INTRODUCCION

El sistema operativo UNIX se inició en una DEC PDP−7 deshechada, en los Laboratorios Bell durante 1969. En 1973, Ritchie y Thompson reescribieron el kernel (núcleo) de UNIX en C, rompiendo así con la tradición de que el software de sistemas está escrito en lenguaje ensamblador.

Hacia 1974 fue introducido en las universidades "con fines educacionales" y al cabo de pocos años estaba ya disponible para uso comercial.

El UNIX es un sistema portable (se ejecutan en una extensa variedad de computadoras), flexible, otente, con entorno progamable, multiusuario y multitarea.

1. LOGGING IN

El administrador del sistema es una persona designada para mantener el sistema, leer el nuevo software, establecer nuevas cuentas y otras tareas.

El administrador del sistema tiene que asignar un login ID (nombre de usuario) y un password a los usuarios para que puedan empezar a trabajar.

1. 1. LOGIN ID

El login ID es el nombre que el administrador del sistema asigna a cada usuario.

1. 2. PASSWORD

En sistemas con varios usuarios, el password permite prevenir que alguien utilice mi login, pués es secreto.

El fichero donde están almacenados los passwords de los usuarios es /etc/password.

Para que un password sea válido debe contener como mínimo 6 caracteres, y el primero debe ser alfabético; pero únicamente los 8 primeros son significativos.

El password puede contener letras en mayúsculas o minúsculas, caracteres numéricos y caracteres especiales.

Para crear un nuevo password o cambiar uno ya existente se utiliza el comando passwd, cuya sintaxis es:

passwd

En la siguiente figura se ilustra cómo se cambia un password.

---

1 /mnt/carmen % passwd

Changing password for carmen on emducms1

Old password:

New password:

Re−enter new password:

Rebuilding passwd and pwrestrict databases...

2 /mnt/carmen %

---

2. PROMPT DEL SISTEMA

Una vez que se ha introducido el login y el password aparecerá en la parte izquierda de la pantalla el signo %, que es el prompt de comando. Indica que el ordenador está preparado para aceptar los comandos.

El sistema tiene por defecto dos promts, que son:

% para Berkeley C Shell (Utilizado en el SIS)

$ para Bourne Shell

El *prompt* puede cambiarse, pero siempre aparece a la izquierda del cursor.

Una vez que se ha ejecutado un comando vuelve a aparecer el prompt, esto significa que el sistema está preparado para leer otro comando.

3. LOGGING OUT

Cuando se ha terminado de utilizar el ordenador, nos podemos desconectar de dos formas:

tecleando logout

o pulsando las teclas CTRL+D (quitado en el SIS)

Si sale el mensaje: "There are stopped jobs" , para poder abandonar el

proceso se utiliza la instrucción jobs y luego se pulsa CTRL+D

4. CORRECCIONES

CTRL+H o la tecla de backspace : borra el último carácter.

CTRL+W : borra la última palabra.

CTRL+U : borra la última línea.

5. CARACTERES DE CONTROL DE TRABAJOS

CTRL+S : para la pantalla.

CTRL+Q : para continuar.

CTRL+O : descarga una salida del terminal.

CTRL+C : para "matar" un proceso (siempre que el programa tenga un control

de terminación).

CTRL+Z: para enviar a "dormir" (o detener) un proceso; luego se puede

"revivir".

Para configurar el teclado se utiliza el comando stty en el fichero .login

6. ESTRUCTURA GENERAL DE LOS COMANDOS

La sintaxis básica de todos los comandos es:

COMANDO [OPCIONES] ARGUMENTOS

donde:

COMANDO : es el nombre del comando.

[OPCIONES] : son caracteres opcionales propios de cada comando.

ARGUMENTOS : son los objetos con los que van a trabajar los comandos

(normalmente son ficheros).

Se pueden poner varios comandos en una misma línea separándolos con un ; .

La longitud de la línea del terminal está limitada, por lo que para continuar escribiendo un comando en la línea siguiente se pone un backslash (\).

Se pueden ejecutar los comandos como un "todo" poniéndolos entre paréntesis.

7. COMANDOS BASICOS DE INFORMACION

7. 1. COMANDO date

Este comando se utiliza para obtener la fecha y la hora. Su formato es :

date

7. 2. COMANDO cal

Se utiliza para obtener el calendario de cualquier mes y de cualquier año.

Su formato es:

cal [mes] año

MES, Los meses se representan por sus valores numéricos (1 − 12).

AÑO, El año es requerido aunque se especifique el mes, pués si ponemos un único número en la línea de

comando, cal considera que se refiere al año y no al mes.

7. 3. COMANDO who

Se utiliza para obtener una lista de los usuarios que están conectados al sistema; en este caso el formato del comando es:

who

La salida del comando who incluye el nombre del usuario, una etiqueta de la terminal que está utilizando y la fecha y hora en que se ha conectado.

Veamos 2 extensiones de este comando:

1. whoami Informa del usuario efectivo que se es.

2. who am i Informa del usuario real que se es.

FICHEROS

Un fichero es una colección de información que se almacena en un disco o

cinta magnética.

Los ficheros del sistema son estructuras que los ordenadores utilizan para

organizar y almacenar información. Existen 2 tipos de ficheros del sistema:

Ficheros ordinarios : estos ficheros contienen datos, textos y programas

ejecutables (comandos).

Ficheros directorios : estos ficheros contienen nombres de ficheros. Los

directorios no se utilizan para almacenar datos, si no que se utilizan para

organizar otros ficheros en grupos.

1. NOMBRES DE FICHEROS

Todos los ficheros tienen asociado un "nombre de fichero" ; este nombre

identifica el fichero y su contenido.

El nombre de un fichero puede tener de 1 a 255 caracteres; pero se pueden

utilizar únicamente los siguientes caracteres:

Letras mayúsculas (A − Z).

Letras minúsculas (a − z).

Números (0 − 9).

Subrayado (_).

Punto (.).

Coma (,).

2. EXTENSION DE UN FICHERO

Las extensiones de ficheros proporcionan una ayuda para clarificar el contenido del fichero. Van precedidas por un punto (.) en el nombre del fichero.

El UNIX utiliza algunas extensiones para realizar ciertas operaciones; por ejemplo, para compilar un programa FORTRAN, el nombre del fichero que contiene las sentencias Fortran, debe tener una determinada extensión (.f).

Pero en la mayoría de los casos las extensiones de los ficheros son opcionales.

3. METACARACTERES

Son símbolos especiales que realizan alguna operación en algunos comandos.

Estos metacaracteres son:

Asteriscos (*) : Se sustituye por cualquier secuencia de caracteres.

Interrogación (?) : Se sustituye por cualquier carácter.

Corchetes ([]) :Se utilizan para especificar una lista de caracteres o un rango. Cuando se utilizan para especificar un rango hay que poner el signo − separando el primer carácter y el último del rango.

En la siguiente figura aparecen ejemplos de cómo se utilizan los metacaracteres.

_____

% ls t*

test1 test1.dato test2 test3

```
% ls test?
test1 test2 test3
% ls *[13]
states1 test1 test3
% ls [o−z]*
%
```

---

4. FICHEROS INVISIBLES

Los ficheros cuyos nombres comienzan con un punto (.) se llaman "ficheros invisibles" porque normalmente no aparecen cuando se pide el listado de un directorio.

Normalmente los ficheros invisibles se utilizan para almacenar información que el sistema utiliza automáticamente.

5. ESTRUCTURA JERARQUICA DE FICHEROS

La estructura del conjunto de todos los ficheros del sistema, es una estructura de árbol invertido.Un directorio equivale a abrir una rama dentro del árbol.

Los directorios pueden contener otros directorios, ficheros ordinarios o estar vacíos.

Un fichero ordinario es siempre el último fichero en un path (camino).

El primer directorio de la estructura es el directorio raíz; todos los demás ficheros y directorios parten de él. El directorio raíz se designa con un nombre especial, /. Ningún otro fichero puede tener este nombre.

En el sistema UNIX, todos los ficheros forman parte de la jerarquía.

Cualquier fichero de esta estructura es parte de una red de directorios conectados. Esta red de directorios, junto con el nombre de un fichero

particular, constituye el pathname para un fichero.

Cada fichero se identifica con un único pathname, que describe su localización con respecto a los otros directorios.

Se puede especificar un nombre de fichero utilizando pathnames absolutos o relativos:

Un pathname absoluto especifica la localización de un fichero desde el directorio raíz. Por lo tanto, todos los pathnames absolutos deben de empezar con un slash (/).

Un pathname relativo especifica la localización de un fichero con respecto al directorio en que se está trabajando, en lugar del directorio raíz, por lo que no empienzan con un slash (/).

El punto (.) se refiere al directorio en que se está, y dos puntos (..) se refiere al directorio anterior.

6. HOME DIRECTORY

El home directory es un subdirectorio del directorio raíz (root) en el que se entra cada vez que se hace login; es donde van a residir los ficheros del usuario. Normalmente tiene el mismo nombre que el nombre de usuario.

7. COMANDO pwd

El comando pwd (print working directory) nos dice en qué directorio estamos. El formato es simplemente :

pwd

8. COMANDO ls

Lista el contenido del directorio en que nos encontramos. Si ponemos sólo ls se obtiene una lista con el nombre de los ficheros; si se quiere obtener más información sobre esos ficheros se utilizan las opciones del comando, cuya sintaxis general es:

ls [−alsF] fichero

−a Lista además los ficheros invisibles (es decir, los que empiezan por

punto)

−l Dá la siguiente información de los ficheros :

TIPO DE FICHERO :

d directorio

− fichero ordinario

TIPO DE PERMISOS :

r lectura

w escritura

x ejecución

NUMERO DE ENLACES

NOMBRE DEL PROPIETARIO DEL FICHERO

TAMAÑO DEL FICHERO (en bytes)

FECHA DE LA ULTIMA MODIFICACION

NOMBRE DEL FICHERO

NOTA.− ls −l es equivalente a ll

−s El tamaño del fichero en kilobytes (1024 bytes) precede al nombre de cada

fichero.

−F Añade un / a los ficheros directorios y un * a los ficheros ordinarios

ejecutables.

NOTA.− ls −F es equivalente a lf

fichero Nombre de un fichero o un directorio.

Estas opciones se pueden combinar para obtener la información que queramos

al mismo tiempo; por ejemplo, ls −sF, dará la lista de los ficheros en la

que el nombre de cada fichero va precedido por su tamaño (en kilobytes) y va

seguido de un slash (/) en el caso de que sea un directorio o de un asterisco (*) en el caso de que sea un fichero ejecutable.

Por último, indicaremos que se puede listar el contenido de un directorio diferente al que estamos, sin más que especificar el path correspondiente a continuación de las opciones; por ejemplo:

ls −s /usr/Johnson/documentación

PROTECCION DE FICHEROS

El sistema UNIX posee un medio sencillo para controlar quién puede acceder o no a sus ficheros.

Existen tres clases diferentes de usuarios de un fichero y tres modos diferentes de acceso al fichero. Estas tres clases de usuarios figuran en la siguiente tabla:

CLASES DE USUARIO

PROPIETARIO

Usuario que ha creado el fichero. El propietario tiene capacidad de controlar quien puede acceder al fichero.

GRUPO Grupo de usuarios, normalmente relacionados por un departamento o función. Un usuario de este tipo puede acceder al fichero, pero no puede cambiar quien puede acceder al fichero.

OTROS Cualquier otro usuario del sistema. Estos usuarios pueden únicamente acceder al fichero si tienen permiso para ello.

Para cada una de las tres clases de usuarios existen 3 modos de acceso diferentes.

LECTURA (r)

Permite examinar el contenido del fichero.<7td>

Permite listar los ficheros contenidos en el directorio.<7td>

ESCRITURA (w)

Permite cambiar el contenido del fichero.

Permite crear y borrar ficheros.

EJECUCION (x)

Permite ejecutar el fichero como un comando.

Permite buscar en el directorio.

1. ESPECIFICACION DE LOS PERMISOS

Para especificar quién puede tener acceso a los ficheros y qué permisos tiene, es necesario designar leer (r), escribir (w) y ejecutar (x) para cada uno de los tres grupos de usuarios: propietario, grupo y otros.

Estos permisos tienen que ponerse en el mismo orden que aparecen cuando se utiliza el comando ls −l ó ll ; es decir:

rwx rwx rwx

| | |

| | |_____ permisos para otros usuarios

| |

| |_____ permisos para el grupo de usuarios

|

|_____ permisos para el usuario propietario

Siempre que esté presente una letra (r, w ó x), quiere decir que el usuario correspondiente tiene el permiso especificado; pero cuando en su lugar aparece un guión (_), el usuario no tiene el permiso correspondiente.

2. CAMBIO DE PERMISOS : COMANDO chmod

El comando chmod (change mode) se utiliza para cambiar los permisos de un fichero ordinario y de un directorio.

Existen dos formas de cambiar los permisos. Se pueden cambiar teniendo en

cuenta los permisos existentes (modo simbólico), o se pueden asignar permisos independientemente de los ya existentes (modo absoluto).

MODO SIMBOLICO

Cuando se utiliza el modo simbólico se pueden añadir o quitar permisos a los ficheros y directorios. El formato del comando chmod simbólico es:

chmod [who] código−operador permisos fichero

[who] Tipo de usuario. Puede tener los siguientes valores:

u : propietario del fichero

g : grupo del que el propietario es miembro

o : usuarios clasificados como otros

a : todos los usuarios del sistema (propietario, grupo y otros)

código−operador Indica la operación que se va a realizar:

+ : añadir permis

− : quitar permisos

permisos Tipo de permiso:

r : permiso de lectura

w : permiso de escritura

x : permiso de ejecución

fichero Nombre de fichero o directorio.

Por ejemplo, supongamos que el fichero mary tiene los siguientes permisos:

rwx r_ _ r_ _ y supongamos que queremos dar al grupo de usuarios y al resto de los usuarios del sistema, el permiso de ejecución; entonces pondríamos:

chmod go +x datos

MODO ABSOLUTO

El modo absoluto se especifica con 3 dígitos numéricos; cada número representa los permisos de cada tipo de usuario. Estos dígitos se obtienen,

para cada clase de usuario, a partir de los valores siguientes:

4 : permiso de lectura

2 : permiso de escritura

1 : permiso de ejecución.

Así tendremos:

0 : ningún permiso

1 : permiso de ejecución

2 : permiso de escritura

3 : permiso de ejecución y escritura (1+2)

4 : permiso de lectura

5 : permiso de lectura y ejecución (4+1)

6 : permiso de lectura y escritura (4+2)

7 : permiso de lectura, escritura y ejecución (4+2+1)

La sintaxis para el comando chmod absoluto es:

chmod modo fichero

modo Son 3 dígitos numéricos. Cada uno de ellos corresponde a los permisos

de cada tipo de usuario.

fichero Nombre de fichero o directorio.

Por ejemplo:

chmod 777 datos

concede permisos de lectura, escritura y ejecución sobre el fichero datos, a

todos los usuarios.

MANEJO DE FICHEROS

Un fichero es una colección de información almacenada en un disco o en cinta

magnética. Esta información se puede recuperar, modificar, añadir a la

almacenada en un fichero ya existente, imprimir, comparar, ..., o sólo

ejecutar.

1. EDICION DE FICHEROS

Los comandos que se utilizan para editar un fichero son: cat y more.

1. 1. COMANDO cat

El comando cat (concatenate) se utiliza para visualizar por pantalla el contenido de uno o más ficheros. Cuando se especifica más de un fichero, cat los edita uno detrás de otro.

La sintaxis del comando es: cat [−ns] fichero(s)

−n Numera las líneas.

−s Elimina las líneas en blanco.

fichero(s) Nombre o nombres de los ficheros que se van a editar.

El comando cat no pagina, entonces se utiliza:

CTRL−S para parar la pantalla.

CTRL−Q para continuar con la edición.

El comando cat permite también concatenar ficheros; para ello se pondría:

cat fichero1 fichero2 ... > fichero n

entonces une los ficheros fichero1 fichero2 ... y lo almacena en el fichero n.

NOTA

Si se tuiliza por equivocación el comando cat sin ningún argumento, intenta leer de la pantalla, por lo que no sale el prompt del sistema (se queda como colgada); entonces hay que poner:

CTRL−C o CTRL−D

por lo que debemos de tener cuidado con esto.

Como el comando cat no pagina, cuando queramos editar un fichero que es muy largo, es aconsejable utilizar el comando more.

COMANDO more

El comando more se utiliza para editar ficheros por la pantalla; la

principal diferencia con cat es que se puede controlar el número de líneas

que aparecen en pantalla, utilizando las teclas siguientes:

Con la "barra espaciadora" se avanza una página.

Con la tecla de return se avanza una línea.

Con la tecla DEL ó q se sale de la edición.

La sintaxis de este comando es: more [−cd] [+número de líneas] [+/path]

fichero(s)

−c Edita pantalla a pantalla.

−d Número de líneas que se van a editar.

+número de líneas Número de la línea a partir de la cual se va a editar.

+/path Path correspondiente al fichero que se va a editar.

fichero(s) Nombre o nombres de los ficheros que se van a editar.

Por ejemplo:

more −c10 +25 +/aplicaci/datos

editará 10 líneas, empezando por la 25, del fichero llamado datos que se

encuentra en el directorio aplicaci.

COMANDOS PARA MANIPULAR FICHEROS

Se pueden manipular ficheros creándolos, borrándolos, cambiando su nombre o

cambiando su contenido.

Los comandos que se utilizan para manipular ficheros son: cp, rm y mv.

COMANDO cp

El comando cp se utiliza para copiar ficheros. Su sintaxis es: cp [−i]

fichero entrada fichero destino

−i Origina que el comando requiera una confirmación, en el caso de que el

fichero destino

ya exista; es decir, pregunta si se desea hacer la copia.

fichero entrada Nombre del fichero que se va a copiar.

fichero destino Nombre del fichero en el que se va a copiar el contenido del

fichero de entrada

En el caso de que el fichero destino ya exista, lo va a machacar, por lo que

es recomendable utilizar la opción −i para que nos pida confirmación y así

evitar posibles errores.

Por ejemplo:

cp −i datos datos.new

COMANDO rm

El comando rm se utiliza para borrar ficheros. La sintaxis de este comando

es: rm [−i] fichero(s)

−i Origina que el comando requiera confirmación para ejecutarse.

fichero(s) Nombre o nombres de los ficheros que se van a borrar.

La opción −i se debe de utilizar para pedir confirmación antes de proceder

al borrado.

COMANDO mv

El comando mv se utiliza para renombrar ficheros; es decir, el contenido del

fichero no cambia, sólo cambia el nombre; o para mover ficheros entre

directorios (se verá en el capítulo siguiente). La sintaxis del comando es:

mv [−i] fichero entrada fichero destino

−i

Origina que el comando requiera una confirmación, en el caso de que el

fichero destino

ya exista; es decir, pregunta si se desea hacer la copia.

fichero entrada

Nombre del fichero que se va a copiar.

fichero destino

Nombre del fichero en el que se va a copiar el contenido del fichero de entrada.

En el caso de que el fichero destino exista, lo "machaca".

IMPRESION DE FICHEROS

COMANDO lpr

El comando lpr se utiliza para imprimir el contenido de uno o más ficheros,

por la impresora del sistema. La sintaxis del comando es: lpr [−r]

[−#número] [−p] fichero(s)

−r Borra el fichero una vez que se ha imprimido.

−#número Número de copias que se quieren imprimir.

−p Imprime al principio de cada página una cabecera que incluye la fecha, el

nombre del fichero y el número de página.

fichero(s) Nombre o nombres de los ficheros que se van a imprimir.

Por ejemplo:

lpr −#3 −p datos.dat

imprime 3 copias del fichero datos.dat y además, en el pricipio de cada

página imprime la fecha, el nombre datos.dat y el número de página.

COMANDO lpq

El comando lpq muestra en pantalla la cola de trabajo de impresión. La

sintaxis del comando es:

lpq

COMANDO lprm

El comando lprm se utiliza para suprimir un fichero de la cola de impresión.

La sintaxis del comando es:

lprm identificador

identificador Nombre de un usuario, o número del trabajo, o nombre de un fichero.

MANEJO DE DIRECTORIOS

Sabemos que los directorios son un tipo de ficheros que contienen nombres de ficheros. Para visualizar los nomnbre de los ficheros que contiene se utiliza el comando ls (pg.).

1. CREAR DIRECTORIOS : COMANDO mkdir

El comando mkdir (make directory) se utiliza para crear nuevos directorios.

Su sintaxis es: mkdir [path] directorio

path

Cuando el directorio no se quiere crear en el que se está, hay que indicar el path de donde se quiere crear.

directorio

Nombre del directorio que se va a crear.

Por ejemplo, supongamos que estamos en /usr/mary y que queremos crear un directorio de nombre programa, en el directorio proyecto, entonces se pondría:

mkdir proyecto/programa

Observemos que el directorio proyecto no está precedido por un slash (/), pués es creado como parte del directorio en el que se está, es decir, en usr/mary.

ACCEDER A UN DIRECTORIO : COMANDO cd

El comando cd (change directory) se utiliza para moverse de un directorio a otro. Su sintaxis es: cd [directorio] [..] [.] [~] [~nombre usuario] [directorio]

Nombre del directorio al que se quiere acceder. Si este directorio no es el inmediatamente siguiente al que nos encontramos hay que indicar el pathname correspondiente, bien sea el absoluto o el relativo.

.. Se refiere al directorio inmediatamente anterior al que nos encontramos.

. Se refiere al directorio en que estamos.

~ Para ir directamente al Home directory.

~nombre usuario Para ir al Home directory del usuario especificado.

Con un mismo comando cd se puede avanzar y retroceder en la estructura de árbol, por ejemplo:

cd /../../proyecto/datos

BORRAR DIRECTORIOS

Para borrar directorios se utilizan los comandos : rmdir ó rm −r.

COMANDO rmdir

El comando rmdir (remove directory) se utiliza para borrar un directorio; pero antes de utilizar este comando se deben de borrar todos los ficheros que contenga (incluidos los ficheros invisibles), es decir, el directorio que se va a borrar tiene que estar vacío.

La sintaxis del comando es: rmdir [directorio]

[directorio]

Nombre o nombres de los directorios que se van a borrar.

Si el directorio que se va a borra contiene algún fichero, cuando se ejecute el comando rmdir dará un mensaje de error.

COMANDO rm −r

En el capítulo anterior hemos visto cómo borrar ficheros utilizando el comando rm, ahora veremos cómo utilizar este comando para borrar directorios, ya que éstos son un tipo de ficheros.

El comando rm −r borra recursivamente todos los ficheros que haya en el directorio y depués borra el directorio.

La sintaxis del comando es: rm −r [−i] directorio

−i

Origina que el comando requiera confirmación para borrar cada uno de los ficheros contenidos en el directorio.

directorio

Nombre del directorio que se va a borrar.

Un ejemplo de este comando es:

rm −r −i aplicaci

COPIAR FICHEROS ENTRE DIRECTORIOS: COMANDO cp

Para copiar ficheros entre dos directorios se utiliza el comando cp con la siguiente sintsxis:

cp directorio1/fichero1 directorio2/fichero2

directorio1

Nombre del directorio, o path, donde se encuentra el fichero que se va a copiar.

fichero1

Nombre del fichero que se va a copiar.

directorio2

Nombre del directorio, o path, donde se va a poner la copia del fichero.

fichero2

Nombre que se le va a dar a la copia del fichero. Si se omite, la copia tendrá el nombre del fichero original (es decir, fichero1).

Si se omite directorio2, toma por defecto el directorio en el que nos encontramos; entonces la sintaxis de comando será: cp directorio1/fichero1 .

RENOMBRAR FICHEROS DE OTROS DIRECTORIOS: COMANDO mv

Para renombrar un fichero y/o trasladarlo a otro directorio se utiliza el

comando mv, con formato:

mv directorio1/fichero1 directorio2/fichero2

directorio1

Nombre del directorio, o path, donde se encuentra el fichero que se va a

copiar.

fichero1<=/th>

Nombre del fichero que se va a copiar.

directorio2

Nombre del directorio,o path, donde se va a poner la copia del fichero.

fichero2

Nombre que se le va a dar a la copia del fichero. Si se omite, la copia

tendrá el nombre del fichero original (es decir, fichero1).

Si se omite directorio2, toma por defecto el directorio en el que nos

encontramos; entonces la sintaxis de comando será: mv directorio1/fichero1 .

COPIAR EL CONTENIDO DE UN DIRECTORIO: COMANDO cpall

Para copiar el contenido de un directorio se utiliza el comando cpall, cuyo

formato es:

cpall directorio1 directorio2

directorio1

Nombre del directorio del que se va a copiar el contenido.

directorio2

Nombre del directorio donde se va a copiar el contenido del directorio1.

Ambos directorios deben de existir antes de ejecutar el comando.

COPIAR LA ESTRUCTURA DE UN DIRECTORIO: COMANDO cp −r

Para ello se utiliza el comando cp −r, con el siguiente formato:

cp −r directorio1 directorio2

directorio1

Nombre del directorio cuya estructura se va a copiar.

directorio2

Nombre del directorio donde se va a copiar la estructura del directorio1.

Ambos directorios deben de existir antes de ejecutarse el comando cp −r.

ORDENES BASICAS DE PROCESAMIENTO DE FICHEROS

1. ORDENACION DE FICHEROS: COMANDO sort

El comando sort se utiliza para ordenar un fichero alfabética o numéricamente, en orden ascendente o descendiente.

La sintaxis del comando es:

sort [−d n r f] [−o nombrefichero] fichero

−d

Ordena en modo diccionario. Primero lista las líneas que empiezan con blancos, después las líneas que empiezan por un número, y por último las que empiezan por caracteres alfabéticos.

Hay que tener en cuenta que con esta opción los números se ordenan alfabéticamente, no numéricamente; es decir, 122 aparecerá antes que 21.

−n

Ordena los números de acuerdo con su valor. Cuando se utiliza esta opción, primero aparecen las línes que empiezan por caracteres alfabéticos y luego los que empiezan por número.

−r

Ordena el fichero en orden descendente en vez de ascendente (que es como lo hace por defecto).

−f

Al ordenar, ignora la diferencia entre letras mayúsculas y minúsculas. Si no se utiliza esta opción aparecen primero las líneas que empiezan con letras minúsculas.

−o nombrefichero

Redirecciona la salida ordenada al fichero especificado.

fichero

Nombre del fichero que se va a ordenar.

Por último, si ponemos:

sort +3 −5 frutas

ordena el fichero frutas utilizando los campos 3, 4 y 5.

Veamos un ejemplo:

_____

% cat frutas

naranja

manzana

4 plátanos

12 peras

% sort frutas

12 peras

4 plátanos

mazana

naranja

% sort −n

manzana

naranja

4 plátanos

12 peras

%

_____

2. VISUALIZACION PARCIAL DE FICHEROS

El sistema Unix posee dos comandos para visualizar parcialmente los ficheros: head y tail.

2.1. COMANDO head

El comando head lista las 10 primeras líneas de un fichero. Su sintaxis es:

head [−número] fichero(s)

número

Indica el número de líneas que se van a editar.

fichero(s)

Nombre o nombres de los ficheros que se van a editar.

2.2. COMANDO tail

El comando tail lista las 10 últimas líneas de un fichero. La sintaxis de este comando es:

tail [−número] [−f] fichero

−número

Indica el número de líneas que se van a editar.

−f

Se utiliza para chequear un fichero de forma contínua. Por ejemplo, si mandamos ejecutar un programa y queremos ir viendo la salida según se va ejecutando, utilizaremos esta opción.

El comando sniff es igual que tail −f , pero no es estándar de UNIX, aunque está disponible en COVEX.

fichero

Nombre del fichero que se va a editar.

3. COMANDO wc

El comando wc (word count) cuenta el número de líneas, palabras y caracteres de un fichero. La sintaxis del comando es:

wc [−lwc] fichero

−l

Cuenta únicamente el número de líneas.

−w

Cuenta únicamente el número de palabras.

−c

Cuenta únicamente el número de caracteres.

fichero Nombre del fichero del que se van a contar las líneas, palabras y caracteres.

En la figura siguiente aparece un ejemplo en el que no se incluyen opciones.

_____

% wc prog.f

336 1402 10344 prog.f

%

_____

336 corresponde al número de líneas, 1402 al número de palabras y 10344 al número de caracteres.

4. BUSQUEDA DE CARACTERES: COMANDO grep

El comando grep busca una cadena de caracteres en uno o más ficheros y lista todas las líneas que la contienen. La sintaxis del comando es:

grep [− v l i w n] cadena1 fichero(s)

−v

Lista las líneas que no contienen la cadena de cracteres.

−l

Lista el nombre del fichero que contiene la cadena de cracteres.

−i

Ignora la diferencia entre letras mayúsculas y minúsculas.

−w

Se debe de utilizar cuando la cadena de caracteres es una única

palabra.

−n

Muestra el número de la línea en la que se encuentra la cadena de

caracteres.

cadena

Cadena de caracteres que se quiere buscar.

fichero(s)

Nombre o nombres de los ficheros en los que se quiere buscar la cadena de

caracteres especidficada.

REDIRECCIONAMIENTO DE ENTRADA/SALIDA

Cualquier comando de UNIX necesita recibir información de algún "lugar" y enviar los resultados del procesamiento a algún "lugar", así como los mensajes de error. Estos "lugares" se llaman, respectivamente, STANDAR INPUT, STANDAR OUTPUT y STANDAR ERROR.

El standar input se refiere al medio desde el cual el comando recibe la información. De forma similar, el standar output se refiere al lugar que el comando envia la salida. Cuando se redireccionan los datos el comando recibe o envía la información desde otra fuente.

El standar error se refiere al medio al que se mandan los mensajes de los errores que se cometen al ejecutar un comando.

Normalmente (aunque depende de cada comando), el standar input es el teclado, y el standar output y el standar error es la pantalla.

REDIRECCIONAMIENTO DE LA SALIDA

El símbolo para redireccionar la salida es: > y se utiliza de la siguiente forma:

comando > nombre_fichero

Las siguientes lineas contienen algunos ejemplo de redireccionamiento de la salida utilizando algunos comandos básicos de UNIX.

who > usuarios

Almacena el listado de que origina el comando who en un fichero llamado usuarios.

sort file_1 > file_2

Almacena el contenido ordenado del fichero file_1 en el fichero file_2.

diff file_1 file_2 > difer

Almacena las diferencias entre los ficheros file_1 y file_2 en el fichero difer

head −2* > heads

Almacena las 2 primeras líneas de cualquier fichero en un fichero llamado heads.

ALGUNAS PRECAUCIONES QUE SE DEBEN TENER AL REDIRECCIONAR LA SALIDA

Veamos 2 problemas que pueden ocurrir si accidentalmente cometemos un error:

A.− REDIRECCIONAR LA SALIDA A UN FICHERO YA EXISTENTE

Cuando se redirecciona una salida, el sistema UNIX crea un fichero con el nombre especificado. Cuando el fichero no existe, al redireccionar la salida

a él crea uno nuevo; pero si el fichero existe borra su contenido y

reescribe encima. Afortunadamente, existe una manera de prevenir borrar

ficheros de esta forma sin darnos cuenta, utilizando el comando:

set noclobber

Si tecleamos este comando antes de redireccionar la salida, en el caso de

que el fichero donde se envía la salida ya exista, aparecerá en la pantalla

el siguiente mensaje:

nombre del fichero file exists

y no "machacaría" el contenido del fichero.

El comando set noclobber evita que se :

redireccione una salida a un fichero ya existente

añada un fichero a otro que no existe.

El comando sólo es efectivo para la sesión en que se teclee. Si se quiere

que permanezca de forma permanente, hay que inluir el comando set noclobber

en el fichero .cshrc. del Home directory.

Si en algún momento se quisiera quitar esta protección, hay que teclear >!.

Veamos un ejemplo:

%set noclobber

%cat agenda

contestar carta al señor Alvarez

%date > agenda

agenda : file exists

%cat agenda

contestar carta al señor Alvarez

%date >! agenda

%cat agenda

Mon May 4 08:50:38 CDT 1989

B.− REDIRECCIONAR LA SALIDA A UN FICHERO UTILIZADO COMO ENTRADA.

Veamos con un ejemplo lo que ocurre cuando se redirecciona la salida al

fichero utilizado como entrada.

%cat frutas

plátano

naranja

manzana

%sort frutas>frutas

%cat frutas

%

Observemos que cuando se ejecuta el comando sort el UNIX borra el contenido

del fichero frutas y crea un fichero nuevo de nombre frutas, por lo que

cuando va a ordenar alfabéticamente el fichero frutas, y éste está vacío.

Por lo tanto, hay que tener cuidado de no redireccionar la salida al fichero

utilizado como entrada, pués se perdería la información.

2. AÑADIR LA SALIDA DE UN COMANDO A UN FICHERO

Se puede añadir la salida de un comando al final de un fichero ya existente

sin borrar su contenido. El símbolo que se utiliza para ello es >> ; se hará

de la siguiente forma:

comando>>nombre_fichero

3. REDIRECCIONAMIENTO DEL STANDAR ERROR

Para redireccionar el standar output y el standar error a un fichero, se

utilizan los símbolos >& de la forma siguiente:

comando >& nombre_fichero

Por ejemplo:

cat datos_1 datos_2 >& datos

entonces, el contenido de ambos ficheros, datos_1 y datos_2 se escribe en el fichero datos junto con cualquier mensaje de error que se produzca.

Para añadir la salida de un comando, así como los mensajes de error a un fichero, se utilizan los símbolos >>& de la siguiente manera:

comando >>& nombre_fichero

Por ejemplo:

cat datos>_1 datos_2 >>& datos

entonces, el contenido de los ficheros datos_1 y datos_2 y cualquier mensaje de error que se produzca, se añade al final del contenido del fichero datos.

4. REDIRECCIONAMIENTO DE LA ENTRADA

El símbolo para redireccionar la entrada es < y se utiliza de la siguiente manera:

comando < nombre_fichero

Por ejemplo:

sort < Agenda > Agenda.ord

ordena alfabéticamente el contenido del fichero Agenda y lo almacena en el fichero Agenda.ord

Si se quiere utilizar como entrada parte del contenido de un fichero, habría que poner en el fichero un "string", que puede ser cualquier símbolo excepto, zzFunyzz. Es decir :

comando << string

.....

.....

.....

string

Por ejemplo:

%cat < Madrid.dat << !

777

666

!

%

es decir, se editará por pantalla el contenido del fichero Madrid.dat hasta

el símbolo !

5. PIPES: TRANSFERENCIA DE DATOS

El sistema UNIX permite transferir datos entre diferentes procesos

(comandos). Este proceso se llama " piping", pués "pipe" es el nombre que se

le dá al símbolo utilizado para transferir datos.

El símbolo para "piping" es | y se utiliza de la siguiente manera:

comando_1 | comando_2 | comando_3 | .....

Es decir, el comando_2 utiliza como entrada los resultados obtenidos por el

comando_1; la salida del comando_2 se utiliza como entrada del comando_3, y

así sucesivamente.

Utilizando pipes no es necesario utilizar ficheros temporales ni hacer pasos

intermedios para obtener la información que se desea. Por ejemplo, si

ponemos:

who | sort | lpr

entonces la lista de usuarios conectados al sistema se ordenan

alfabéticamente y se imprime (odenada) por la impresora del sistema.

FILTROS

Un filtro es cualquier comando situado entre dos pipes y manipula los datos

obtenidos por un comando previo antes de utilizarse por el comando situado a

continuación del filtro. Una línea de comando puede contener varios filtros.

En el ejemplo:

who | sort | lpr

el comando sort actúa como filtro.

COMANDO tee

Cuando después de un pipe aparece el comando tee la redirección de la salida

la hace a dos sitios, a un fichero especificado y al standar output:

comando | comando | tee nombre_fichero | comando | .....

En el ejemplo:

who | sort | tee listin | more

la lista de usuarios ordenada alfabéticamente, aparece por pantalla y se

almacena en el fichero listin

AYUDAS "ON−LINE" DEL SISTEMA UNIX

Todo sistema CONVEX posee 2 tipos de ayudas "on−line":

Info System

manual pages.

1. INFO SYSTEM

El Info System de CONVEX es una utilidad interactiva que sirve de ayuda en

cuanto a la forma de operar el sistema UNIX.

Para activar este tipo de ayuda (Info System) se utiliza el comando info.

Este comando dará la información sobre los comandos del UNIX; dicha

información se puede obtener de distintas formas:

a través de un menú

utilizando un nombre tópico del comando

utilizando el nombre del comando.

1.1. DISTINTOS MODOS DE OBTENER INFORMACION

1.1.1. OBTENCION DE LA INFORMACION A TRAVES DE MENUS

Para ello se ejecuta el comando info poniendo simplemente: info

En la figura siguiente se muestra el resultado de la ejecución de este

comando. A partir de ese menú principal, se va seleccionando la opción que

se desee y se obtendrán otros menús. A tarvés de esos menús buscaremos la

información deseada.

CONVEX INFO SYSTEM MAIN MENU

==========

1. Contact other users or machines

2. Use ConvexOS online help

3. Execute Commands

4. Edit, find, print, modify, analyze,

and archive files

5. Develop programs

6. Check user, job, or system status

7. Modify system and file accessibility

8. Perform arithmetic calculations

Enter <1..8>, <q>uit, <?> help, <t>opic/command list, a command name, a

topic

Please type your selection and press <RETURN>:

1.1.2. ACCEDER A LA INFORMACION UTILIZANDO UN NOMBRE TOPICO

Utilizando el comando info, se puede acceder a la información sobre un

comando usando su "nombre tópico". En el caso de que no se conozca su "nombre tópico", podemos obtener la lista de "nombres tópicos" de los comandos, pulsando una t en el menú principal que sale cuando se ejectuta el comando info.

1.1.3. ACCEDER A LA INFORMACION UTILIZANDO EL NOMBRE DEL COMANDO

Si se conoce el nombre del comando del que se quiere obtener información, se puede acceder directamente a la información deseada, sin más que poner el nombre del comando como argumento de info, es decir : info comando

1.2. DESCRIPCION DE UN COMANDO UTILIZANDO EL COMANDO info

La descripción de un comando que aparece en la pantalla cuando utilizamos el comando info se divide en 4 secciones: descripción, sintaxis, ejemplo y referencias:

La sección de descripción incluye la información general sobre lo que realiza el comando que se está describiendo.

La sección de sintaxis incluye la sintaxis o formato del comando.

La sección de ejemplo incluye un ejemplo del comando así como la salida que origina dicho ejemplo.

La sección de referencia tiene 2 partes:

1. "See Also": incluye el número de página o el capítulo del manual en el que está la información mostrada por pantalla.
2. "Related Commands": incluye una lista con los comandos que realizan tareas similares a la del comando que se describe.

Después de estas 4 secciones indica las teclas que hay que pulsar según la operación que deseemos ejecutar: TECLA ACCION

<b> Muestra la pantalla anterior.

<n> Muestra la pantalla siguiente.

<ESC> Muestra el menú principal.

<q> Sale del Info System.

<?> Ayuda.

<m> Muestra la página del manual donde está la información del comando.

<x> Ejecuta el comando del que dá la información.

<r> Vuelve a la información en la que estaba antes de solicitar la ayuda.

2. MANUAL PAGES

Para acceder al "manual pages" se utiliza el comando man. El formato de este

comando es: man comando

comando Nombre del comando de UNIX del que queremos obtener la información.

La salida de este comando tiene varias secciones; en la figura siguiente se

explica el contenido de cada una de ellas. SECCION CONTENIDO

NAME

Nombre del comando y una línea de descripción del comando.

SYNOPSIS

Sintaxis o formato del comando. Los argumentos opcionales van entre

corchetes [].

DESCRIPTION

Explicación de lo que realiza el comando. Si el comando puede incluir

argumentos, éstos se explican en esta sección. Si se incluye algún ejemplo,

aparecerá también en esta sección.

FILES

Nombres de los ficheros que se utilizan en el comando.

SEE ALSO

Nombres de otros comandos que realizan funciones similares a la del comando

descrito o que se pueden utilizar en conjunción con el descrito.

DIAGNOSTICS

Discusión de cualquier mensaje de diagnóstico que produce el sistema.

BUGS

Explicación de los errores más frecuentes.

El comando man admite la opción −k ; entonces el formato del comando es:

man −k palabra clave

−k Origina que en la salida aparezca una lista con todos los comandos

relativos a la palabra clave especificada.

palabra clave Puede ser: una palabra, parte de una palabra, o una frase (en

este caso, la frase va entre comitas sencillas).

NOTA.−

Para obtener más información a cerca del comando man, se pondría:

man man

UTILIZACION DEL C SHELL

Un shell en UNIX es un intérprete de comandos.

El Unix soporta dos tipos diferentes de shells, el Bourne shell y el C

shell. Ambos shells tienen la misma finalidad, pero por diferentes caminos.

El C shell es realmente una ampliación del Bournell shell.

En este capítulo veremos por lo tanto una descripción del C shell. El Cshell

posee un lenguaje de programación similar al lenguaje de programación C, de

ahí el nombre de C shell.

Las capacidades básicas del C shell son :

Crear notaciones taquigráficas para un comando o una serie de comandos.

Ejecutar varios trabajos simultáneamente, con o sin nuestra intervención.

Parar un trabajo y empezar de nuevo.

Ejecutar comandos usados previamente.

Personalizar el ambiente a las necesidades personales.

Escribir programas a nivel de comandos, para realizar cualquier tipo de tarea.

1. CONTROL DE TRABAJOS

Los ordenadores CONVEX tienen sistema multitarea, es decir, se puede utilizar el ordenador para trabajar en uno o más procesos o tareas al mismo tiempo.

Un comando puede ejecutarse en foreground o en background.

Cuando se ejecuta un trabajo en foreground, el prompt del sistema no reaparece hasta que el comando ha terminado de ejecutarse; sin embargo, cuando se ejecuta un trabajo en background reaparecerá el prompt del sistema aunque no se haya terminado de ejecutar el comando.

Para correr un trabajo en background, hay que poner un ampersand (&) al final de la línea de comando, justo antes de presionar <return>.

1.1. PASAR UN TRABAJO DE FOREGROUND A BACKGROUND: COMANDO bg

Hay veces que se está ejecutando un programa en foreground y resulta que es demasiado largo, por lo que nos interesa seguir trabajando. Afortunadamente el UNIX permite pasar un trabajo de foreground a background.

Para pasar un trabajo que se está corriendo en foreground a background, se hace en 2 pasos:

Se suspende el trabajo tecleando y entonces aparecerá el prompt del sistema.

Se manda ejecutar el trabajo en background con el comando bg. La sintaxis de este comando es simplemente: bg

1.2. COMANDO jobs

El comando jobs lista por pantalla todos los trabajos que se han lanzado en background. Su sintaxis es: jobs

La salida de este comando es la siguiente:

[1] + Running bmdp 2d epa.2d1 salepa.2d1

[2] − Running bmdp 2d epa.2d2 salepa.2d2

1ª columna Corresponde al número de orden en que se van a ejecutar los

comandos.

2ª columna Puede estar en blanco, o contener un signo más (+), o un signo

menos (−):

El signo + indica el último trabajo que se ha parardo.

El signo − indica el penúltimo trabajo que se ha parado.

3ª columna Indica si el trabajo está corriéndose en background (Running) o

se ha parado (Stopped).

4ª columna Contiene el nombre del comando que se está corriendo en

background o que está parado.

1.3. PASAR UN TRABAJO DE BACKGROUND A FOREGROUND: COMANDO fg

Algunas veces se necesita pasar un trabajo que se está corriendo en

background a foreground para poder utilizar el programa. Para ello se

utiliza el comando fg cuyo formato es: fg [% número trabajo]

número trabajo Es el número del trabajo que se quiere pasar a foreground.

Este número es el que apareceen la 1ª columna de la salida del comando jobs.

Si no se especifica ningún número, se pasará a foreground el trabajo que

aparezca con un

signo − , en la salida del comando jobs.

2. ALIASES: RENOMBRAR COMANDOS

El c shell permite definir aliases. Un alias es un nemotécnico que

representa a un comando o a un grupo de comandos.

Para definir un alias se utiliza el comando alias, cuya sintaxis es: alias

nemotécnico comando(s)

comando(s) Comando o comandos, que va a contener el alias que se está definiendo. Si se especifican varios comandos, éstos deberán de ir separados por un punto y coma (;) o por pipes (|).

Si el comando especificado contiene algún espacio en blanco, se debe de englobar entre comillas simples (').

Cuando se define un alias utilizando el comando anterior, el nemotécnico tiene efecto hasta que se hace log out (^D), es decir, cada alias que se define es válido para la sesión en la que estamos. Por lo tanto, si se quiere que los alias que se definan se mantengan en otras sesiones, éstos se deben de incluir en el fichero .cshrc.

Por ejemplo:

alias DIR ls −a

Para saber qué aliases se tienen definidos, se pone simplemente: alias entonces aparecerá en la pantalla una lista con todos los aliases que están definidos.

En el ejemplo siguiente se define un alias que ordena y pagina la salida del comando who.

alias who 'who | sort | more'

Para cancelar un alias se utiliza el comando unalias, cuya sintaxis es:

unalias nemotécnico

3. UTILIDAD HISTORY

El C shell posee una característica llamada " utilidad history", que recuerda los últimos comandos ejecutados; normalmente recuerda los 20 últimos. Esta utilidad permite además, repetir cualquier comando o modificarlo antes de ejecutarlo de nuevo.

3.1. COMANDO history

Para visualizar la lista de comandos previamente ejecutados se invoca la utilidad history poniendo simplemente: history

y el sistema UNIX responde dando una lista de los comandos que se han ejecutado desde que se hizo log in, precedidos por un número que se llama identificador de comando. Este identificador es el que permite repetir la ejecución del comando deseado. Para hacer esto se pone: ! nº identificador

El signo ! indica al shell que debe de buscar el comando en la lista de history; y el nº identificador indica que debe de ejecutar el comando identificado por ese número.

Se puede especificar el nº de los últimos comandos que queramos visualizar, así como que la lista aparezca en orden inverso, poniendo: history [−r][n]

−r

Origina que la lista de comandos que se han ejecutado aparezca en orden inverso.

n

Número de comandos que se desean visualizar.

En la figura siguiente se presenta un ejemplo de la utilidad history:

_____

6 /mnt/carmen % history

1 pwd

2 ls −s

3 cd reportaje

4 vi semana12

5 lpr semana12

6 history

7 /mnt/carmen % !5

lpr semana12

8 /mnt/carmen %

_____

Este método permite repetir un comando, pero es necesario visualizar la

lista que origina history para conocer el nº identificador del comando.

Existen otras 2 formas de localizar el comando:

especificando la posición relativa

especificando el nombre del comando.

3.2. REPETICION DE UN COMANDO ESPECIFICANDO SU POSICION RELATIVA

Supongamos que queremos repetir un comnado que ya habíamos ejecutado con

anterioridad, n veces antes que el último comando, entonces pondríamos: !−n

3.3. REPETICION DE UN COMANDO ESPECIFICANDO SU NOMBRE

Para repetir un comando se puede utilizar, todo o parte, del nombre de un

comando, de este modo no es necesario acordarse del nº identificador o del

la posición relativa del comando.

Veamos, de acuerdo con el ejemplo de la figura 1., las distintas

posibilidades que hay para repetir la ejecución de un comando especificando

su nombre o parte del nombre:

Especificando el nombre completo del comando : !nombre comando

En el ejemplo, si ponemos:

!vi

se ejecutará el comando vi semana12

Especificando las primeras letras del comando (sólo es necesario la primera)

: !caracter1 caracter2 ...

En el ejemplo, si ponemos:

!l

se ejecutará el último comando que empiece por l; es decir, lpr semana12

Especificando alguna letra contenida en el nombre del comando : !?caracter?

En el ejemplo, si ponemos:

!?s?

se ejecutará el comando ls −s

3.4. MODIFICACION DE UN COMANDO EJECUTADO PREVIAMENTE

El C shell permite modificar o corregir un error que se haya cometido en un comando ejecutado anteriormente. Para hacer esto se pondría: ^ string antiguo ^ string nuevo

string antiguo contiene el error

string nuevo contiene los caracteres por los que se va a sustituir el string antiguo.

COMUNICACION INTERACTIVA

Para enviar mensajes a otros usuarios del sistema se utilizan los comandos talk y write. Si se necesita obtener información sobre el usuario con el que nos vamos a comunicar utilizaremos el comando finger.

1. INFORMACION SOBRE USUARIOS

Hemos visto que con el comando who se obtiene un listado de todos los usuarios que están conectados al sistema, pero hay veces que queremos obtener información de un usuario que no se encuentra en el listado anterior, para ello utilizaremos el comando finger.

Si se utiliza el comando finger sin ningún argumento, se obtiene un listado semejante al del comando who incluyendo además la oficina y el teléfono. Si el comando se utiliza con uno o más argumentos , se obtendrá una información más detallada acerca de los nombres especificados.

La sintaxis del comando finger es: finger nombre(s)

nombre(s) Nombre o nombres de los usuarios de los que se desea obtener información.

Este comando es muy útil cuando se quiere mandar un mail y no recordamos el código del usuario al que vamos a enviar el mensaje.

2. COMUNICACION CON OTROS USUARIOS

El sistema UNIX posee una utilidad que permite establecer una comunicación interactiva (por pantalla) con otro usuario del sistema; para ello se utiliza el comando talk. La sintaxis de este comando es: talk nombre usuario

nombre usuario Nombre del usuario (login) con el que se desea establecer comunicación.

Cuando se hace una llamada (talk) a un usuario, éste recibe por la pantalla un mensaje avisando que desean establecer conversación. El mensaje incluye el nombre del usuario que ha realizado la llamada. Para establecer comunicación se tecleará talk usuario ; entonces la pantalla se divide en dos, donde escribirán los dos usuarios de la conversación.

Por ejemplo, si alguien me hace un talk, en la pantalla aparecerá:

---

14 /mnt/carmen %

Message from Talk_Daemon@emducms1 at 9:31 ...

talk: connection requested by jc@emducms1.sis.ucm.es.

talk: respond with: talk jc@emducms1.sis.ucm.es

Entonces para establecer la comunicación pondremos:

talk jc@emducms1.sis.ucm.es

Observemos que en este caso hubiese sido suficiente con poner, talk jc pués

la persona que nos ha llamado lo ha hecho a través del ordenador Convex.

Una vez hecho esto, la comunicación queda establecida,apareciendo la

siguiente pantalla

_____

_____

[Connection established]

_____

_____

_____

_____

Para finalizar la comunicación se pulsarán las teclas CTRL+c

Si no se quiere que permitir a otros usuarios comunicarse mediante un talk a

su terminal se utiliza el comando: mesg n

3. ENVIAR UN MENSAJE A OTRO USUARIO

Cuando no se quiere establecer conversación con otro usuario, si no que sólo

se desea mandar un mensaje, se utiliza el comando write, cuya sintaxis es:

write nombre usuario

nombre usuario Nombre del usuario (login) al que se va a enviar un mensaje.