1

INTRODUCCIÓN A LA SHELL DE UNIX/LINUX

Unix/Linux

- Unix apareció en 1969 y fue creado en los laboratorios Bell AT&T por Ken Thompson y Dennis Ritchie.
- Unix está completamente escrito en lenguaje C.

- Linux fue creado en 1991 por el finlandés Linus Torvald, fecha en la cual Linux 0.02 ya era capaz de ejecutar la *shell Bash* y el compilador GNU de C *gcc*.
- La filosofía de diseño de Linux está muy influenciada por Minix (*Mini-Unix*), creado por Andrew Tanembaum para un IBM PC.
- Linux es de código abierto. Ello implica que muchos programadores de todo el mundo añadan aplicaciones, formando el proyecto *GNU/Linux*, de donde salen las distribuciones Linux.

INTRODUCCIÓN

- **Unix/Linux es un Sistema Operativo**:

- **Multiusuario**: varios usuarios pueden compartir al mismo tiempo los recursos del ordenador.
- **Multitarea**: cada usuario puede ejecutar al mismo tiempo varias tareas.
- **Multiproceso**: es capaz de trabajar con varios procesadores
- **Multiplataforma** (*portable*)
- **Servidor de red**.

Componentes de Unix/Linux

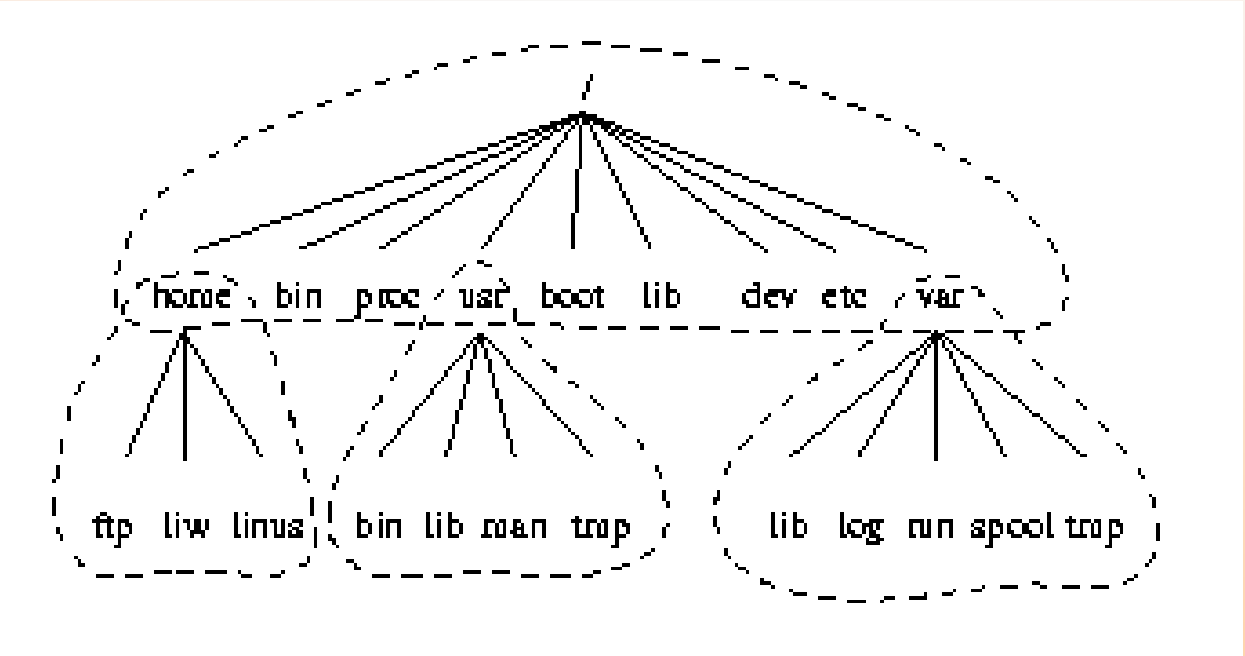
- **Núcleo o kernel**: ejecuta los programas y gestiona los dispositivos hardware.
- **Shell**: proporciona la interfaz de usuario, recibiendo las órdenes de éste a través de scripts y mandándolos al núcleo para ser ejecutadas.
- **Sistema de archivos**: organiza la forma en que se almacenan los archivos.
- **Utilidades**: editores, compiladores y otros programas.

SISTEMA DE ARCHIVOS

El esquema más común de un sistema Linux contiene 4 sistemas de ficheros:

1. / : raíz (root)
2. /usr: aplicaciones y librerías usuario
3. /var: archivos tamaño extensible
4. /home: directorios de usuarios

La filosofía de Unix/Linux es reunir los archivos de acuerdo con su propósito; los comandos están en un sitio, los ficheros de datos en otros, la documentación en otro, etc.



SISTEMA DE ARCHIVOS

/ (directorio raíz)

/bin (Archivos binarios, ejecutables esenciales)

/sbin (Archivos binarios del superusuario, esenciales)

/dev (Archivos controladores de dispositivos "devices")

/etc (Archivos de configuración del sistema)

/tmp (Archivos temporales)

/home (Contiene los directorios personales de cada usuario)

/usr (Aplicaciones para los usuarios)

/var (Archivos de tamaño extensible: impresora, mail, logs, ...)

/proc (Comunicación directa con el núcleo)

/lib (Librerías esenciales para el sistema)

/mnt (Donde se montarán los sistemas de archivos)

SISTEMA DE ARCHIVOS

Tal y como se ha visto, la estructura de directorios en Unix/Linux tiene forma de árbol cuya raíz es el directorio **/** (*root*). De éste cuelgan todos los demás.

Cada vez que un usuario entra en el sistema accede a un directorio *predefinido* para dicho usuario. Este directorio suele llamarse como el usuario y su directorio padre es ***/home***.

.	**representa directoria actual**
..	**representa directorio padre**

Para poder navegar por el árbol de directorios, la shell proporciona el comando **cd** (*change directory*) cuyas opciones son:

Navegación por el árbol de directorios

- **cd /path** : lleva al directorio cuya trayectoria completa es /path
- **cd.** : lleva al directorio actual
- **cd..** : sube al directorio padre
- **cd** (sin parametros) : lleva al *home* de tu usuario
- **cd ~** *nombreusuario* : lleva al *home* de nombreusuario
- **cd ~** : lleva al *home* de tu usuario
- **cd /** : lleva al directorio raíz
- **cd -** : lleva al último directorio visitado.

Concepto de shell

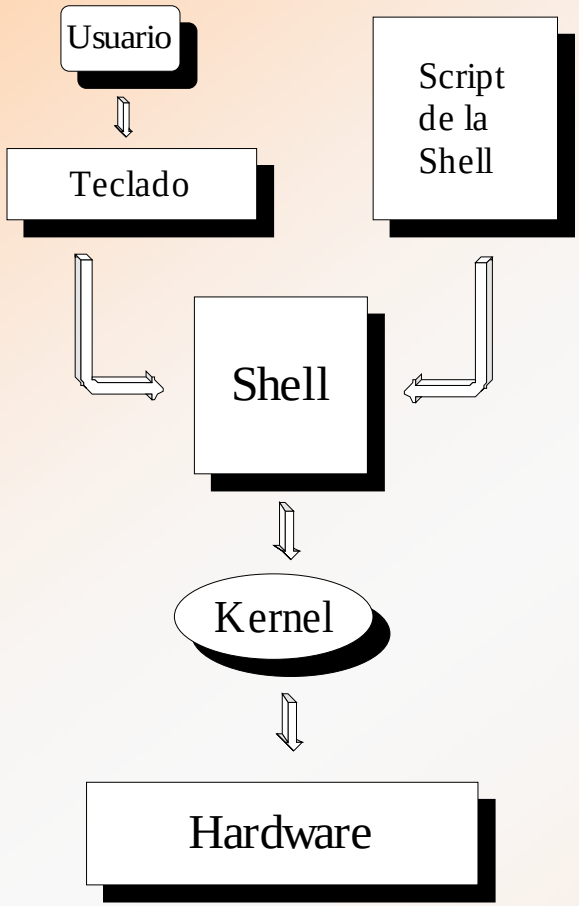
Un **shell** o *intérprete de comandos* es el proceso encargado de traducir los comandos que los usuarios introducen a instrucciones que el sistema operativo entiende. El shell facilita al usuario escribir órdenes en la línea de comandos.

La forma que tiene el sistema operativo de indicar que se encuentra a la espera de una orden es mostrar un símbolo, denominado *prompt* del sistema, seguido del cursor. Habitualmente, el *prompt* del sistema es el carácter $ o el carácter % para los usuarios y el carácter # para el administrador del sistema (root).

Tareas de la shell

- Lee y analiza la entrada de la línea de comandos.
- Maneja caracteres especiales, redirecciones, tuberías y control de trabajos (en primero o segundo plano).
- Busca el comando en el disco y si lo encuentra, lo ejecuta. Esto se llama utilizar la shell interactivamente.
- Maneja señales.
- Prepara la ejecución de programas.

SHELL



Si quieren ejecutarse repetidamente una serie de comandos, pueden escribirse archivos cuyo contenido son dichos comandos. Esto archivos se denominan *scripts*. Incluyen mecanismos para evaluar condiciones, realizar saltos, ejecutar bucles, de forma similar a como lo hace un lenguaje de programación. La figura muestra la forma de trabajo de la shell y otros componentes del PC.

Principales shells de Unix/Linux

- Shell de Bourne (*bsh*, de ATT)
- *C shell* (*csh*, de Berkeley)
- Shell de *Korn* (*ksh*, extensión de la shell de Bourne).

Las tres se comportan de forma similar, pero divergen a la hora de programar scripts (archivo conteniendo un conjunto de órdenes).

La shell de Bourne es la shell estándar en modo de *superusuario*, y la que se usa para para administrar los sistemas Unix. En ella está escrita la mayoría de los scripts de administración. Se arranca con el comando */bin/*bsh. El símbolo que la acompaña es '$'.

La **shell de C** añade cierto número de características, como la historia de los comandos ejecutados, alias, completado de nombres de ficheros, aritmética y control de trabajos. Es más lenta para los mismos scripts escritos en la shell de Bourne. Se arranca con el comando ***/bin/csh***. El símbolo que la acompaña es '%'.

La **shell de Korn** es un superconjunto de la shell de Bourne. Dispone de características extras de las de la C shell, como alias, funciones, expresiones regulares con 'comodines' (* y ?), artimética, control de trabajos, coprocesamiento, y características especiales de depurado. Se arranca con el comando ***/bin/ksh***. El símbolo que la acompaña es '$'.

METACARACTERES

- * : sustituye a cualquier número de caracteres dentro de un texto.
- ? : sustituye a un único carácter dentro de un texto.
- | : tubería o pipe. Utiliza la salida de un comando como entrada a otro.
- > : redirecciona la salida estándar hacia un archivo, creándolo si no existe o sustituyendo su contenido si es que ya existe.
- >> : redirecciona la salida estándar hacia un archivo, creándolo si no existe o añadiendo nueva información si es que ya existe.
- 2> : idéntico a > pero redireccionando hacia la salida estándar de errores
- 2>>: idéntico a >> pero redireccionando hacia la salida estándar de error
- & : ejecuta un proceso en segundo plano o background
- \ : carácter de escape. El siguiente carácter posterior a éste se ignora
- [..] : sustituye cual valor incluido entre los corchetes.

METACARACTERES

Ejemplos

- **c?** : incluye c1, c2, cb, ck, c_, etc.
- **c?b??** : incluye c1b12, chbk2, etc
- **a*** : incluye todos los términos que empiezan por *a*.
- ***a*** : incluye todos los términos que contienen el carácter *a*.
- **c[12a]** : incluye a c1, c2, ca.
- **c[1-4]**: incluye c1, c2, c3 y c4.
- **c[!xy]** : incluye todos los términos que empiezan por *c* y su segundo carácter no es ni *x* ni *y*.

ENTRADA/SALIDA ESTÁNDAR

- **Un sistema Unix/Linux dispone de tres formas para comunicarse con el exterior:**

- **Entrada estándar**: se utiliza para introducir datos en la shell. Abre el descriptor 0 (**stdin**).

- **Salida estándar:** se utiliza para mostrar datos al ejecutar órdenes o procesos. Abre el descriptor 1 (**stdout**).

- **Errores estándar:** se utiliza para mostrar errores al ejecutar órdenes o procesos. Abre el descriptor 2 (**stderr**). Por defecto estos errores aparecen por la salida estándar. Sin embargo, es posible redirigirlos hacia la salida de errores estándar mediante los metacaracteres **2>** o **2>>,** que veremos posteriormente

Redirecciones

Es el mecanismo por el cual se dirige la entrada o la salida estándar de un comando desde o hacia un archivo.

- Para redirigir la entrada estándar:

orden < fichero (orden “lee” desde fichero)

- Para redirigir la salida estándar:

orden > fichero (orden “escribe/sobreescribe” en fichero)

- Si se utiliza el operador ‘>>’, la salida del comando se añade al final del archivo:

orden >> fichero (orden “añade datos” a fichero)

Tuberías

La tubería (el carácter '|') permite utilizar la salida de un comando para servir como entrada de otro. Es una herramienta muy importante en Unix/Linux.

```
ls –l | more
ls -l | grep txt
```

En estos dos ejemplos *ls –l* es un comando que muestra una relación de los archivos del directorio actual. El comando *more* para la salida cuando la pantalla se llena y se queda a la espera de teclear algo. Y el comando *grep* con un parámetro busca dentro de un archivo si existe el patrón indicado en el parámetro. Luego *ls –l | grep txt* presentará por pantalla aquellos archivos que contengan en su interior la cadena de caracteres "txt".

Tuberías (cont.)

El siguiente ejemplo muestra una orden compuesta que ordena todos los ficheros con extensión ".txt", elimina las líneas duplicadas y guarda los datos en el fichero "resultado.sal".

```
cat *.txt | sort | uniq > resultado.sal
```

Este otro realiza una copia de un fichero convirtiendo a mayúsculas todos los caracteres del fichero original.

```
cat fich | tr 'a-zñáéíóúü' 'A-ZÑÁÉÍÓÚÜ‘ > fich.sal
```

Archivos

En Unix/Linux el elemento básico de organización de la información es el **archivo**.

Un archivo es un *conjunto de bytes* tratados como una unidad y referenciados por un *nombre*.

En Unix/Linux tanto los ficheros como directorios como todo tipo de dispositivos de E/S, son tratados como archivos.

Tipos o modos de ficheros:

Los ficheros en Unix/Linux son de varios tipos:

- fichero regular: archivo normal.

d directorio: contiene otros ficheros y directorios.

b dispositivo de bloque: la unidad de las operaciones de E/S es el bloque

c dispositivo de carácter: las operaciones de E/S se realizan en forma de carácter.

l enlace simbólico: son sinónimos de otros ficheros.

p tubería con nombre: comunica la salida de un proceso con la entrada de otro.

s socket: comunica diferentes procesos (socket de comunicaciones).

FICHEROS, USUARIOS Y PERMISOS

Un sistema Unix/Linux es multiusuario, por lo que los archivos de cada usuario deben estar **protegidos** del resto de usuarios. Unix/Linux dispone de tres tipos de permisos y tres tipos de usuarios.
Cada usuario puede realizar una serie de operaciones sobre un fichero, operaciones tales como *leerlo, modificarlo* o *ejecutarlo*. Estas acciones están contempladas en lo que se denomina **permisos del fichero**.

- **Usuarios**:
  - **usuario** o propietario (user)
  - **grupo** (group): conjunto de usuarios. Cada usuario pertenece al menos a un grupo.
  - **otros** usuarios (others): restos de usuarios y que no están en nuestro grupo.

- **Permisos:**
  - **lectura (r):** permite leer el contenido de un archivo o listar el contenido de un directorio.
  - **escritura (w):** permite modificar y borrar un archivo. En el caso de un directorio permite crear y borrar archivos dentro del directorio.
  - **ejecución (x):** permite ejecutar archivos o entrar en directorios.

Estos tres permisos pueden ser fijados para cada uno de los tres tipos de usuarios. De esta manera, un archivo o directorio tendrá una cadena de 3x3=9 caracteres indicando los permisos.

u	g	o
rw-	r--	r--

Permisos de lectura y escritura para el usuario y solamente de lectura para el grupo y resto de usuarios.

Además, hay 8 formas de combinar los permisos por cada usuario. Ello implica que podrán numerarse cada una de dichas maneras desde el 0 al 7.

Val.	Permisos
0	---
1	--x
2	-w-
3	-wx
4	r--
5	r-x
6	rw-
7	rwx

Si asignamos:

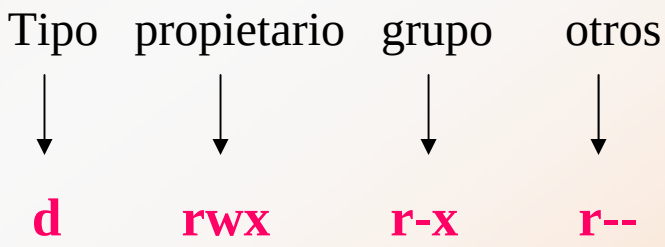
- valor 1 al permiso de ejecución
- valor 2 al de escritura
- valor 4 al de lectura

tendremos los valores que aparecen en la figura de la izquierda.

De esta forma rw- r-- r-- se transforma en el número con dígitos en octal **6 4 4**

Poniendo todo junto:

Se pueden mostrar los permisos de un archivo a través del comando ls –l (se verá más adelante). Dicho comando muestra, entre otros datos, una máscara de 10 caracteres de los cuales el primero indica el tipo de fichero y los nueve restantes son los permisos.



Este fichero es un directorio. El propietario podrá recorrer dicho directorio (r), crear y borrar ficheros dentro del directorio (w) y puede acceder a dicho directorio.

Ejemplos

- chmod u+w hola.c : añade permiso de escritura sobre el archivo hola.c al propietario.
- chmod o-r hola.c: suprime el permiso de lectura de hola.c al resto de usuarios.
- chmod rw hola.c: añade permiso de lectura y escritura sobre el archivo hola.c a todos los usuarios.
- chmod rw *.c: añade permiso de lectura y escritura sobre todos los archivos con extensión .c a todos los usuarios.
- chmod 644 hola.c: establece el permiso de lectura y escritura para el propietario y de lectura para el grupo y resto de usuarios.
- chmod = hola.c: desactiva todos los permisos de hola.c
- chmod 000 hola.c: idéntico a chmod = hola.c

ÓRDENES BÁSICAS

1. Órdenes de manejo de directorios

- ls: listado del contenido del directorio:
  - ls –a: incluye los archivos ocultos (empiezan por .)
  - ls –l: listado en formato ‘largo’
  - ls –t: ordena la salida por fecha
  - ls –R: listado recursivo

- cd: cambio de directorio (ya visto anteriormente)
- pwd: muestra del directorio de trabajo actual
- mkdir: creación de un nuevo directorio
- rmdir: borrado de un directorio.

2. Órdenes de manipulación de ficheros (I)

- man: muestra las páginas del manual asociado a un comando
- cat: concatena archivos y muestra el resultado por la pantalla
- more: muestra contenido de archivos de pantalla en pantalla
- cp: copia uno o mas archivos
  - -R: Copia un directorio recursivamente.
  - -p: Copia preservando permisos, propietario, grupos y fechas.
  - -d: Conserva los enlaces simbólicos como tales y preserva las relaciones de los duros.

    -*a*: Lo mismo que -dpR.

2. Órdenes de manipulación de ficheros (II)

- rm: borra archivos
  - -r: borrado recursivo, es decir, de subdirectorios
  - -f: no hace preguntas acerca de los modos de los archivos
  - -i: interactivo, solicita confirmación antes de borrar cada archivo.
- mv: cambia de nombre o mueve de sitio un archivo
- wc: cuenta líneas, palabras y caracteres dentro de un fichero
- sort: ordena las líneas de un archivo y las muestra por la pantalla
  - -n: ordena teniendo en cuenta los números
  - -f: no tiene en cuenta mayúsculas ni minúsculas
  - -r: ordena de forma inversa

2. Órdenes de manipulación de ficheros (III)

- **diff**: muestra las diferencias entre dos archivos

- **cut**: muestra columnas o campos de caracteres. Su formato básico es

  cut -f | c campos -d delimitador

  -fnum: especifica el campo *num*
  -fnum1, num2: especifica los campos *num1* y *num2*
  -fnum1-num2: especifica los campos desde *num1* a *num2*
  -cnum1-num2: especifica los columnas desde *num1* a *num2*

find: busca archivos y directorios y ejecuta comandos sobre ellos.

-name patrón: busca archivos cuyo nombre se encuentra en *patrón*
-size tamaño: busca archivos cuyo tamaño máximo es *tamaño*
-type tipoarchivo: busca archivo del tipo especificado por *tipoarchivo*

2. Órdenes de manipulación de ficheros (IV)

- **head**: extrae las primeras líneas de un fichero (por defecto 10 líneas).

  -n: muestra las *n* primeras líneas del fichero

- **tail**: extrae las últimas líneas de un fichero (por defecto 10 líneas).

  -n: muestra las *n* últimas líneas del fichero

  -f: muestra las últimas 10 líneas, refrescando cada vez que un proceso añade datos al archivo. Es muy útil para seguimiento de archivos de trazas (.log).

  +n: muestra el texto a partir de la línea número *n*

3. Órdenes de estado

- date: muestra la fecha y hora actuales
- ps: muestra los procesos en ejecución, y su estado
- finger: muestra información sobre los usuarios trabajando en el sistema
- ping: muestra si una máquina remota está encendida
- who: muestra qué usuarios están en el sistema, junto con el puesto que ocupan y la hora de entrada
- w: muestra información sobre los usuarios, tiempo de CPU, tiempo desocupado, procesos ejecutándose, etc.
- hostname: devuelve el nombre de la máquina
- uname: información sobre el sistema operativo

3. Órdenes de estado (II)

- **free:** información sobre la cantidad de memoria disponible y usada
- **last**: información sobre los últimos usuarios que han entrado en el sistema
- **du**: muestra el espacio ocupado por un directorio
- **set**: información sobre el entorno del usuario actual
- **/sbin/route**: información sobre la tabla de rutas de nuestro sistema
- **/sbin/ifconfig**: información sobre los distintos dispositivos de red de la máquina
- **/sbin/netstat**: información sobre las conexiones a nuestro sistema y desde nuestro sistema.

4. Órdenes de tratamiento de cadenas de caracteres (I)

- grep: muestra la existencia de ocurrencias de una cadena.

  -i: insensible a mayúsculas u minúsculas

  -r: busca recursivamente en subdirectorios

  -v: muestra las líneas que no tienen la cadena

Ejemplos:

grep PATH .bashrc (busca la cadena PATH en el archivo *.bashrc*)

grep autor *.c (busca la cadena autor en los archivos con extensión *.c*)

grep –r autor . (busca la cadena autor en el directorio actual y recursivamente en sus subdirectorios)

who | grep marga (comprueba si el usuario *marga* está conectado en este momento)

4. Órdenes de tratamiento de cadenas de caracteres (II)

- sort: ordena uno o más ficheros en secuencia, alfabética o numéricamente.

  -n: ordena teniendo en cuenta los números

  -f: no tiene en cuenta mayúsculas ni minúsculas

  -r: ordena de forma inversa

  +n: ordena a partir del campo n+1

- tr: traduce o borra caracteres de la entrada estándar.
- uniq: borra líneas duplicadas dentro de un archivo

Ejemplos

- cp a1.log a2.log: crea una copia del archivo *a1.log* en *a2.log* dentro del mismo directorio .
- cp -R dir1 dir2: copia el directorio *dir1* y todos sus directorios de forma recursiva en el directorio *dir2*..
- rmdir dir1: elimina el directorio *dir1* si es que está vacío..
- rmdir –r dir1: elimina el directorio *dir1* y sus subdirectorios de forma recursiva.
- ping 192.168.2.7: comprueba si existe comunicación de red con la máquina con dirección IP 192.168.2.7
- who | wc –l : cuenta el número de usuarios conectados en el sistema.

Ejemplos (cont.)

- **sort entrada.txt**: muestra una salida ordenada alfabéticamente de cada una de las líneas del archivo *entrada.txt*.
- **sort –r entrada.txt**: muestra una salida ordenada alfabéticamente mayor a menor.
- **sort +2 entrada.txt**: igual que sort pero tomando como entrada las líneas a partir del campo 2.
- **find /home -name *.c**: busca todos los ficheros con extensión *.c* dentro del directorio /home
- **find -user jcarlos *.txt**: busca ficheros del usuario *jcarlos* con extensión .txt
- **find / –size +100k**: busca todos los ficheros que ocupan más de 100kb
- **find /home -type d**: busca solamente los directorios de /home

Ejemplos (cont.)

- tr ‘[a-z]’ ‘[A-Z]’ < entrada.txt : muestra por pantalla el archivo *entrada.txt* con todos sus caracteres en mayúsculas.
- cut -f1,2,5,7 direcciones.txt : muestra las columnas 1,2,5 y 7 del archivo *direcciones.txt*. Las columnas deben estar separadas por tabuladores.
- cut –d: -f1,2,5,7 direcciones.txt : muestra las columnas 1,2,5 y 7 del archivo *direcciones.txt*. Las columnas deben estar separadas por el carácter “:” que actúa de delimitador.
- cut –d; -f1,2,5,7 direcciones.txt : idem que en el caso anterior pero ahora el delimitador es el carácter “:”