

Comandos Básicos

Introducción al intérprete de comandos de GNU/Linux

Nota de Copyright

© 2005 Diego Chaparro. Algunos derechos reservados.

Este trabajo se distribuye bajo la licencia Creative Commons Attribution-ShareAlike. Para obtener la licencia completa, véase <http://creativecommons.org/licenses/by-sa/2.1/es>

Comandos Básicos

Consejos Básicos

- Mayúsculas distintas de minúsculas
- Cuidado con las flechas
- Al iniciar sesión:
 - No corregir errores al introducir la password
 - Presionar un par de veces ENTER hasta que aparezca el login
- Acordarse de cerrar la sesión

Shells

- También llamado intérprete de comandos.
- Es un programa.
- Es el intermediario entre el usuario y el sistema operativo.
- Algunas shells:
 - ash (smallest)
 - bash (Bourne Again Shell)
 - ksh (Public Domain Korn Shell)
 - tcsh (emula la csh)
 - zsh (una de las más grandes)

Shells

- Cuando se entra en un sistema Linux cada usuario tiene una shell por defecto. En Linux bash.
- Para saber qué shell tenemos:

```
echo $SHELL
```
- Cada letra que escribimos no se manda al SO. Solo cuando presionamos Enter.
- Podemos escribir varios comandos separados por ;

BASH

- Gracias a la librería readline tenemos combinación de teclas para realizar acciones, iguales a las del emacs:

Ctrl+B: Mueve el cursor un carácter hacia atrás
Ctrl+F: Mueve el cursor un carácter hacia adelante
Esc+b: Mueve el cursor al comienzo de la palabra a la izquierda
Esc+f: Mueve el cursor al comienzo de la palabra a la derecha
Ctrl+A: Mueve el cursor al principio de la línea
Ctrl+E: Mueve el cursor al final de la línea
Del: Borra el carácter de la izquierda
Ctrl+D: Borra el carácter de la derecha
Esc+Del: Borra la palabra de la izquierda
Esc+d: Borra la palabra de la derecha
Ctrl+k: Borra desde el cursor al final de la línea

BASH

- Completado de línea:
 - Si empezamos a escribir un comando, pulsamos tabular y busca si hay comandos que empiezan como hemos escrito. Si encuentra solo uno lo muestra, si hay varios emite un sonido y si volvemos a pulsar tabulador nos muestra todas las opciones existentes.
 - Si estamos escribiendo un PATH, también completa buscando un fichero que coincida con lo que estamos escribiendo.
- Historia
 - Bash guarda una lista de los x últimos comandos usados, podemos recuperarlos de varias formas, pero la más sencilla es pulsando los cursores arriba y abajo

Gestión de Ficheros

Pequeña introducción:

- Un sistema de ficheros está compuesto por archivos y directorios
- Organización en forma de árbol
- El elemento principal es el elemento raíz o “/”

Gestión de Ficheros

cd (*Change Directory*): cambiar de directorio

- Al hacer login cada usuario está situado en su directorio de usuario
- *Sintaxis: cd PATH*
- PATH puede ser:
 - absoluto: creado desde el raíz
 - relativo: creado a partir del directorio actual
- Si no ponemos el directorio volvemos al directorio HOME

pwd (*Print Working Directory*): muestra el directorio actual

- *Sintaxis: pwd*

Gestión de Ficheros

- **ls** (*LiSt*): Muestra el contenido de un directorio
 - *Sintaxis: ls nombredirectorio*
 - Si no ponemos nombre de directorio saca el listado del directorio actual
 - Algunas opciones:
 - a Muestra todos los ficheros, ocultos incluidos
 - d Muestra el nombre de un directorio, no su contenido
 - l Formato largo, mostrando detalles
 - r Orden inverso
 - R Listado recursivo
 - color={never,always,auto} Colores basados en tipo de fichero

Gestión de Ficheros

file : Identifica el tipo de los ficheros

- *Sintaxis: file nombrefichero*
- Los tipos pueden ser:
 - Fichero de texto normal
 - Directorio
 - Impresora
 - Disco duro
 - Floppy
 - ...

Gestión de Ficheros

cat (ConcATenate): Concatena ficheros y escribe el resultado en la salida estándar

- *Sintaxis: cat [fichero]*
- Si no indicamos nombre de fichero, la entrada sería la entrada estándar
- Algunas opciones:
 - b Numera las líneas no vacías
 - n Numera todas las líneas
 - v Muestra los caracteres no imprimibles

Gestión de Ficheros

more : Paginador de texto, muestra el texto en la pantalla poco a poco

- *Sintaxis: more [nombrefichero]*
- Opciones:
 - d Muestra mensajes de información
 - num Especifica el número (*num*) de líneas por pantalla
 - +num Empieza a mostrar desde la línea *num*
- Combinación de teclas útiles:

ESPACIO	Muestra la siguiente pantalla de texto
RETURN	Muestra la siguiente línea
q or Q or INTERRUPT	Terminar
=	Muestra el número de línea actual

Gestión de Ficheros

less : Paginador de texto, muestra el texto en la pantalla poco a poco

- *Sintaxis: less [nombrefichero]*
- less añade la siguiente funcionalidad:
 - Permite usar los cursores para ir hacia delante y atrás
 - Navegar mediante número de líneas, porcentaje de fichero
 - No se termina al terminar el fichero
- Combinaciones de teclas útiles:

num + G	Va a la línea número <i>num</i>
g	Va al inicio del fichero
G	Va al final del fichero

Gestión de Ficheros

wc (Word Count) : Muestra el número de caracteres, líneas o palabras de un fichero

- *Sintaxis: wc [filename]*
- Si ponemos el nombre de varios ficheros, mostrará los resultados de cada fichero y el total de todos ellos
- Opciones:
 - c Cuenta el número de caracteres
 - w Cuenta el número de palabras
 - l Cuenta el número de líneas

Gestión de Ficheros

head : Muestra las primeras líneas de un fichero

- *Sintaxis: head [fichero]*
- Por defecto muestra 10 líneas
- Si queremos variar el número de líneas le pasamos el parámetro:

-n

donde *n* es el número de líneas que queremos que se muestren.

Gestión de Ficheros

tail : Muestra las últimas líneas de un fichero

- *Sintaxis: tail [fichero]*
- Por defecto muestra 10 líneas
- Opciones:
 - n Muestra las *n* últimas líneas del fichero
 - +n Muestra el contenido desde la línea *n* hasta el final
 - f Muestra como se va actualizando el contenido de un fichero

Gestión de Ficheros

touch : Cambia la fecha y hora del último acceso o modificación

- *Sintaxis: touch [options] [date] fichero*
- Si el archivo no existe, lo crea con tamaño 0
- Algunas opciones:
 - a Modificar la fecha de acceso
 - m Cambiar la fecha de modificación
 - t time Use la fecha especificada en lugar de la del sistema (MMDDhhmm)

Gestión de Ficheros

cp (CoPy) : Copia archivos y directorios

- *Sintaxis: cp origen destino*
- Por defecto sobrescribe ficheros del mismo nombre
- Algunas opciones:
 - -b No sobrescribe ficheros (hace backup)
 - -i Modo interactivo (pregunta cuando puede haber problemas)
 - -v Muestra los ficheros que han sido copiados
 - -f Forzar la copia
 - -r o -R Copia recursiva

Gestión de Ficheros

- **mv (MoVe)** : Renombra o mueve ficheros
 - *Sintaxis: mv [option] [origen] [destino]*
 - Opciones:
 - b Crea una copia de seguridad de los ficheros que sobrescribe
 - i Pregunta antes de sobrescribir ficheros
 - v Muestra el nombre de los ficheros que ha movido
 - f Fuerza a que se realiza la operación
 - También se usa para mover directorios:
 - Si el directorio destino no existe, el directorio origen es renombrado
 - Si el directorio destino existe, se copia el directorio como un subdirectorío de éste

Gestión de Ficheros

rm (ReMove) : Borra ficheros y/o directorios

- *Sintaxis: rm [option] fichero*
- Opciones:
 - i Pide confirmación antes de borrar
 - f Fuerza el borrado de archivos protegidos contra escritura sin preguntar
 - r Borra ficheros y directorios recursivamente
- Una opción útil, siempre usada con precaución, es *-rf*, que borra recursivamente y sin preguntar.

Gestión de Ficheros

mkdir (MaKe DIRectory) : Crea un directorio

- *Sintaxis: mkdir [option] directorio*
- Por defecto el directori padre debe existir
- Opciones:
 - p Crea todos los directorios padres si no existen
 - m Especifica los permisos que queremos dar al directorio

Gestión de Ficheros

rmdir (ReMove DIRectory): Borra directorios

- *Sintaxis: rmdir [option] directorio*
- Solo borra directorios que estén vacíos
- Opciones:
 - p Elimina los directorios padres también

Redirecciones

Entrada/Salida

- Los programas no suelen estar programados para coger la entrada o salida de un sitio concreto.
- En lugar de eso se define el concepto de entrada estándar y salida estándar.
- La entrada estándar es el dispositivo en el que los comandos reciben los datos de entrada. Por defecto, el teclado.
- La salida estándar suele ser el dispositivo de salida de los programas. Por defecto, el monitor.
- El intérprete de comandos es el que se encarga de establecer la entrada/salida estándar para los comandos que se ejecutan

Redirecciones

- Podemos definir una **redirección** como el cambio de la entrada/salida de un comando
- Tipos de redirecciones:
 - < fichero Hace que la entrada estándar sea el fichero
 - > fichero Hace que la salida estándar sea el fichero, si el fichero existía perdemos su contenido
 - >> fichero Hace que la salida estándar sea el fichero, pero concatena la salida al final de este fichero
 - | Hace que la salida estándar de un comando sea la entrada estándar del siguiente

Redirecciones

Vamos a ver algunos ejemplos:

- `cat > fichero`
- `cat < fichero`
- `cat fichero | less`
- `cat fichero | wc -l`
- `ls > fichero`
- `ls | cat >fichero`
- `cat >> fichero`
- `cat < fichero >> fichero2`

Filtros

grep : Busca línea a línea en un fichero un patrón o una cadena

- *Sintaxis: grep [options] patron [lista ficheros]*
- Hay tres comandos relacionados (grep, egrep y fgrep) que se usan para el mismo objetivo, con unas pequeñas diferencias, egrep usa expresiones extendidas y fgrep usa cadenas en lugar de expresiones regulares
- Opciones:
 - n Muestra el número de cada línea que coinciden
 - c Muestra la cantidad de líneas que coinciden
 - v Muestra las líneas que no coinciden
 - f Le pasamos un fichero que contiene los nombres de los ficheros a buscar

Filtros

grep

- Opciones:
 - i Ignora las diferencias mayúsculas/minúsculas
 - w Busca las coincidencias de la palabra entera
- Hay que encerrar las expresiones con comillas simples. Las dobles pueden funcionar, pero es recomendable usar simples

Filtros

split : Parte un fichero en trozos de 1000 líneas por defecto.

- Cada trozo lo numera como [prefijo].aa, [prefijo].ab, ...
- *Sintaxis: split [opciones] [fichero [prefijo]]*
- Opciones:
 - n Lo separa en trozos de tamaño *n*
 - Especifica que la entrada sea la entrada estándar
- Ejemplos:
 - split fichero
 - split -200 fichero
 - split -200 fichero trozo.

Filtros

- **cut** : Seleccionar columnas de las líneas de un fichero
 - *Sintaxis: cut [opcion] fichero*
 - Opciones:
 - c Seleccionamos las columnas
 - f Seleccionamos los campos separados por el delimitador TAB por defecto
 - d Especificamos el delimitador
 - s No aparece la línea en la que no hay delimitador, por defecto si saldría
 - Ejemplos:
 - `cut -c1-3,7`
 - `cut -d: -f1,2`

Filtros

- **paste** : Concatena las líneas de varios ficheros de entrada
 - *Sintaxis: paste [opcion] [ficheros]*
 - Opciones:
 - d Especificamos el delimitador que queremos que nos ponga entre las líneas de cada fichero. Por defecto es un tabulador
 - Ejemplos:
 - paste f1 f2 f3
 - paste -d: f1 f2 f3

Filtros

- **join** : Enlaza líneas de dos ficheros a partir de un campo clave
 - *Sintaxis: join [opcion] fichero1 fichero2*
 - Opciones:
 - j1 num El campo número *num* es el campo clave del primer fichero
 - o num1.num2 Especifica que la salida sea el campo *num2* del fichero *num1*
 - t delim Especifica el delimitador a usar entre los campos, por defecto es el tabulador o los espacios
 - Join supone que los ficheros están ordenados por el campo clave

Filtros

join

- Ejemplo:
 - Tenemos dos ficheros:

MENU

carne cordero

carne ternera

fruta naranja

pasta macarrones

DIETA

carne antonio

carne juan

fruta antonio

fruta manuel

- join menu dieta
- join menu dieta -o 2.2,1.2

Filtros

tee : Lee de la entrada estándar y lo escribe en un fichero y en la salida estándar.

- Nos vale para crear dos copias, normalmente una para guardarla y otra para procesarla en ese momento.
- Sintaxis: `tee [opcion] fichero`
- Opciones:
 - a Añade al fichero, en lugar de sobrescribirlo

Filtros

sort : Ordena líneas en base a la ordenación de los caracteres ASCII

- *Sintaxis: sort [option] [fichero]*
- Opciones:
 - +num Ordena a partir del campo número *num*
 - num Ordena hasta el campo número *num*
 - t delim Especifica el delimitador de campos, por defecto es tabulador o espacios
 - r Ordenación inversa
 - n Ordenación numérica
- Ejemplos:
 - sort +1 fichero
 - sort -t: +2 fichero

Filtros

uniq : Elimina las líneas repetidas consecutivas

- *Sintaxis: uniq [opcion] [ficheroentrada] [ficherosalida]*
- Opciones:
 - c Muestra el número de veces que se repiten las líneas
 - d Muestra el número de veces que se repiten las líneas, pero solo las que se repiten

Filtros

comm : Compara dos conjuntos. Deben ser dos ficheros ordenados y sin duplicados

- *Sintaxis: comm [opcion] fichero1 fichero2*
- Cada línea es un elemento del conjunto
- Muestra el resultado en tres columnas:
 - Los elementos que están en el conjunto 1 y no en el 2
 - Los elementos que están en el conjunto 2 y no en el 1
 - Los elementos que están en los dos
- Podemos especificar las columnas que NO queremos ver con la opción -col1col2
- Ejemplo: `comm -12 fichero1 fichero2`

Filtros

cmp : Compara cualquier tipo de ficheros.

- *Sintaxis: cmp [opcion] fichero1 [fichero2 [skip1 [skip2]]]*
- Hace la comparación byte a byte. Su principal objetivo es saber si son iguales o no.
- Si son iguales no saca ningún resultado. Si no lo son señala la posición del primer byte en el que difieren

Filtros

diff : Compara ficheros cuyo contenido son textos

- *Sintaxis: diff [opcion] ficheros*
- Solo es de utilidad cuando los ficheros a comparar tienen una mayoría de líneas idénticas
- Muestra las diferencias entre esos dos ficheros:
 - especificando las líneas de cada uno de los ficheros
 - el tipo de diferencia (c: cambio, a: solo presente en el segundo fichero, d: solo presente en el primero)
 - el contenido de cada fichero (< son los datos del primer fichero y > del segundo)
- También tenemos **diff3**, para comparar tres ficheros

Comprimir / Empaquetar

tar : Se usa para empaquetar varios ficheros en uno solo, manteniendo la estructura de directorios

- Se deben usar PATH relativos, nunca absolutos porque habría problemas al desempaquetar
- *Sintaxis: tar [opcion] [fichero_tar] [ficheros]*
- Opciones:
 - c Crea un nuevo archivo
 - f Especifica el nombre del archivo tar
 - t Muestra los archivos contenidos en un tar
 - x Extrae el contenido de un tar

Comprimir / Empaquetar

tar

- Opciones:

- v (Verbose) Muestra lo que va haciendo
- w Modo interactivo. Pregunta cada archivo a extraer
- r Añade un nuevo archivo al un tar existente
- u Actualiza o añade un fichero a un tar existente

- Ejemplos:

- tar -cf archivo.tar . Empaqueta el directorio actual
- tar -tf archivo.tar Ver el contenido de un tar
- tar -xvf archivo.tar Extrae los archivos de un tar

Comprimir / Empaquetar

- **gzip** : Comprime un fichero borrando el archivo original
 - *Sintaxis: gzip [opcion] [fichero]*
 - Opciones:
 - v Muestra información sobre el fichero comprimido, ratio de compresión, etc...
 - Ejemplo: `gzip fichero`

Nos devolvería un fichero de nombre `fichero.gz` y borraría el original
- Podemos empaquetar y comprimir a la vez:
 - `tar -cvzf archivo.tar.gz ó`
 - `tar -cf archivo.tar . | gzip archivo.tar`

Comprimir / Empaquetar

- **gunzip** : Extrae los archivos comprimidos con gzip
 - *Sintaxis: gunzip [opcion] archivo.gz*

Comprimir / Empaquetar

- **zcat** : Sirve para ver ficheros, como cat, pero permite ver el contenido de los ficheros comprimidos con gzip o compress
 - *Sintaxis: zcat fichero_comprimido*
 - Ejemplo:
 - `zcat archivo.tar.gz | tar -t`