



CEU

*Fundación San Pablo  
Andalucía*

| 25<sup>o</sup> aniversario

# SISTEMAS INFORMÁTICOS (DISEÑO APLICACIONES MULTIMEDIA Y DISEÑO DE APLICACIONES WEB) TEMA 9: INTRODUCCIÓN A LINUX

Profesor: Rafael Madrigal Toscano

# TEMA 9: INTRODUCCIÓN A LINUX

1. HISTORIA GENERAL
2. INSTALACIÓN Y GESTIÓN DEL SISTEMA (UBUNTU)
3. ARQUITECTURA DE SISTEMA
4. SHELLS Y EDITORES
5. ESTRUCTURA DE DIRECTORIOS
6. FLUJO DE DATOS
7. PERMISOS
8. ENTORNO DE TRABAJO BASH
9. PROCESOS

# TEMA 9: INTRODUCCIÓN A LINUX

**10. DISPOSITIVOS DE ALMACENAMIENTO.MONTAJE Y DESMONTAJE**

**11.MANEJO DE PAQUETES**

# TEMA 9: INTRODUCCIÓN A LINUX

## 8. ENTORNO DE TRABAJO EN BASH

Para asignarle valor a una variable en el shell se emplea el operador de asignación tradicional (=) entre el nombre de la variable y el valor asignado (no deben haber espacios intermedios).

```
MENSAJE="Hola Mundo"
```

Para acceder al valor de una variable en el shell se emplea el carácter \$ seguido por el nombre de la variable. Para imprimir en la terminal el valor de una variable se puede utilizar el comando echo.

```
echo $MENSAJE
```

Dentro de un shell se pueden ejecutar otros shells que serían hijos del primero (subshells) heredando todo o parte del entorno de trabajo del padre. Para que una variable mantenga su valor en los shells hijos es necesario indicarlo explícitamente mediante el comando export.

```
$ export MENSAJE
```

# TEMA 9: INTRODUCCIÓN A LINUX

## 8. ENTORNO DE TRABAJO EN BASH

Tanto la asignación del valor como el exportar una variable se pueden hacer a la vez:

```
$ export MENSAJE="Hola amigo"
```

Para ver las variables del entorno definidas se puede emplear el comando `set`. Este además se relaciona con las opciones que es otra forma de regir el comportamiento del shell.

Las opciones se activan (on) o desactivan (off).

Estas se utilizan para indicar propiedades del shell muy específicas por lo que no nos vamos a detener en ellas.

Si se desea conocer más al respecto se puede hacer `$ help set | less`.

Para eliminar el valor de una variable se emplea el comando `unset`.

```
$ unset MENSAJE
```

# TEMA 9: INTRODUCCIÓN A LINUX

## 8. ENTORNO DE TRABAJO EN BASH

Algunas variables del entorno en bash son:

- ❑ **PATH:** guarda la secuencia de caminos donde el shell busca los programas que se intenten ejecutar en la línea de comandos cuando no se utilizan los caminos absolutos.

Estos caminos se separan por el carácter : (dos puntos)

Ejemplo: `$ echo $PATH`

`/bin:/usr/bin:/usr/X11R6/bin:/usr/local/bin`

Para añadir un nuevo camino puede hacerse:

`export PATH=$PATH: /bin`

- ❑ **USER:** contiene el login del usuario actual.
  - ❑ **PAGER:** almacena el paginador utilizado por defecto por algunos programas. Por ejemplo, el comando `man` utiliza esta variable para determinar que paginador empleará, aunque primero chequea otra variable llamada `MANPAGER` y si esta no tiene valor entonces acude a `PAGER`, que de no tener valor tampoco, asumirá al `less` como paginador por defecto. También asociada a `man` existe la variable `MANPATH` donde se especifican los caminos de los manuales que despliega `man`, y `LANG` para indicar el idioma.
- Ejemplo: `$ export PAGER=more`  
`$ man bash`

# TEMA 9: INTRODUCCIÓN A LINUX

## 8. ENTORNO DE TRABAJO EN BASH

Algunas variables del entorno en bash son:

❑ HOME: guarda el directorio base del usuario actual.

Ejemplo: `$ echo $HOME`

❑ EDITOR: contiene el editor por defecto del usuario actual. De no tener valor asociado se utiliza vi. En entornos gráficos se pueden indicar editores visuales aunque para ello se prefiere emplear la variable VISUAL.

❑ PS1: almacena la estructura del prompt principal del usuario. Permite una serie de macros.

`\d` : contiene la fecha del sistema.

`\h` : contiene el nombre de la máquina.

`\T` : contiene la hora del sistema.

`\u` : contiene el login del usuario.

`\w` : contiene el nombre completo del directorio de trabajo actual.

`\W` : contiene la base del directorio actual (Ejemplo: para `/home/pepe/doc` la base es `doc`).

`\$` : si el ID del usuario es 0 (root) contiene el valor `#` y sino, `$`.

`\#` : contiene el número del comando actual desde la creación del shell.

El prompt principal por defecto tiene la forma: `"[\u@\h \W]\$ "`

Ejemplo:

`$ export PS1="[\T,\u\#]\$ "`

# TEMA 9: INTRODUCCIÓN A LINUX

## 8. ENTORNO DE TRABAJO EN BASH

Algunas variables del entorno en bash son:

- ☐ PS2: guarda el prompt secundario. Este es el que se presenta cuando el shell no puede interpretar lo que se ha escrito hasta el momento.
- ☐ PWD: contiene el directorio de trabajo actual. Esta variable la pueden emplear algunos programas para saber desde donde se invocaron.
- ☐ SECONDS: almacena la cantidad de segundos transcurridos desde la invocación del shell actual.



# TEMA 9: INTRODUCCIÓN A LINUX

## 8. ENTORNO DE TRABAJO EN BASH

### **FICHEROS PERFILES**

Para cada usuario existen tres ficheros muy importantes que permiten definir en gran medida las características del shell durante la interacción con este.

Estos constituyen shells scripts y se conocen como ficheros perfiles:

1. **.bash\_profile** ó **.profile** : se ejecuta siempre que se abra una nueva sesión en el sistema. Cada vez que un usuario se conecte al sistema se ejecutará el script `.bash_profile`, en el caso de que se utilice `bash` como shell. Para ser compatible con sus versiones anteriores, `bash` en caso de que no existiera `.bash_profile`, ejecuta `.bash_login`, o sino, `.profile`. En este fichero se pueden colocar las asignaciones a las variables del entorno siendo debidamente exportadas a través de `export`. También se puede establecer la máscara de permisos usando `umask`.
2. **.bash\_logout**: se ejecuta al terminar una sesión de trabajo.
3. **.bashrc**: se ejecuta siempre que se invoque un nuevo shell. Por lo general en él se colocan las definiciones de funciones y los alias de comandos. Cuando se crea un nuevo usuario se le colocan en su directorio base estos tres ficheros cuyos patrones están el directorio `/etc/skel`. Estos ficheros son los que ajustan los perfiles para el shell. Aparte, existen en el home de cada usuario otros ficheros que cargan perfiles para los entornos gráficos como el `gnome`, perfiles para el `open office`, etc.

# TEMA 9: INTRODUCCIÓN A LINUX

## 8. ENTORNO DE TRABAJO EN BASH

### ***EJECUCIÓN EN BASH***

Existen dos formas de ejecutar un shell script (fichero con instrucciones bash):

1. Invocándolo directamente por su nombre. Para esto el camino al fichero debe encontrarse en la variable PATH del entorno, sino será necesario especificar el camino completo del fichero. El fichero debe además, poseer los permisos de ejecución adecuados. Lo que ocurrirá en este caso es que se creará un shell hijo que será el que realmente ejecute (interprete) al script.
2. Utilizando el comando source. De esta forma se ejecutará el script en el shell actual. En este caso no será necesario que el fichero correspondiente posea permisos de ejecución. El comando se puede sustituir por el carácter punto. Por ejemplo, si se modificara el .bash\_profile no será necesario, para activar los cambios, desconectarse y conectarse nuevamente al sistema, simplemente se puede hacer:

```
$ source .bash_profile ó $ . .bash_profile
```

# TEMA 9: INTRODUCCIÓN A LINUX

## 8. ENTORNO DE TRABAJO EN BASH

### **EJECUCIÓN EN BASH**

#### Ejercicio

Un ejercicio interesante puede ser hacer un shell script (editar un fichero) llamado program.sh con la siguiente línea:

```
echo $SECONDS
```

Ejecutar luego: `$ . program.sh` (que es lo mismo que `$ source program.sh`) por último asignarle al fichero permisos de ejecución para ejecutarlo de la forma tradicional:

```
$ chmod a+x program.sh
```

```
$ ./program.sh
```

¿Podéis explicar las diferencias en la salida?

# TEMA 9: INTRODUCCIÓN A LINUX

## 9. PROCESOS

Un proceso es una instancia de un programa en ejecución. En Linux se ejecutan muchos procesos de forma concurrente aunque realmente sólo uno accede al procesador en un instante de tiempo determinado. Esto es lo que caracteriza a los sistemas multitarea.

Cada proceso en el momento de su creación se le asocia un número único que lo identifica del resto. Además a un proceso están asociadas otras informaciones tales como:

- ☐ El usuario que lo ejecuta.
- ☐ La hora en que comenzó.
- ☐ La línea de comandos asociada.
- ☐ Un estado. Ejemplos: sleep, running, zombie, stopped, etc.
- ☐ Una prioridad que indica la facilidad del proceso para acceder a la CPU. Oscila entre -20 y 19, donde -20 es la mayor prioridad.
- ☐ La terminal donde fue invocado, para el caso de que este asociado a alguna terminal.

# TEMA 9: INTRODUCCIÓN A LINUX

## 9. PROCESOS

Para ver los procesos y sus características se emplea el comando `ps`.  
Una salida típica de este comando es:

[Enlace](#)

```
cgermany@Galactica:~$ ps a
PID TTY STAT TIME COMMAND
783 tty4 Ss+ 0:00 /sbin/getty -8 38400 tty4
789 tty5 Ss+ 0:00 /sbin/getty -8 38400 tty5
799 tty2 Ss+ 0:00 /sbin/getty -8 38400 tty2
801 tty3 Ss+ 0:00 /sbin/getty -8 38400 tty3
805 tty6 Ss+ 0:00 /sbin/getty -8 38400 tty6
961 tty1 Ss+ 0:00 /sbin/getty -8 38400 tty1
1369 tty7 Ss+ 0:05 /usr/bin/X :0 -nr -verbose -auth /var/run/gdm/auth-f
2808 pts/0 Ss 0:00 bash
2841 pts/0 R+ 0:00 ps a
cgermany@Galactica:~$ ps u
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
cgermany 2808 0.1 0.6 5752 B112 pts/0 Ss 02:10 0:00 bash
cgermany 2842 0.0 0.2 2712 1060 pts/0 R+ 02:12 0:00 ps u
cgermany@Galactica:~$
```



**ps** = display processes running in current shell.  
**ps -e** = display running daemons.  
**ps -f** = display processes with full options.  
**ps -l** = list more information on processes.  
**ps -ef** = display all processes and daemons on all ttys.  
**ps a** = list all processes that run on terminals.  
**ps x** = list all processes that do not run on terminals.  
**ps aux** = list all processes running on or off of terminals and format the results with additional information (similar to a long listing).

**UID** = user ID  
**PID** = process ID  
**PPID** = parent process ID  
**C** = CPU utilization  
**STIME** = time started  
**TTY** = terminal process is running on  
**TIME** = time process has taken on CPU  
**CMD** = the command, program or process

# TEMA 9: INTRODUCCIÓN A LINUX

## 9. PROCESOS

Como puede apreciarse para cada proceso se muestra

- ✓ su ID (identificación o numero),
- ✓ la terminal desde donde se invocó,
- ✓ el tiempo de CPU que se le ha asignado hasta el momento y el comando que lo desencadenó.

Por defecto ps muestra en formato reducido los procesos propios del usuario y la terminal actual. Algunas opciones son:

- x : muestra todos los procesos del usuario actual sin distinción de terminal.
- a : muestra todos los procesos de todos los usuarios.
- f : muestra las relaciones jerárquicas entre los procesos.
- e : muestra el entorno de cada proceso.
- l : utiliza un formato más largo (muestra más información).
- u : utiliza un formato orientado a usuario.

Otro comando para ver el estado de los procesos en ejecución es top, que permite hacerlo dinámicamente.

top es más bien un programa interactivo con el cual se pueden observar los procesos más consumidores de CPU por defecto.

Este comportamiento se puede modificar tecleando:

Para observar todos los posibles comandos durante la interacción con top se puede pulsar h. Para salir se presiona q.

- M: ordenará según el empleo de la memoria.
- P: ordenará según el empleo de la CPU.
- N: ordenará por ID.
- A: ordenará por antigüedad.

El programa top muestra además algunas estadísticas generales acerca del sistema:

- La hora actual y en la que se inició el sistema.
- La cantidad de usuarios conectados.
- Los promedios de carga de la CPU en los últimos uno, cinco y quince minutos transcurridos.
- Un resumen estadístico de la cantidad de procesos en ejecución y su estado (sleeping, running, zombie y stopped).

Un resumen del empleo de la memoria física y virtual (swap).

Los tres primeros aspectos se pueden ver también con el comando uptime y el último con free.

# TEMA 9: INTRODUCCIÓN A LINUX

## 9. PROCESOS (CONTROL)

El shell bash permite ejecutar los procesos en foreground (primer plano) o background (segundo plano). Los primeros son únicos por terminal (no puede haber más de un proceso en primer plano en cada terminal) y es la forma en que se ejecuta un proceso por defecto. Sólo se retorna al prompt una vez que el proceso termine, sea interrumpido o detenido. En cambio, en background pueden ejecutarse muchos procesos a la vez asociados a la misma terminal. Para indicar al shell que un proceso se ejecute en background, se utiliza la construcción:

<línea de comando> &

Ejemplo: # updatedb &

Para modificar el estado de un proceso en foreground desde bash existen dos combinaciones de teclas muy importantes que este interpreta:

1. Ctrl-c : trata de interrumpir el proceso en foreground (1º plano). Si es efectivo, el proceso finaliza su ejecución (se le mata).
2. Ctrl-z : trata de detener el proceso en foreground. Si es efectivo el proceso continúa activo aunque deja de acceder al procesador (está detenido y pasa a background ó 2º plano).



# TEMA 9: INTRODUCCIÓN A LINUX

## 9. PROCESOS (CONTROL)

Para ver los procesos detenidos o en background en un shell se emplea el comando integrado a bash `jobs`, que mostrará una lista con todos los procesos en dichos estados mediante los comandos asociados y un identificador numérico especial.

Ejemplo: `# jobs`

```
$ jobs -l
[1] 4229 Ejecutando      gedit &
[2]- 4239 Parado        sleep 50
[3]+ 4241 Parado        find / -name Descargas 2> /dev/null
$ jobs -p
4229
4239
4241
$ jobs -r
[1] Ejecutando      gedit &
$ jobs -s
[2]- Detenido      sleep 50
[3]+ Detenido      find / -name Descargas 2> /dev/null
```

Algunas opciones:

<b>-l</b>	Muestra el PID además de la información anterior.
<b>-p</b>	Solo muestra el PID de los trabajos
<b>-r</b>	Solo muestra los trabajos que están en estado de ejecución.
<b>-s</b>	Solo muestra los trabajos que están en estado de detenido.

# TEMA 9: INTRODUCCIÓN A LINUX

## 9. PROCESOS (CONTROL)

Los procesos detenidos se pueden llevar al background y estos a su vez pueden trasladarse al foreground. Para ello se emplean respectivamente los comandos integrados al bash: **bg** y **fg**, pasándoles como argumento el identificador especial del proceso. Si no se especifica un argumento se asumirá el trabajo marcado con un signo ``+" que sería el último detenido o llevado al background.

Ejemplos:

```
$ bg 2  
[2]- cp /var/log/messages /tmp &  
$ fg  
cp /var/log/messages /tmp
```

Si se comenzara a ejecutar un proceso y éste se demora mucho y no interesan por el momento sus resultados se puede detener y enviarlo al background haciendo Ctrl-z. Se puede volver a traerlo a 1º plano con fg.

# TEMA 9: INTRODUCCIÓN A LINUX

## 9. PROCESOS (COMUNICACIÓN)

El comando para interactuar con los procesos es kill.

Éste permite enviar señales con significados muy diversos.

Los programas o comandos deben estar preparados para atrapar y tratar estas señales, al menos las más importantes. Ejemplos:

\$ kill -l (un listado de todos los tipos de señales)

- |               |             |              |             |
|---------------|-------------|--------------|-------------|
| 1) SIGHUP     | 2) SIGINT   | 3) SIGQUIT   | 4) SIGILL   |
| 5) SIGTRAP    | 6) SIGABRT  | 7) SIGBUS    | 8) SIGFPE   |
| 9) SIGKILL    | 10) SIGUSR1 | 11) SIGSEGV  | 12) SIGUSR2 |
| 13) SIGPIPE   | 14) SIGALRM | 15) SIGTERM  | 17) SIGCHLD |
| 18) SIGCONT   | 19) SIGSTOP | 20) SIGTSTP  | 21) SIGTTIN |
| 22) SIGTTOU   | 23) SIGURG  | 24) SIGXCPU  | 25) SIGXFSZ |
| 26) SIGVTALRM | 27) SIGPROF | 28) SIGWINCH | 29) SIGIO   |

Sintaxis:

kill -l [señal]

kill [-s señal] <id>

kill [-señal] <id>

Por defecto kill envía la señal TERM que indica al proceso que debe terminar (15). La señal 9 o KILL lo finaliza forzosamente (es como una orden TERM pero imperativa). La señal HUP es interpretada por muchos comandos y programas como una indicación de que releen los ficheros de configuración correspondientes (que reinicien su ejecución).

# TEMA 9: INTRODUCCIÓN A LINUX

## 9. PROCESOS (COMUNICACIÓN)

Ejemplos:

\$ kill 1000	# envía la señal 15 (TERM) al proceso 1000
\$ kill -9 10101	# envía la señal 9 (KILL) al proceso 10101
\$ kill -4 18181	# envía la señal 4 (ILL) al proceso 18181
\$ kill -HUP 199	# envía la señal HUP al proceso 199
\$ kill %2	# envía la señal 15 (TERM) al trabajo 2 (en background o detenido)

También existe `killall` que permite enviar señales a los procesos a través de sus nombres.

A diferencia del ID, el nombre de un proceso no es único, o sea pueden existir muchos procesos con el mismo nombre y de ahí la utilidad de este comando.

Sintaxis: `killall [opciones] [-señal] <nombre>`

Algunas opciones:

- `-i` : forma interactiva. Pregunta para cada proceso si se desea enviar la señal o no.
- `-v` : reporta si fue exitoso el envío de la señal.

Ejemplo: `$ killall -9 ssh`

# TEMA 9: INTRODUCCIÓN A LINUX

## 9. PROCESOS (PRIORIDADES)

En Linux existe la posibilidad de iniciar los procesos con prioridades distintas a las asignadas por parte del sistema. Para ello se puede emplear el comando `nice`. Este al invocarse sin argumentos imprime la prioridad asignada por defecto a los procesos del usuario actual.

Ejemplos:

```
$ nice tar cvf /tmp/etc.tgz /etc
# incrementa en 10 la prioridad por defecto del comando
$ nice - 10 updatedb
# ejecuta un comando con prioridad 10
$ nice - -10 updatedb
# ejecuta un comando con prioridad -10
```

La otra forma de emplear a `nice` es indicando la nueva prioridad precedida del signo “-” y la línea de comando que desencadena el proceso. Si no se indicara la prioridad se incrementa en 10 la por defecto. Sólo el usuario `root` puede asignar a sus procesos prioridades con valores inferiores a cero.

# TEMA 9: INTRODUCCIÓN A LINUX

## 10. DISPOSITIVOS DE ALMACENAMIENTO. MONTAJE Y DESMONTAJE

Como ya hemos visto, los dispositivos físicos de la máquina en general y los de almacenamiento de información, en particular, son manipulados a través de ficheros especiales ubicados en el directorio /dev.

Los discos duros, las particiones de estos, las unidades de disquete y de CD-ROM son ejemplos de estos dispositivos con los cuales interactuamos constantemente.

Pero trabajar directamente sobre los dispositivos representados de esa forma casi nunca es conveniente ni resulta cómodo, por lo que usualmente se incorporan al sistema de ficheros tradicional. Esto es lo que conocemos como “montar” el dispositivo.

Las particiones de los discos, se montan en /, /home y /usr.

En el fichero fstab del directorio /etc se especifican donde y como se montan.

```
m3t4g4m3@nexolinux ~ $ cat /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options> <dump> <pass>
proc /proc proc nodev,noexec,nosuid 0 0
# / was on /dev/sda3 during installation
UUID=d094a83c-e9fb-4059-8782-c8d9ed79ce50 / ext4 errors=remount-ro 0 1
# /home was on /dev/sda6 during installation
UUID=a5392540-01e7-443d-b88f-fc43b899c031 /home ext4 defaults 0 2
# swap was on /dev/sda5 during installation
UUID=d50bf89e-b003-4768-ac1d-6a2936dff5ee none swap sw 0 0
/dev/sdb2 /media/SOFTWARE auto defaults 0 0
/dev/sdb1 /media/OCIO auto defaults 0 0
```

# TEMA 9: INTRODUCCIÓN A LINUX

## 10. DISPOSITIVOS DE ALMACENAMIENTO. MONTAJE Y DESMONTAJE

Cada línea en este fichero describe un dispositivo, indicando los siguientes aspectos para cada uno:

- ☐ Nombre del dispositivo o etiqueta.

Ejemplos: /dev/hda1, /dev/sdc1, /dev/fd0,  
LABEL=/home, LABEL=/cursos.

También puede aparecer codificada como UUID.

- ☐ Directorio donde se monta.

Ejemplos: /, /mnt/floppy, /tmp, etc.

- ☐ Sistema de ficheros.

Ejemplos: ext2, msdos, nfs, swap, iso9660, auto, etc.

- ☐ Opciones de montaje. Ejemplos: ro, rw, exec, auto, user, etc.

Dos valores numéricos: el primero toma los valores 0 ó 1 indicando si al dispositivo se le hará dump (especie de backup) o no. El segundo número expresa la prioridad que tiene el dispositivo cuando se chequea la integridad del File System durante el inicio del sistema.

# TEMA 9: INTRODUCCIÓN A LINUX

## 10. DISPOSITIVOS DE ALMACENAMIENTO. MONTAJE Y DESMONTAJE

Las opciones de montaje son numerosas. Hay bastantes más, pero nos quedaremos con un extracto:

- ☐ auto : indica que el dispositivo se monta siempre que se inicie el sistema. La opuesta es noauto.
- ☐ rw: indica que el dispositivo se monta con permisos de lectura y escritura.
- ☐ ro: indica que el dispositivo se monta con permisos de lectura solamente.
- ☐ owner: indica que el primer usuario distinto de root conectado al sistema localmente tiene derechos a montar y desmontar el dispositivo (se adueña de este).
- ☐ user: indica que cualquier usuario puede montar y solo el mismo usuario podrá desmontar el dispositivo. La opción opuesta es nouser.

Para ver o cambiar la etiqueta de un dispositivo se puede emplear el comando `e2label`.



# TEMA 9: INTRODUCCIÓN A LINUX

## 10. DISPOSITIVOS DE ALMACENAMIENTO. MONTAJE Y DESMONTAJE

Si queremos que un dispositivo se monte automáticamente cada vez que el sistema se inicie, basta con colocar una línea apropiada en el fichero `/etc/fstab`.

Si lo que queremos es montar o desmontar dispositivos directamente, podemos emplear los comandos `mount` y `umount` respectivamente.

Estos mantienen una lista de los dispositivos montados en el fichero `/etc/mtab`.

Sintaxis:

`mount [opciones] [dispositivo] [dir]`

`umount [opciones] <dir> | <dispositivo>`

Algunas opciones:

- ❑ `-a`: en el caso de `mount` monta todos los dispositivos que tienen la opción `auto` en el fichero `fstab`, y para `umount` desmonta todo lo que está en el fichero `/etc/mtab`.
- ❑ `-t <tipo>` : indica el tipo de file system a montar.
- ❑ `-o <opciones>` : especifica las opciones de montaje (separadas por comas).

# TEMA 9: INTRODUCCIÓN A LINUX

## 10. DISPOSITIVOS DE ALMACENAMIENTO. MONTAJE Y DESMONTAJE

Hoy en día la mayoría de las distribuciones cuentan con algún tipo de programa monitor en memoria que se encargará de montar automáticamente cualquier dispositivo externo que conectemos al sistema. El desmontaje de los dispositivos sin embargo debe ser manual.

Un comando muy útil en Linux es `df`. Este se emplea para conocer información acerca de las particiones y dispositivos montados actualmente en el sistema.

Para cada dispositivo se muestra por defecto su tamaño, el espacio empleado, que porcentaje esta empleado, así como el directorio donde se ha montado.

Sintaxis: `df [opciones] [directorio]`

Algunas opciones:

- ☐ `-h` : (human readable view) por defecto los tamaños se muestran en bytes y con esta opción se hace de forma más legible (Ej. G para gigabytes y M para megabytes).
- ☐ `-T` : muestra además el tipo de file system de cada dispositivo.
- ☐ `-i` : describe la utilización de los i-nodos de cada partición, en lugar del espacio.

# TEMA 9: INTRODUCCIÓN A LINUX

## 10. DISPOSITIVOS DE ALMACENAMIENTO. MONTAJE Y DESMONTAJE

Otros comandos útiles para el manejo de discos son:

1. `fdformat`

permite formatear un disquete a bajo nivel (sin ponerle un file system determinado).

2. `dd`:

permite duplicar ficheros o partes de estos ya sean regulares o especiales (hacer imágenes).

Ejemplo: `$ fdformat /dev/fd0`

Sintaxis: `dd [opciones]`

Las opciones son en forma de pares `<llave>=<valor>`. Algunas opciones:

- ☐ `if=<fichero>` : especifica el nombre del fichero de entrada o origen.
- ☐ `of=<fichero>` : indica el nombre del fichero de salida o destino.
- ☐ `bs=<n>` : especifica la cantidad de bytes leídos y copiados a la vez o tamaño de bloque. Por defecto es 512.
- ☐ `count=<n>` : indica la cantidad de bloques a copiar del origen al destino. Por defecto se copian todos los bloques presentes en el origen.

# TEMA 9: INTRODUCCIÓN A LINUX

## 10. DISPOSITIVOS DE ALMACENAMIENTO. MONTAJE Y DESMONTAJE

Ejemplos:

```
# dd if=/kernel-image of=/dev/fd0
```

```
# dd if=/dev/hda1 of=/mnt/floppy/boot_sector count=1 bs=512
```

```
# dd if=/dev/cdrom of=CDimage.iso
```

**eject** : desmonta, si es necesario, y luego expulsa un dispositivo a nivel de software.

**mkfs** : se utiliza para crear un sistema de ficheros en un dispositivo.

**fsck** : chequea y repara un sistema de ficheros.

Sirve de interfaz para otros comandos más específicos como `fsck.msdos`, `fsck.reiserfs`, `fsck.minix`, `fsck.ext2`, `e2fsck`, `dosfsck` y `reiserfsck`.

Ejemplos:

```
# fsck /dev/hdc5
```

```
# dosfsck /dev/fd0
```

# TEMA 9: INTRODUCCIÓN A LINUX

## 11. MANEJO DE PAQUETES

Los paquetes no son más que ficheros con cierto formato que, manipulados por un comando u otra aplicación son capaces de instalarse en el sistema de acuerdo a las especificaciones que determinó su fabricante.

Entre las especificaciones puede citarse a los ficheros y directorios que crea el paquete, el tamaño que ocupa una vez instalado, una descripción breve y otra más amplia de su utilidad, las dependencias que puede haber respecto a otros paquetes, etc.

En Linux no se utiliza un único sistema de paquetes, sino que podemos encontrar varias alternativas como son:

.rpm: (RedHat Package Manager) → Red Hat, SuSE, mandrake, fedora, mandriva, mageia, PCLinuxOS, openSUSE entre otros.

.deb: Formato de [paquetes de software](#) de la distribución de [Linux](#), [Debian GNU/Linux](#) y derivadas Ubuntu-

Como **Debian**, su nombre proviene de **Deborah** Murdock, exesposa del fundador de la distribución **Ian Murdock**.

# TEMA 9: INTRODUCCIÓN A LINUX

## 11. MANEJO DE PAQUETES

En Red Hat la herramienta por excelencia para administrar paquetes es el comando rpm que posee muchísimas opciones.

Es capaz de instalar, actualizar, desinstalar, verificar y solicitar programas.

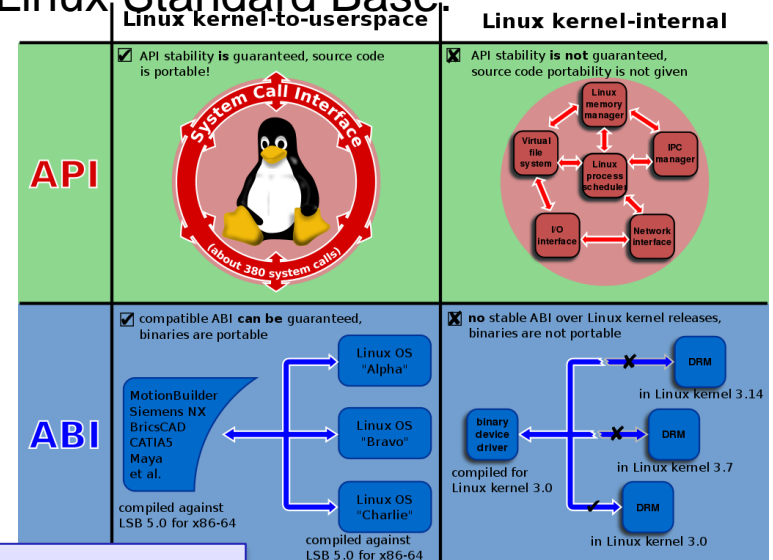
RPM es el formato de paquete de partida del Linux Standard Base.



Para el administrador de sistemas que realice mantenimiento y actualización de software, el uso de gestor de paquetes en vez de construirlos manualmente tiene ventajas como simplicidad, consistencia y la capacidad de que aquellos procesos se automaticen.

Ejemplos:

rpm -qa	→ muestra paquetes instalados.
rpm -qi foo	→ muestra la información de un paquete RPM.
rpm -ql foo	→ lista ficheros de un paquete RPM instalado.
rpm -qc foo	→ lista solo los ficheros de configuración.
rpm --checksig foo	→ verifica firma de un paquete RPM.
rpm -ivh "localFile.rpm"	→ instala un paquete.
rpm -e "localFile.rpm"	→ desinstala un paquete.



# TEMA 9: INTRODUCCIÓN A LINUX

## 11. MANEJO DE PAQUETES

En Red Hat la herramienta por excelencia para administrar paquetes es el comando rpm que posee muchísimas opciones.

Es capaz de instalar, actualizar, desinstalar, verificar y solicitar programas.

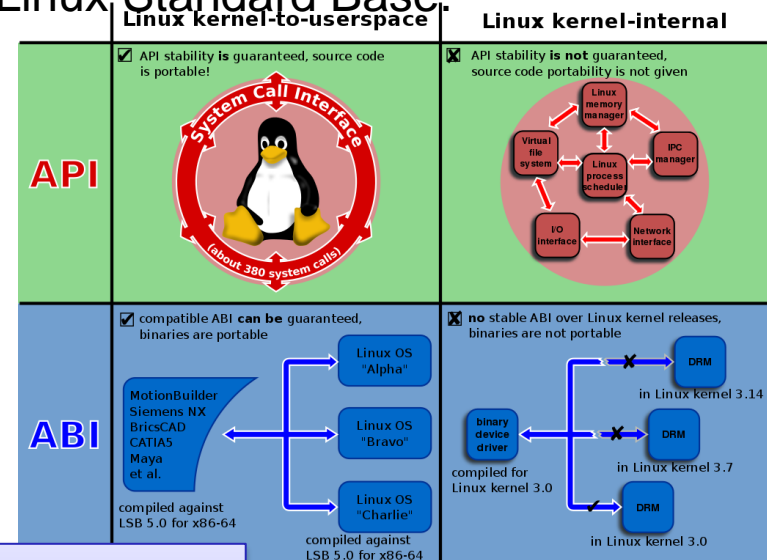
RPM es el formato de paquete de partida del Linux Standard Base.



Para el administrador de sistemas que realice mantenimiento y actualización de software, el uso de gestor de paquetes en vez de construirlos manualmente tiene ventajas como simplicidad, consistencia y la capacidad de que aquellos procesos se automaticen.

Ejemplos:

rpm -qa	→ muestra paquetes instalados.
rpm -qi foo	→ muestra la información de un paquete RPM.
rpm -ql foo	→ lista ficheros de un paquete RPM instalado.
rpm -qc foo	→ lista solo los ficheros de configuración.
rpm --checksig foo	→ verifica firma de un paquete RPM.
rpm -ivh "localFile.rpm"	→ instala un paquete.
rpm -e "localFile.rpm"	→ desinstala un paquete.



# TEMA 9: INTRODUCCIÓN A LINUX

## 11. MANEJO DE PAQUETES

En el principio existían los .tar.gz. Los usuarios tenían que descomprimir, destaar y compilar cada programa que quisieran usar en su sistema GNU/Linux.



Actualmente el sistemas de paquetes permite la instalación sin que el usuario se tenga que preocupar de las bibliotecas u otros programas necesarios.

Cuando Debian fue creado, se decidió que el sistema incluyera un programa que se encargara de manejar los paquetes (programas) instalados en el ordenador. Este programa se llamó DPKG. Así fue como nació el primer "paquete" en el mundo GNU/Linux, aún antes de que RedHat decidiera crear su propio sistema de paquetes "rpm".



Rápidamente llegó un nuevo dilema a las mentes de los creadores de GNU/Linux. Ellos necesitaban un modo fácil, rápido y eficiente de instalar programas, que manejara automáticamente las dependencias (programas que dependen de otros) y se hiciera cargo de la configuración mientras se actualizan. Nuevamente Debian fue pionera y creó el APT, Herramienta Avanzada de Empaquetamiento (Advanced Packaging Tool).



# TEMA 9: INTRODUCCIÓN A LINUX

## 11. MANEJO DE PAQUETES

Hay interfaces gráficas como Synaptic, PackageKit, Gdebi o en Ubuntu Software Center (este último solo para Ubuntu), desde la versión 3.0. Es posible convertir un paquete deb a otros formatos de paquete (como RPM), y viceversa, usando la aplicación alien.

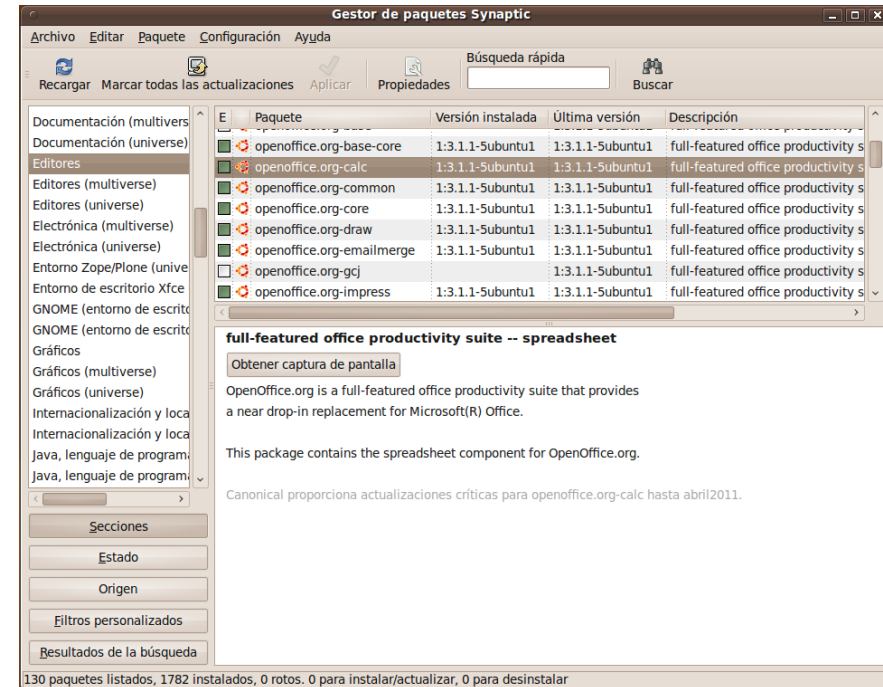
[Más información.](#)

### Estructura interna

Los paquetes deb son archivos ar estándar de Unix que incluyen dos archivos tar en formato gzip, bzip2 o lzma: uno de los cuales alberga la información de control y el otro los datos.

Estos paquetes contienen tres archivos:

- ✓ debian-binary - número de versión del formato deb. Este es "2.0" para las versiones actuales de Debian.
- ✓ control.tar.gz - toda la meta-información del paquete.
- ✓ data.tar, data.tar.gz, data.tar.bz2 o data.tar.lzma: - los archivos a instalar.



# TEMA 9: INTRODUCCIÓN A LINUX

## 11. MANEJO DE PAQUETES

### Manejo de paquetes

Desde el terminal con el comando dpkg

Para instalar paquetes:

```
dpkg -i paquete.deb
```

Para verificar que se ha instalado correctamente:

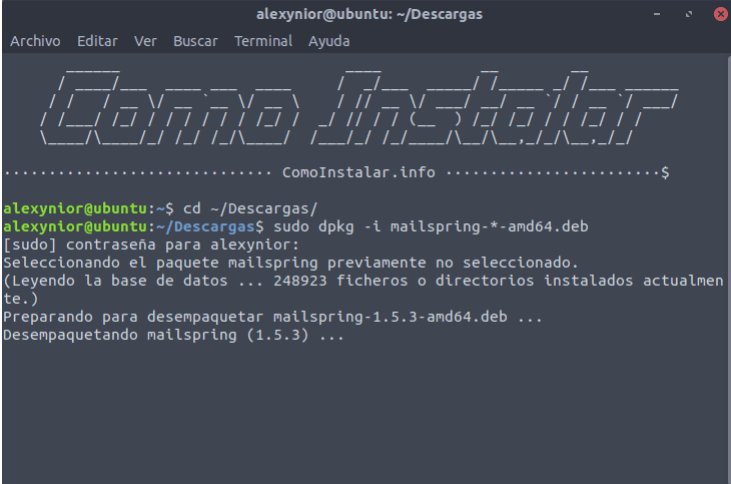
```
dpkg -l | grep 'paquete'
```

Para eliminar la instalación:

```
dpkg -r paquete.deb
```

Para eliminar y borrar todos los datos del programa: -P (purge)

```
dpkg -P paquete.deb
```



```
alexynior@ubuntu: ~/Descargas
Archivo  Editar  Ver  Buscar  Terminal  Ayuda

..... ComoInstalar.info .....$

alexynior@ubuntu:~$ cd ~/Descargas/
alexynior@ubuntu:~/Descargas$ sudo dpkg -i mailspring-*-amd64.deb
[sudo] contraseña para alexynior:
Seleccionando el paquete mailspring previamente no seleccionado.
(Leyendo la base de datos ... 248923 ficheros o directorios instalados actualmen
te.)
Preparando para desempaquetar mailspring-1.5.3-amd64.deb ...
Desempaquetando mailspring (1.5.3) ...
```

# TEMA 9: INTRODUCCIÓN A LINUX

## 11. MANEJO DE PAQUETES

### Repositorios

Es un servidor accesible mediante internet que almacena paquetes y programas para que nosotros los podamos descargar e instalar en nuestra distribución GNU-Linux.

Las diferentes distribuciones GNU-Linux disponen de sus propios repositorios en los que se hallan los programas que nosotros podemos instalar en nuestro equipo.

Además de los repositorios propios de cada una de las distribuciones, también se pueden añadir y usar repositorios de terceros que contendrán versiones más actuales del software que tenemos instalado o programas que no han incluido los creadores de la distro que usamos.

Ya hemos visto algunos gestores de paquetes, como por ejemplo rpm, apt. Existen otros como YaST o Pacman.

Los gestores son las herramientas que usaremos para descargar e instalar el software de un repositorio.

# TEMA 9: INTRODUCCIÓN A LINUX

## 11. MANEJO DE PAQUETES

### Repositorios

Si queremos instalar el reproductor de vídeo VLC ejecutaremos el siguiente comando:

```
sudo apt-get install vlc
```

En el momento de ejecutar el comando sucederá lo siguiente:

1. Con el gestor de paquetes apt nos conectaremos al repositorio de internet que contiene los paquetes que queremos descargar. Antes de empezar la descarga, mediante un par de claves asimétricas, un sistema de firmas y una función hash se comprobará que los paquetes a descargar provienen de un repositorio seguro y no han sido modificados por nadie.
2. Una vez realizada la comprobación se descargarán los paquetes y dependencias necesarias para instalar VLC.
3. Una vez descargados los paquetes se procederá a la instalación de los mismos.



# TEMA 9: INTRODUCCIÓN A LINUX

## 11. MANEJO DE PAQUETES

### Repositorios

Para disfrutar de la resolución automática de dependencias, es necesario detallar correctamente el origen de los paquetes. Éstos viene determinados en el fichero `/etc/apt/sources.list`

Y algunos casos en `/etc/apt/sources.list.d/`

Cada línea del archivo determina un repositorio.

Las principales herramientas de paquetes .deb son:

- `dpkg`
- `apt-get`
- `apt`
- `aptitude`

Busqueda de paquetes: `sudo apt-cache search ftp`

# TEMA 9: INTRODUCCIÓN A LINUX

## 11. MANEJO DE PAQUETES

### Repositorios

Los repositorios comprometen un alto grado de seguridad, ya que el software incluido se prueba exhaustivamente y se compila para que sea compatible con una distribución y versión particular. Por lo tanto, se espera que las actualizaciones ocurran sin «bugs» inesperados.

En general, agregar un repositorio no estándar es un paso simple. El comando `sudo apt-add-repository` en Ubuntu, por ejemplo, se puede usar para agregar un repositorio.

La opción `–help` para el comando `apt-add-repository` muestra estos ejemplos de comando:

```
apt-add-repository http://extras.ubuntu.com/ubuntu
```

En RedHat, Fedora y sistemas similares, deberías usar un comando como el que se muestra a continuación para ver los repositorios que usan los comandos de actualización. Ten en cuenta que estamos utilizando el comando `dnf` en este ejemplo. Este es el reemplazo del antiguo comando `yum`.

```
$ sudo dnf repolist
Last metadata expiration check: 0:18:37 ago on Sat 15 Sep 2018 12:28:02 PM EDT.
repo id      repo name          status
*fedora      Fedora 28 - x86_64 57,327
*updates     Fedora 28 - x86_64 - Updates 18,739
```

# TEMA 9: INTRODUCCIÓN A LINUX

## 11. MANEJO DE PAQUETES

### Repositorios

#### Eliminación de paquetes

Desinstala un paquete, sólo ese paquete, pero no la totalidad de dependencias y los archivos de configuración del programa. En este caso por ejemplo en terminal sería:

```
# apt-get remove vlc
```

Cuando queramos desinstalar una aplicación el comando más apropiado es este. Desinstala completamente una aplicación, y se eliminarán también las dependencias que no son requeridas por otros paquetes y los archivos de configuración; no dejara elementos no útiles en el sistema que sólo ocupan espacio. Por ejemplo:

```
# apt-get remove --purge vlc
```

#### Para limpiar el sistema:

Elimina paquetes que se instalaron automáticamente para satisfacer las dependencias de otros paquetes pero que ya no son necesarios; o, también, cuando desinstalamos una aplicación y no se desinstalan todos las dependencias que no son necesarias para otros paquetes.

Se puede usar el comando:

```
# apt-get autoremove
```

# TEMA 9: INTRODUCCIÓN A LINUX

## 11. MANEJO DE PAQUETES

### Repositorios

#### Cache

Limpia cache de paquetes descargados e instaladas, no desinstala ningún paquete del sistema.

```
# apt-get clean
```

Para saber cuánto espacio están ocupando estos archivos en consola ejecutamos:

```
# du -sh /var/cache/apt/archives
```

Elimina de la cache sólo aquellos paquetes que ya no se pueden descargar o que son inservibles, en cualquier caso inútiles.

Actualización:

`apt-get upgrade; apt-get update`

```
# apt-get autoclean
```

La principal diferencia entre update y upgrade es que update es hacer y mantener algo actualizado o una versión mejor de la versión anterior, a menudo para resolver bugs o problemas. Sin embargo, upgrade es elevar algo a un nivel más alto agregando o reemplazando algunos componentes.





# TEMA 9: INTRODUCCIÓN A LINUX

## 11. MANEJO DE PAQUETES

### Repositorios

Inspección de paquetes

`dpkg -I nombre_del_paquete` → Muestra el estado del paquete

`dpkg -S nombre_del_archivo` → Busca el paquete que instalo el archivo

`dpkg -L nombre_del_paquete` → Lista los archivos instalados por el paquete

`dpkg --contens paquete.deb` → Lista el contenido del paquete especificado

`apt cache show nombre_del_programa` → Muestra descripción y los detalles sobre el paquete del programa especificado.

Firma de paquetes

Para garantizar la autenticidad de cada paquete, es posible verificar su firma.

Por ejemplo en Fedora, las claves se incorporan con:

`rpm --import /usr/share/rhn/RPM-GPG-KEY-FEDORA`

La verificación se hace con:

`rpm --checksig nombre del paquete`

La integridad se puede comprobar

con la opción `-v`

Otras opciones

Cómo verificar la firma PGP del software descargado en Linux

julio 10, 2021 52 minutos de lectura

```
tecint@ubuntu:~/Downloads$  
tecint@ubuntu:~/Downloads$ gpg --verify tixatl_2.84-1_amd64.deb.asc tixatl_2  
gpg: Signature made Thu 24 Jun 2021 07:44:37 PM EAT  
gpg: using RSA key 9DEASE350F9D285E46D3B7E3CE737F191AF5DCFB  
gpg: Good signature from "Tixatl Software Inc. (KH) <support@tixatl.com>" [un]  
gpg: WARNING: This key is not certified with a trusted signature!  
gpg: There is no indication that the signature belongs to the owner.  
Primary key fingerprint: 9DEA 5E35 0F9D 285E 46D3 B7E3 CE73 7F19 1AF5 DCFB  
tecint@ubuntu:~/Downloads$  
tecint@ubuntu:~/Downloads$  
tecint@ubuntu:~/Downloads$  
tecint@ubuntu:~/Downloads$ gpg --show-keys tixatl.key  
pub rsa2048 2015-10-04 [SC]  
9DEASE350F9D285E46D3B7E3CE737F191AF5DCFB  
uid Tixatl Software Inc. (KH) <support@tixatl.com>  
sub rsa2048 2015-10-04 [E]
```

Noticias destacadas



Te has perdido...



