**Si visualizamos el contenido de archivo_5 veremos que almacena el contenido del archivo_3 y a continuación el contenido del archivo_4**

Comando sort

El comando sort ordena las líneas de un fichero mostrándolas por la salida estándar. Esta ordenación no produce ninguna modificación en los archivos tratados. De no especificarse un fichero toma la entrada estándar.

Sintaxis: **sort [opciones] [fichero]**

Algunas opciones:

-r : ordena al revés.

-f : trata las mayúsculas y minúsculas por igual, de no hacerlo así por defecto sort ordena según los caracteres ASCII.

-n: ordena según el valor numérico.

-k nº de columna: indica el número de columna por la que queremos ordenar. La primera columna es la columna 1.

-t y a continuación el carácter separador: Nos permite especificar el carácter separador entre los distintos campos que componen el fichero.

Ejemplo 1: Introduce el siguiente conjunto de líneas desde el teclado, cuando pulsemos la combinación CTRL+ D devolverá a la pantalla las mismas líneas pero de forma ordenada.

$sort

méndez

luque

rodríguez

CTRL+ D **fin de entradas**

Devolvera por pantalla:

luque

méndez

rodríguez

Ejemplo 2:

Crea un fichero llamado ordenar con el siguiente contenido:

Juan Perez

Ana Ruiz

Maria Arco

Jose Buena.

Ejecuta el comando sort tal y como se muestra a continuación y observa las salidas obtenidas:

$ sort ordenar

$ sort –r ordenar

$ sort +1 ordenar

Ejemplo 3:

Podemos ordenar unos números con sort y mandar el resultado a un fichero llamado pruebasort:

$sort >pruebasort

3

2

1

CTRL+ D

$

Mostramos el fichero pruebasort y vemos como sort ordenó los números:

$cat pruebasort

1

2

3

$

Si queremos ordenar más números y guardarlos en el mismo fichero:

$sort >>pruebasort

6

5

2

CTRL+ D

$

El resultado lo vemos con cat

$cat pruebasort

1

2

3

2

5

6

$

Ejemplo 4: Creamos el fichero numeros con el siguiente contenido

$cat >numeros

101

112

10

3113

64

19

1111

CTRL+ D

$

Vamos a ordenar el contenido del fichero numeros mostrando en pantalla la ordenación:

$sort numeros

10

101

1111

112

19

3113

64

$

Como puede apreciarse la ordenación no es correcta, o al menos no es la que esperábamos. Esto es así porque, por defecto, sort ordena las palabras según los caracteres ASCII que la componen. Si lo que deseamos es ordenar según el valor numérico asociado a esos caracteres, deberemos utilizar la opción –n (ordena numéricamente), tal y como se muestra a continuación:

$sort -n numeros

10

19

64

101

112

3113

1111

$

Los campos separadores utilizados por defecto son los tabuladores, y en algunas versiones de sort, también los espacios en blanco, pero también podemos decirle que utilice cualquier tipo de separador específico, utilizando para ello la opción –t y a continuación el separador. Como ejemplo vamos a ordenar el archivo que figura a continuación, denominado sortfich, por el último campo:

Ejemplo 5:

$ cat >sortfich

blanco:113:Marte:1543:Manuel

verde:111:Jupiter:1968:Sebastian

azul:24:Venus:19110:Ana

rojo:35:Neptuno:1122:Javier

amarillo:135:Tierra:1234:Raul

$

Como podemos apreciar, los distintos campos están separados por dos puntos. Eso no es ningún problema para sort, ya que podremos especificar el carácter de separación de campos que deseemos.

$sort –t: -k 5 sortfich y la salida proporcionada será

azul:24:Venus:19110:Ana

rojo:35:Neptuno:1122:Javier

blanco:113:Marte:1543:Manuel

amarillo:135:Tierra:1234:Raul

verde:111:Jupiter:1968:Sebastian

$

Comando wc

El comando wc imprime el número de líneas, palabras y bytes de uno o varios ficheros. Si son varios ficheros hace también un resumen de los totales. De no especificarse un fichero toma la entrada estándar.

Sintaxis:

wc [opciones] [ficheros]

Algunas opciones:

-l sólo cuenta líneas.

-c sólo cuenta bytes.

-w sólo cuenta palabras.

Ejemplos:

$ wc ordenar

$ wc –l ordenar

$ wc –w ordenar

$ wc –c ordenar

Comando cut

El filtro cut se usa para cortar y pasar a la salida estándar las columnas o campos de la entrada estándar o del archivo especificado mediante redirección.

El comando cut nos permite cortar una línea de texto, para obtener un subconjunto en lugar de la línea completa. Podemos cortar por número de caracteres, por campos, etc.

Sintaxis:

cut [opciones] [ficheros]

Algunas opciones:

-c N-M corta desde el carácter número N hasta el carácter número M.

-c N- corta desde el carácter número N hasta el final

-c –N corta desde el principio hasta el carácter número N

-c N,M corta el carácter número N y el carácter número M

-d”:” –f1 separa la línea en campos divididos por el carácter : y nos muestra sólo el primer campo

-d”-“ –f3 separa la línea en campos divididos por el carácter – y nos muestra sólo el 3 campo.

Ejemplos:

$ cut –c 1-3 ordenar	**corta los desde el carácter 1 al 3 del fichero ordenar**
$ cut –c 1,3 ordenar	**corta el carácter 1 y el 3 del fichero ordenar**
$ cut –d” “ –f2 ordenar	**corta la segunda columna del fichero ordenar**
$ cut –d”:” –f1 /etc/passwd	**corta la primera columna del fichero /etc/passwd**

Comando grep

El comando grep es un filtro que nos permite buscar patrones en ficheros. Por defecto devuelve todas las líneas que contienen un patrón (cadena de texto) determinado en uno o varios ficheros. Utilizando las opciones se puede variar mucho este comportamiento. Si no se le pasa ningún fichero como argumento hace la búsqueda en su entrada estándar.

Sintaxis:

grep [opciones] <patrón> [ficheros]

Algunas opciones:

-c devuelve sólo la cantidad de líneas que contienen al patrón.

-i ignora las diferencias entre mayúsculas y minúsculas.

-H imprime además de las líneas, el nombre del fichero donde se encontró el patrón. Es así por defecto cuando se hace la búsqueda en más de un fichero.

-l cuando son múltiples ficheros sólo muestra los nombres de aquellos donde se encontró al patrón y no las líneas correspondientes.

-v devuelve las líneas que no contienen el patrón.

-r busca en un directorio de forma recursiva.

-n imprime el número de cada línea que contiene al patrón.

Ejemplos:

$ grep home /etc/passwd	**Muestra las líneas del archivo passwd donde aparece la palabra home**
# grep -n home /etc/passwd	**Además de la línea devuelve el nº de línea**
$ grep -c root /etc/group	**Devuelve el nº de líneas que contienen la palabra root en el fichero group**
$ grep -lr root /etc/security/	**Devuelve los archivos que contienen la palabra root**

Comando tr

El comando tr se emplea como traductor. Como todo filtro, tr lee datos en la entrada estándar, los procesa y deposita los resultados en la salida estándar. El empleo más evidente de tr es como conversor de letras mayúsculas a minúsculas, y viceversa.

Sintaxis:

tr [-dsc] cadena1 cadena2

Supongamos que tenemos un archivo denominado fich con el siguiente contenido:

$cat fich

Este es un archivo de texto

QUE CONTIENE LETRAS MAYÚSCULAS Y minúsculas

$

A este archivo vamos a aplicarle la orden tr con diversas opciones:

Ejemplo 1:

Vamos a convertir todos los caracteres del rango de la **A** a la **Z** en sus correspondientes del rango de la **a** a la **z**.

$ tr [A-Z] [a-z] <fich

este es un archivo de texto

que contiene letras mayúsculas y minúsculas

$

Ejemplo 2:

Vamos a realizar ahora el proceso inverso al realizado en el ejemplo 1, convertir de minúsculas a mayúsculas. Para ello, emplearemos la orden siguiente:

$ tr [a-a] [A-Z] <fich

ESTE ES UN ARCHIVO DE TEXTO

QUE CONTIENE LETRAS MAYÚSCULAS Y MINÚSCULAS

$

Ejemplo 3:

También podemos sustituir un rango de caracteres por un carácter cualquiera de la forma siguiente:

$ tr [A-Z] x <fich

este es un archivo de texto

xxx xxxxxxxx xxxxxx xxxxxxxxxx x minúsculas

$

Como puede apreciarse por la salida proporcionada, hemos convertido el rango de caracteres de la **A** a la **Z** por el carácter **x**.

Ejemplo 4:

El comando tr también puede ser empleado para eliminar determinados caracteres de un archivo. Para ello debemos emplear la opción –d y a continuación indicarle el carácter o caracteres que deseamos eliminar.

$ tr –d [A-Z] <fich

ste es un archivo de texto

minúsculas

$

Ejemplo 5:

En el caso anterior eliminamos cualquier carácter del archivo fich que esté comprendido en el rango A-Z. Ahora vamos a hacer lo mismo, pero eliminando las minúsculas:

$ tr –d [a-z] <fich

E

QUE CONTIENE LETRAS MAYUSCULAS Y

$

Ejemplo 6:

Otra de las opciones de tr es la posibilidad de eliminar caracteres repetidos en el texto. Para ello, debemos emplear la opción –s. Supongamos que tenemos un fichero denominado fich2 con el siguiente contenido:

$ cat fich2

Aqquuiiiiiii tteeeeennnnnngggooooo rrreeeepppppeeeettttttiiiiddooossssssss

scciiiiieeeeerrrrtoooooossss ccaaaaaarraaccccctteeerrreeeesssssss

$

Para eliminar caracteres repetidos, haremos lo siguiente:

$ tr –s [a-z] <fich2

Aqui tengo repetidos

ciertos caracteres

$

Ejemplo 11:

Por último, la opción –c se empleará para indicar el complemento de un patrón de caracteres:

$ tr –c [A-Z] “ “ <fich

E

QUE CONTIENE LETRAS MAYUSCULAS Y

$

Como puede observarse en este ejemplo hemos sustituido todo carácter que no pertenezca al patrón [A-Z] por un espacio en blanco.

11.6.2 COMANDO FIND

Comando find

El comando find es uno de los más poderosos en un sistema UNIX/Linux, pero también uno de los que tienen una sintaxis más compleja. Se ha ubicado a find en un punto aparte porque no es un filtro.

Permite buscar de forma recursiva en un directorio a todos los ficheros que cumplan ciertas condiciones. Las condiciones pueden estar relacionadas con el nombre de los ficheros, el tamaño, los permisos, el tipo, las fechas de acceso y modificación, etc.

Una vez localizados, podemos hacer que ejecute distintas acciones sobre ellos.

Sintaxis:

find [camino] [opciones]

Algunas opciones:

-name <expresión> permite especificar patrones para los nombres de los ficheros a buscar.

-user <expresión> con esta opción, find seleccionará los archivos que pertenezcan al usuario especificado a continuación.

-group <expresión> con esta opción, find seleccionará los archivos que pertenezcan al grupo especificado a continuación.

-iname <expresión> permite especificar patrones para los nombres de los ficheros a buscar sin tener en cuenta mayúsculas y minúsculas.

-type <tipo> permite indicar el tipo de fichero a buscar. Este puede ser d para directorios, f para ficheros regulares, l para enlaces simbólicos, b para archivo especial de dispositivos tipo bloque, c para archivo especial de dispositivos tipo carácter, p para tuberías, etc.

-size +/-<n> permite indicar el tamaño máximo y/o mínimo de los ficheros a buscar. Por defecto el tamaño se expresa en bloques de 512 bytes, pero si se precede este

por un carácter c se referirá a bytes, k a kilobytes, w a palabras de dos bytes y b a bloques.

-perm [+|-]<modo> permite referirse a aquellos ficheros cuyos permisos sean exactamente modo, incluya todos los de modo (signo -) o incluya alguno de los de <modo> (signo +). El valor de <modo> se expresa en forma numérica.

Todos los operadores anteriores pueden ser negados con el carácter ! seguido de un espacio en blanco. En este caso, find buscará todos los archivos que no cumplan la especificación indicada.

Se pueden especificar simultáneamente varias opciones, en cuyo caso se seleccionarán los archivos que cumplan todas ellas (operación and). Si dichas opciones las conectamos con el operador -o (operación or), find seleccionará los archivos que cumplan cualquiera de ellas.

-exec <comando> ; podemos indicar a find que ejecute una orden determinada y la aplique a los archivos que encuentre. Para construir la orden que queremos ejecutar con cada archivo que encuentre find contamos con la expresión {} que se sustituye por el nombre del archivo encontrado. Debemos además concluir la orden con el carácter ;. Hay que tener en cuenta que muchos intérpretes de órdenes (bash por ejemplo) consideran a ; como un carácter especial, por lo tanto será necesario colocar el símbolo \ delante del ; para evitar dicha interpretación.

Ejemplos:

$ find /etc -name '*.conf'	**Busca en /etc todos los ficheros con extensión conf**
$ find / -size +10240k -size -20480k	**Busca los ficheros cuyo tamaño esté entre 10240k y 20480k**
$ find -type d	**Busca desde el directorio actual todos los directorios**
$ find /home/noelia –size +1500 –mtime -5	**Visualizamos todos los archivos que cuelgan de mi directorio de trabajo cuyo tamaño sea mayor de 1500 bloques y que hayan sido modificados en los últimos 5 días**
$ find $HOME –type f –o ! –user $LOGNAME	**Visualizamos todos los archivos que cuelgan de nuestro directorio de trabajo, que sean tipo archivo o que no nos pertenezcan**

$ find /home/noelia –mtime -2 –name *.tmp –exexc rm {} \;

En este último ejemplo borramos de nuestro directorio home todos los archivos que hayan sido modificados en los últimos dos días y cuyo nombre termine en .tmp

11.6 TUBERÍAS (*pipelines)*

Hay ocasiones en las que puede resultar conveniente que la salida de una orden actúe como entrada para otra. La forma de realizar esta conexión en UNIX/Linux consiste en utilizar tuberías o pipelines. A modo de ejemplo, supongamos que queremos saber el número de personas que están conectadas al sistema en un instante determinado. Una forma muy sencilla de hacerlo sería la siguiente:

$ who >archivo_temporal

$ wc –l <archivo_temporal

11

$rm archivo_temporal

Como veremos en el último punto del tema, el comando who nos presenta una línea por cada usuario conectado al sistema (realmente puede que en una terminal en un momento dado está conectado un usuario determinado que resulta que no fue el que arrancó esa terminal, realmente who, hace referencia al usuario que inicio la terminal)

En la primera línea de comando enviamos la salida de who a un fichero que hemos denominado archivo_temporal.

La siguiente línea de comando utiliza el comando wc con el modificado –l, la entrada la recibe mediante redireccionamiento del fichero archivo_temporal, y el comando contará el número de líneas que componen dicho fichero, mostrando la salida en la salida estándar, la pantalla.

Por último borramos el archivo archivo_temporal, para dejar las cosas como estaban inicialmente.

El proceso seguido es relativamente simple, aunque si algo engorroso. Las tuberías son una forma de evitar esta pérdida de tiempo, puesto que permite que la salida de una orden sirva como entrada de la otra. Una forma de resolver lo anterior con tuberías sería la siguiente:

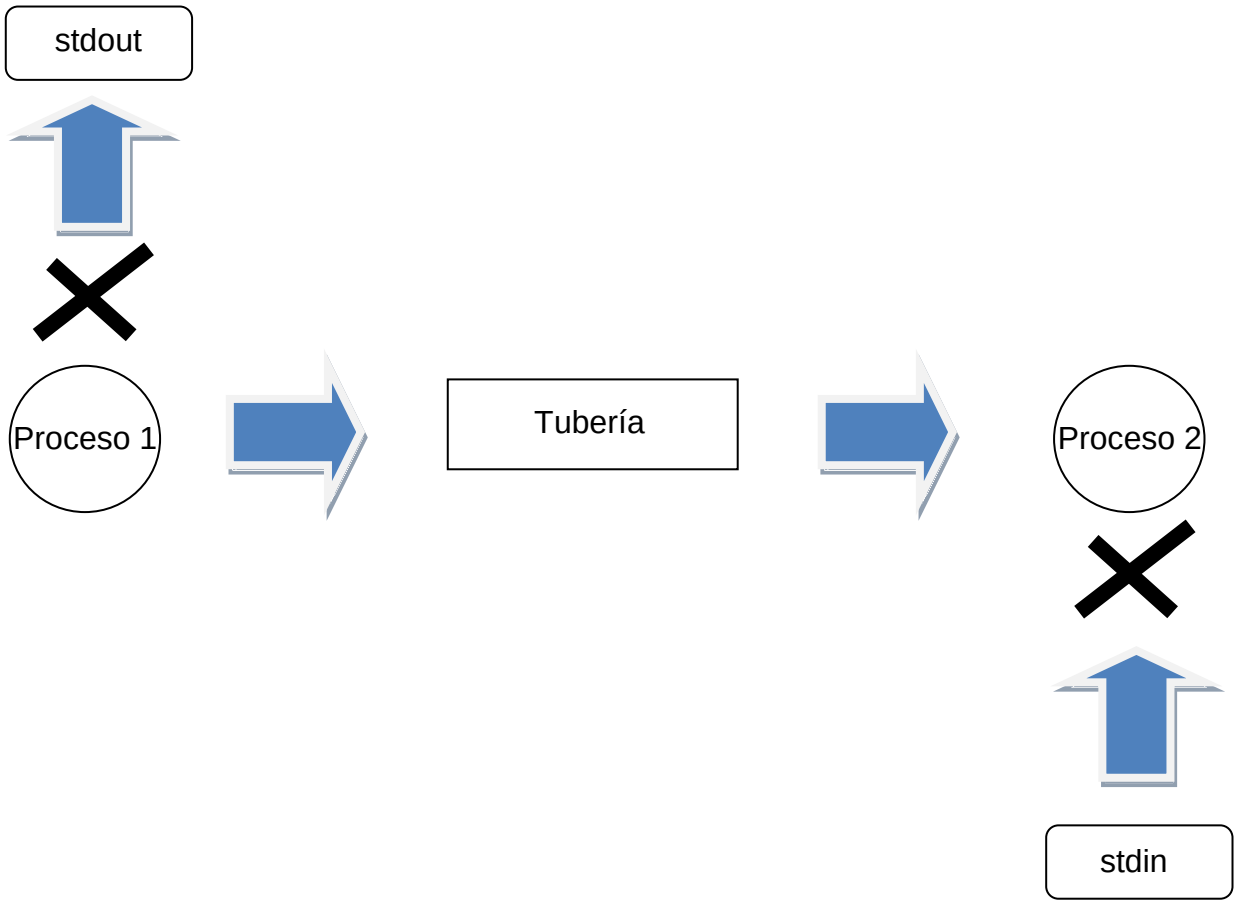
$ who | wc –l

11

$

Como acabamos de ver las tuberías pueden ser utilizadas para combinar comandos, de forma que la salida del primero es enviada a la entrada del segundo y así sucesivamente.

Para crear las tuberías utilizamos el símbolo | (barra vertical, carácter de canalización, AltGr+1). Cuando avisamos el carácter | estamos avisando al shell de que internamente cree un mecanismo que permita la comunicación entre las dos órdenes, situadas a los lados del carácter tubería. El shell redirige la salida de la primera orden a la entrada de la segunda orden, la figura que aparece a continuación refleja la situación descrita:



Ejemplos:

# ls –l /home	Muestra por pantalla el contenido del directorio home
# ls –l /home |sort	Muestra por pantalla el contenido del directorio home, pero ordenado
# ls –l /home |sort >fichero1	Realiza lo mismo que la anterior solo que no muestra la salida por pantalla, la redirecciona al fichero fichero1
$ls /usr/bin |more	Muestra el contenido de /usr/bin por paginado

$ls |sort -r |head -1	Muestra por pantalla la primera fila del listado del directorio actual ordenado alfabéticamente de mayor a menor. **(Como vemos en este ejemplo se pueden emplear más de una tubería en una misma línea de comandos)**
$cat fichero | wc –l	Con el comando cat visualizamos el contenido del fichero, pero esta salida, en vez de ir a la pantalla, se mete en la tubería que va hacia la entrada de la orden wc que con su opción –l nos muestra en pantalla el número de líneas que ha leído.

11.7 COMANDOS PARA LA GESTIÓN DE DISPOSITIVOS

Como comentamos anteriormente, a diferencia de Windows, Linux no asocia una letra a cada uno de los sistemas de archivos. Linux tiene una estructura de árbol de archivos única, y cada sistema de archivos está montado en algún lugar de la estructura del árbol. Es decir, Linux monta la raíz de su sistema en la partición que contiene al sistema de archivos raíz, y a partir de ahí puede montar otros sistemas de archivos en la estructura de árbol sobre diferentes puntos de montaje.

Debemos tener en cuenta que:

• En principio, solamente el administrador del sistema root podrá montar un sistema de archivos.

• El directorio que actúa como punto de montaje debe existir.

• Es preferible que el punto de montaje esté vacío, puesto que el sistema ocultará todos los ficheros, excepto los del sistema montado, hasta que desmontemos.

Los ficheros **/etc/fstab** y **/etc/mtab** contienen información sobre los dispositivos montados. Estos ficheros se estudiaran con detalle en el último punto del tema 8.

Los controladores correspondientes a los dispositivos del sistema se encuentran en el directorio ***/dev***. Por lo tanto, para acceder a las distintas unidades de disco, montaremos el dispositivo correspondiente del directorio */dev* e indicaremos el punto de montaje. Además, le diremos al sistema qué tipo de sistema de archivos queremos montar.

Comando mount (esta acción se conoce como ``montar", en definitiva es asociar el dispositivo a un directorio determinado)

Su sintaxis básica es la siguiente:

mount –t <tipo> <dispositivo> <punto de montaje>

Donde:

-t tipo: indica el tipo de sistema de archivos que usa la unidad de disco para guardar los ficheros. Los tipos más usuales son:

• iso 9660 para CD ROM.

• vfat para disquete y FAT.

• msdos para particiones de MS DOS.

• ext2 para particiones de Linux.

***<dispositivo>*:** indica el dispositivo que se va a montar, y que estará representado por un archivo en directorio /dev.

***<punto de montaje>*:** es el directorio en el que se pondrá a disposición del usuario el contenido de la unidad montada.

Por regla general, se utiliza el directorio */mnt*, que contiene una serie de subdirectorios relativos a los diferentes sistemas de archivos de que dispone el sistema. De todas formas, el usuario siempre puede crear un directorio vacío con el nombre que él elija para montar las unidades de disco que desee donde desee. Otro directorio que también se usa es el /media.

Para montar el disquete o el CD ROM deberán estar insertados en la unidad los soportes correspondientes.

Ejemplos:

Montaje del disquete

#mount –t vfat /dev/fd0 /mnt/floppy

Para ver el contenido del disquete accedemos al directorio */mnt/floppy*.

#mount /dev/fd0 /floppy

Montaje del CD ROM

#mount –t iso9660 /dev/cdrom /mnt/cdrom

Para ver el contenido del CD ROM accedemos al directorio /mnt/cdrom

#mount /media/cdrom0

Montaje de la primera partición de Windows

#mount –t vfat /dev/sda1 /mnt/win_c

#mount –t ntfs /dev/sda1 /media/win_c

Cuando el usuario haya dejado de usar la unidad, deberá desmontarla. Para ello utilizamos la orden:

Comando umount.

En el caso del CD ROM y el disquete, habrá que desmontar las unidades antes de sacarlos. La sintaxis de umount es:

umount <punto de montaje>|<dispositivo>

Ejemplos:

#umount /mnt/floppy (o /dev/fd0)

#umount /mnt/cdrom (o /dev/cdrom)

#umount /mnt/win_c (o /dev/sda1)

11.8 COMANDOS PARA DESCONECTARSE DEL SISTEMA

Comando exit

El comando exit permite terminar el shell actual. Si se tiene un único shell es equivalente a desconectarse del sistema, pero si se está en un subshell sólo se terminará este, retornando al shell anterior.

Comando logout

El comando logout permite desconectarse del sistema a partir de un login shell (primer shell que se ejecuta al establecer la conexión).

La secuencia de caracteres Ctrl-d permite terminar el shell actual. Si es un login shell equivaldrá a hacer logout y si no, a hacer exit.

Comando halt, reboot, poweroff

Estos comandos nos permiten suspender, reiniciar o apagar el sistema.

Comando shutdown

Este comando nos permite “echar abajo” el sistema. Algunas opciones interesantes:

-c cancela un shutdown que está en proceso de ejecución

-f Reinicia más rápido, ya que no controla la integridad de los sistemas de archivos

-h Cuando el sistema se apaga, apaga el ordenador (o lo intenta)

-r Cuando el sistema se apaga, intenta reiniciarlo

Después de estas opciones, se le indica cuando queremos apagar el sistema:

now lo apaga inmediatamente, ahora.

20:00 lo apaga a las 8 de la tarde

+10 lo apaga en 10 minutos

 # shutdown -h +4 lo apaga en 4 minutos

 # shutdown -h now lo apaga ahora mismo.

 # shutdown -r now lo reinicia ahora mismo

11.9 COMANDOS VARIOS

Comando clear

Borra el contenido de la pantalla.

Comando who

Muestra por pantalla una línea por cada usuario que en ese momento está conectado al sistema (realmente muestra a los usuarios que iniciaron cada una de las terminales). Muestra de izd a drcha el nombre del usuario el número de terminal y la fecha y hora. Una variedad de este comando es:

whoami, que muestra información solo del usuario conectado que pide información.

Comando su

Sintaxis: su [-] [usuario]

La orden su permite cambiar nuestro identificador de usuario. Cuando se invoca, no pide la palabra clave (password) del usuario al que queremos cambiar (excepto si somos el superusuario, en cuyo caso podremos logearnos como cualquier otro usuario sin necesidad de teclear su contraseña). Si a su no le pasamos como parámetro ningún nombre de usuario, asumirá que deseamos convertirnos en el administrador del sistema (también conocido como superusuario o root).

Obviamente si desconocemos la contraseña del usuario con el que pretendemos logearnos, la orden fallará.

La opción – se emplea para indicar a su que se tomen los parámetros de inicio (directorio de arranque, variables de entorno, etc.) definidos para el usuario al que nos convertimos. Por defecto estos parámetros no se toman.

Ejemplo:

$ su –lucas

Password:

Comando id

Sintaxis: id [-ug] [usuario]

La orden id devuelve el identificador (número) de usuario y de grupo del usuario que le indiquemos. Si no se le indica usuario, id visualizará los identificadores asociados al usuario que invoca la orden.

-u Visualiza sólo el UID (identificador de usuario)

-g Visualiza únicamente el GID (identificador de grupo)

Ejemplo:

$ id

Uid=508 (chan) gid=504 (igx) grupos=504 (igx)

$ id lucas

Uid 519 (lucas) gid=519 (lucas)

Comando echo

Sintaxis: echo [–n] [cadena de caracteres]

La orden echo repite todo lo que pasemos como parámetro. Esta orden como veremos en el último tema dedicado al bloque de Linux se utiliza mucho dentro de los programas Shell.

La opción –n se emplea para evitar el retorno de carro

Ejemplo:

$ echo Esta orden repite todo

Esta orden repite todo

$

Comando alias

El comando alias permite asignarle otros nombres a los comandos. De esta forma se pueden abreviar o llamarlos de forma más nemotécnica.

Sintaxis: **alias [nombre[=valor]...]**

Sin argumentos muestra todos los alias definidos por el usuario actual. Para deshabilitar un alias se emplea el comando unalias.

Ejemplo:

$ alias l='ls -l --color'

Comando history

El comando history nos permite controlar el histórico de comandos que se van almacenando en el shell. El histórico se almacena por defecto en el fichero ~/.history lo que nos indica que existe un histórico diferenciado para cada usuario. Si ejecutamos el comando history nos devolverá la relación de comandos para ese usuario.

 # history

Una posibilidad es enviar la salida de la orden history a un fichero, con lo que conseguimos tener una relación de los comandos que hemos ejecutado.

Comando uname

El comando uname nos da información sobre el sistema, como el nombre del kernel, su versión, el nombre de la maquina, etc.

 # uname –a

Comando date

El comando date sirve para consultar y cambiar la fecha y hora del sistema.

 # date

El comando date tiene bastantes opciones, y con él se pueden hacer bastantes cosas. Es recomendable echarle un vistazo a sus páginas de manual para entenderlos bien. Veamos algunos ejemplos.

$ date --date=”2 days ago”	**# nos da la fecha de hace dos días**
$ date --date=”3 months 1 day”	**# nos da la fecha que será en 3 meses y un día.**
$ date --date=”25 Dec”	**# nos da el dia que será el 25 de Diciembre del año actual**
$ date --set=”2006-6-20 11:59 AM”	**# ajusta la fecha del sistema**
$ date --set=”+3 minutes”	**# adelante la hora del sistema en 3 minutos.**

Comando cal

Sintaxis: cal [mes] [año]

Muestra el calendario de la fecha indicada.