

Administración General Linux: Conceptos de instalación y configuración inicial



Ing.Luis Vazquez*

Índice

1. Pasos básicos en una instalación típica de un sistema Linux	2
2. Booteo de una máquina Linux.	2
3. Particiones	2
3.1. Tipos de particiones	2
3.2. Nombres de dispositivos para particiones en Linux	3
3.3. Creando particiones con <code>fdisk</code> o <code>cfdisk</code>	3
3.4. Inicializando el <i>swap</i> y creando los filesystems	4
3.4.1. Tipos de filesystems	4
3.4.2. Reparación de filesystems (<code>fsck</code>)	5
4. Dispositivos	5
5. Como apagar un sistema Linux por consola	6
6. Como crear un disco de booteo	6
6.1. Recuperar password de root	6
7. Uso del LILO	7
8. Secuencia de arranque de un sistema Linux	8
9. Single-user mode	10

*luis.vazquez@imago.com.uy

1. Pasos básicos en una instalación típica de un sistema Linux

1. Bootear un sistema Linux de instalación (desde un floppy o cdrom).
2. Crear las particiones con `fdisk` o `cfdisk`
3. Crear los el *swap* y los *filesystems* en las particiones con `mkfs` y `mkswap`.
4. Montar las particiones en los directorios y en el orden adecuados.
5. Instalar el software *Linux* y configurarlo.
6. Instalar un *boot-loader* (LILO o GRUB) o crear un diskette de *booteo*.

2. Booteo de una máquina Linux.

Lo primero que tenemos que lograr es que el *kernel* del sistema sea cargado en memoria y comience a ser ejecutado. Puede ser con un diskette o cdrom de booteo, o con un *kernel* Linux instalado en el disco duro. Debemos usar un *boot-loader* que se instala en el MBR (*Master Boot Record*) o sector de booteo de un disco (o partición) y se ocupa de cargar el núcleo.

En generar se usa LILO (*Linux Loader*). Presenta un prompt (a veces es necesario Shift o Ctrl) desde el cual podemos pasarle parámetros al kernel.

```
boot: linux root=/dev/hdb1 mem=64M ro ramdisk=4096
```

Los parámetros de uso más común son:

root Nos permite especificarle al kernel que partición montar inicialmente como raíz (/)

mem Podemos especificar el tamaño real de memoria RAM en el sistema en caso que no lo detecte correctamente (pasa en algunos BIOS).

ro Montar (inicialmente) la partición raíz en modo ReadOnly.

ramdisk Le decimos que cree un bloque de *ramdisk* en la memoria RAM y le indicamos el tamaño. Este bloque puede ser usado luego por el sistema como un pequeño *filesystem*.

3. Particiones

3.1. Tipos de particiones

Las particiones en un disco duro se dividen en *primarias* o físicas (numeradas del 1 al 4) y las particiones *lógicas* (del 5 en adelante) que se crean como subdivisiones dentro de otra partición (a la que se suele llamar *extendida*).

Como regla práctica si uno necesita menos de 4 particiones crea particiones primarias, en tanto si va a necesitar mas de 4, debe crear 3 particiones primarias y luego continúa con particiones lógicas.

3.2. Nombres de dispositivos para particiones en Linux

En Linux los discos y sus particiones se acceden a través de archivos de dispositivo ubicados bajo el directorio `/dev` (estos archivos NUNCA deben ser accedidos directamente por el usuario)

- Primer Floppy: `/dev/fd0` (A:)
- Primer disco duro IDE (master en controlador 1): `/dev/hda`
- Primer partición primaria: `/dev/hda1`
- Primer partición lógica: `/dev/hda5`
- Slave en controlador 1: `/dev/hdb`
- Master en controlador 2: `/dev/hdc`
- Primer disco SCSI: `/dev/sda`
- Primer CD-ROM SCSI: `dev/scd0`
- etc....

3.3. Creando particiones con `fdisk` o `cdisk`

La rutina básica para crear particiones en Linux es `fdisk`. Para usarla se ejecuta:

```
# fdisk /dev/hda
```

Una vez dentro las opciones más importantes son:

```
m=help, q= quit, p=mostrar tabla, n=nueva partición, w=write to
disk(ojo!!)
```

Un programa equivalente pero más amigable (en base a menús) es:

```
# cfdisk /dev/hda
```

Al hacer las particiones estas se pueden especificar indicando los cilindros donde comienza y termina, o los tamaños en mega-bytes (M) o en bloques. Un *block* en Linux son 1024 bytes! Aun en el caso de poner un tamaño, `fdisk` alinea el fin de la partición al cilindro mas próximo.

Se debe cambiar el tipo de partición a 82 para el *swap* (en `fdisk`: L=lista, t=cambiar tipo)

Antes de crear un *filesystem* en un disco floppy es necesario hacerle un formato de bajo nivel. Esto se hace con `fdformat`.

```
# fdformat /dev/fd0
```

3.4. Inicializando el *swap* y creando los filesystems

Los comandos para dar formato y activar la partición son `mkswap` y `swapon`

```
# mkswap -c /dev/hda2 nroblocks
```

El parámetro *nroblocks* por lo general es opcional, y en caso de necesitarlo se puede sacar de `fdisk`. La opción `-c` es para que nos avise de posibles bloques con errores. Es más lento pero más seguro, en especial en un disco usado!!

Luego debemos activar la partición:

```
# swapon /dev/hda2
```

3.4.1. Tipos de filesystems

Crear el filesystem es crear la estructura lógica que es capaz de almacenar los archivos y directorios del sistema. Es análogo (aunque no igual) al "formato de alto nivel" en DOS. Los tipos de sistemas de archivo más usados en Linux son:

```
ext2, ext3, reiserfs, xfs, vfat, iso9660, minix, nfs, smbfs, proc
```

Independientemente del tipo de *filesystem*, Linux presenta una interfaz común: el árbol de directorios y archivos. Las particiones se *montan* como directorios en esa estructura de árbol usando el comando *mount*. Luego de arrancar el sistema ejecuta

```
mount -a
```

lo que hace que se monten todos los *filesystems* especificados en `/etc/fstab` y con las opciones allí especificadas.

Para crear un *filesystem* en una partición se usa `mkfs`. Este programa es en realidad un *front-end* para todas las rutinas específicas de cada tipo de filesystem:

```
mkfs.ext2, mkfs.ext3, mkfs.vfat, mkfs.minix, mkfs.msdos, mkfs.xfs
```

Por ejemplo para crear un filesystem de tipo *ext2* en un floppy nuevo se puede usar:

```
# mkfs -c -v -t ext2 /dev/fd0 1440
```

La opción `-t fstype` permite especificar el tipo de *filesystem* a crear. Por ejemplo el comando anterior es equivalente a `mkfs.ext2`.

En realidad en el caso de *ext2* el comando `mkfs.ext2` es un alias de la rutina especializada `mke2fs`.

El formato *ext3* es en realidad una extensión al formato *ext2* que introduce un *journal*, que generalmente hablando es un bloque que se reserva en el sistema de archivos para introducir redundancia de modo de hacer al sistema más robusto ante caídas. Por lo tanto el comando `mke2fs` con la opción `-j` es equivalente a `mkfs.ext3`. Así

```
# mke2fs -c -j /dev/hda5
```

crea un *ext3* en la partición `/dev/hda5`

Atención: Nunca usar estos comandos en `/dev/hda!!!!`

Para el tipo *reiserfs* se usan los comandos `mkreiserfs`, `reiserfsck`.

3.4.2. Reparación de filesystems (fsck)

El programa `fsck` se usa para detectar y corregir problemas en los filesystems. Igual que `mkfs` es un *front-end* para los programas específicos de cada tipo. Por ejemplo para chequear una partición con *ext2*

```
# fsck -t ext2 /dev/hda2
```

No es buena idea hacerlo con particiones montadas `rw`. En el caso de la partición raíz (`/`) se puede pasar a *Modo Single User* y entonces hacer

```
# mount -o remount,ro /
```

antes de ejecutar `fsck`.

Opciones importantes:

- a confirmar automáticamente. Usar con cuidado!
- c chequear bad blocks.
- v verbose, despliega más información.
- r reparar pidiendo confirmación (Modo interactivo).
- y asume respuesta de *yes* siempre (se usa si no bootea, pero se corre un riesgo).
- f : forzar el chequeo aunque todo parezca en orden (ver opción -b)

Para el caso de *ext2* (y *ext3*) existe el comando más específico `e2fsck` que es interactivo por defecto.

En los *filesystem* de tipo *ext2* se usa una estructura llamada *superblock* para guardar la información sobre la estructura del sistema de archivos. Como respaldo se mantienen copias de este *superblock* en determinadas posiciones dentro de la partición. El siguiente comando puede ser útil para recuperar un *filesystem ext2* cuyo superbloque primario ha sido corrompido:

```
e2fsck -c -f -b 8193
```

Le decimos que trate de buscar y usar un *superblock* de *backup* ubicado en el bloque 8193 de la partición para reparar el filesystem (ver `man e2fsck`).

4. Dispositivos

Los archivos especiales ubicados bajo `/dev` se conocen como *device files* y permiten el acceso directo a los diferentes dispositivos de hardware. Son accedidos y usados por los *drivers* del sistema. Estos archivos son diferentes a los archivos comunes del sistema.

Se pueden crear con

```
mknod -m permisos nombre type mayor menor
```

Por ejemplo

```
mknod -m 660 /dev/audio0 c 14 4
```

El tipo del *device file* puede ser **c** (*char device*) o **b** (*block device*). Los números *mayor menor* permiten al *kernel* asociar ese dispositivo con el *driver* adecuado.

En general las distribuciones Linux ya los crean por nosotros al instalarse. Se esta empezando a usar otro enfoque denominado *devfs* en el cual los dispositivos son creados automáticamente por pedido del *kernel*.

5. Como apagar un sistema Linux por consola

El comando para dar de baja o reiniciar un sistema es **shutdown**. Por ejemplo para apagar la maquina hacemos:

```
# shutdown -h now
```

y si queremos reiniciar dentro de 3 minutos ejecutamos

```
# shutdown -r +3 "Re-booteando para lanzar nuevo kernel"
```

La opción **-h** indica *halt* y **-r** indica *reboot*.

En los sistemas Linux modernos esto se hace mas rápido con **init 0** para apagar y con **init 6** para reiniciar. La combinación **Ctrl-Alt-Del** es lo mismo que **init 6** y eso se configura en */etc/inittab*.

6. Como crear un disco de booteo

Esto se puede hacer simplemente copiando un archivo *kernel* a un floppy, por ejemplo usando **dd** (debe caber).

```
# dd if=/boot/vmlinuz of=/dev/fd0 bs=8192
```

Al bootear antes que nada intenta montar la partición **/** que tiene especificada por defecto dentro del *kernel*, por lo tanto este *kernel* debe incluir al menos los *drivers* para el tipo de *filesystem* que sea.

La partición raíz por defecto se puede cambiar (así como otros parámetros) con **rdev**:

```
# rdev /boot/vmlinuz /dev/hda3
```

También se puede tener la partición raíz en un floppy (pasar **/dev/fd0** en **rdev**) y usar esto para intentar recuperar un sistema en el cual no podemos montar la partición raíz en el disco duro o hemos perdido el *password* de *root*. Para esto se usa también otra configuración en la cual el directorio raíz se crea en un *filesystem* en memoria RAM denominado *ramdisk*.

6.1. Recuperar password de root

- Bootear desde floppy
- **mount -t ext2 /dev/hda /mnt**
- **cd /mnt/etc**
- **vi shadow** y borrar el password de *root*.
- Rebootear y entrar como *root* sin password.

7. Uso del LILO

LILO es un *boot-loader* que se puede cargar en el MBR del disco o en una partición primaria (como *boot-loader* secundario) y permite lanzar Linux así como otros sistemas.

Si se instala en el MBR es el primer programa en correr!!

Si lo instalo como secundario \Leftarrow en el *boot record* de la partición (primaria o extendida) que contiene la partición raíz (/). Otro *boot-loader* deberá llamar a este LILO.

Solo se puede instalar LILO en el MBR de un disco o en una particion primaria (o extendida), pero no en una partición lógica.

En dual boot lo más fácil es pasarle por arriba al loader de MS-Win y ponerlo en el MBR.

El comportamiento de LILO, así como las distintas opciones de *boot*, se configuran en el archivo `/etc/lilo.conf`

Ejemplo de `lilo.conf` para booteo primario *dual-boot* con Windows:

```
boot = /dev/hda ## instalar en MBR de hda
## estas 2 lineas siempre así
map=/boot/map      ## nombre del mapa del sistema creado por LIL
## archivo contiene nuevo boot-loader a usar
## en general es un link a boot-text.b o boot-menu.b
install=/boot/boot.b
default=linux ## Cual es la opcin a bootear por defecto (ver label)
prompt ## Indica que muestre boot: antes de arrancar
timeout=50 ## Tiempo a esperar en el prompt
vga=mode ## modo vga a usar por defecto
password=MiClave ## password a usar con opcion restricted
image=/boot/vmlinuz ## kernel a bootear en esta opcion
label=linux      ## nombre de la opcion
root=/dev/hda5  ## partición a usar como raíz
restricted ## indica que no puedo pasar parámetros al kernel
read-only ## montar / read-only
vga=0x0F07 ## codigo de modo vga para esta imagen
append= 'mem=512M' # Permite pasar parámetros al kernel!!
other=/dev/hda1 ## opción que no es una imagen sino otro sector de booteo
label=win       ## nombre de esta opcion
table=/dev/hda ## donde se encuentra la tabla de partición
other=/dev/hda4 ## partición con otro lilo en este caso
label=debian   ## nombre de esta opcion
table=/dev/hda
```

Si una de las entradas tiene la opción `restricted`, no podré ingresar parámetros al bootear a menos que también exista una opción `password`, en cuyo caso deberé ingresarlo.

Ejemplo de `lilo.conf` para booteo encadenado:

```
lba32  ## Genera direcciones logicas de block de 32-bit
boot = /dev/hda4
map=/boot/map
install=/boot/boot.b
```

```
vga=normal
root=/dev/hda7
image=vmlinuz
label=Debian
read-only
```

Algunas opciones solo pueden especificarse globalmente

```
lba32, boot, map, install, default, prompt, timeout
```

en tanto otras se pueden especificar para cada imagen

```
vga, read-only, root
```

Una vez que tenemos listo `lilo.conf` el programa

```
# lilo
```

se encarga de leer la configuración y de instalar LILO en el sistema. Se debe estar seguro antes de ejecutarlo o la máquina puede no poder volver a bootear por si misma (ver `man lilo.conf`).

8. Secuencia de arranque de un sistema Linux

El primer programa que se ejecuta al bootear un sistema Linux es `init`. Este programa se encuentra en `/sbin/init` y usa `/etc/inittab` como archivo de configuración por lo que la partición raíz (`/`) debe poder ser montada por el kernel al comenzar a bootear.

La función principal de `init` es iniciar procesos según lo especificado en el archivo `/etc/inittab`.

Las líneas en el archivo `/etc/inittab` se componen de 4 campos separados por `:` (dos puntos)

```
label:runlevel:acción:comando
```

Lo primero que se ejecuta es el comando especificado con la acción `sysinit`. Este comando generalmente es un *script* encargado de la inicialización básica del sistema, incluyendo la inicialización de los dispositivos y la construcción del árbol de directorios del sistema, montando los filesystems de acuerdo a lo indicado en `/etc/fstab`. Por lo general este *script* se encuentra en `/etc/rc.d/rc.sysinit` (*RedHat, Mandrake*) o en `/etc/init.d/rcS` (*Debian*).

Luego se determina el *runlevel* a correr por defecto, especificado en `/etc/inittab` con una línea de la forma

```
id:3:initdefault:
```

Finalmente se ejecutan en orden todas las entradas de `/etc/inittab` que tengan especificado este *runlevel*. La más importante (y por lo general la primera) es una línea de la forma

```
l3:3:wait:/etc/rc.d/rc 3
```

Habr  una l nea de esta forma por cada *runlevel*. Especifica que se ejecute el *script* `/etc/rc.d/rc` pas ndole el n mero del *runlevel* como par metro.

El *script* `/etc/rc.d/rc` acepta un n mero de *runlevel* y llevar el sistema al estado apropiado para ese nivel. Para hacer esto se fija en los *scripts* presentes en el directorio `/etc/rc[i].d/`, siendo *i* el n mero del *runlevel*. Por ejemplo en `/etc/rc3.d` podemos encontrar:

```
K01gdm K11anacron ... S10sysklogd S11klogd S15bind9 ...
```

Los nombres de los *scripts* en ese directorio deben comenzar con la letra **K** o **S** seguida por un n mero de 2 d gitos y deben reconocer como m nimo las opciones **start** y **stop**. Ser n ejecutarlos con la siguiente convenci n:

- Si el nombre comienza con **K** se ejecuta con par metro **stop**. El servicio ser  detenido.
- Si el nombre comienza con **S** se ejecuta con par metro **start**. El servicio ser  arrancado.
- Primero se ejecutan todos los **K** y luego los **S** en orden alfab tico (que queda determinado por el n mero).

En realidad para no repetir muchas veces los mismos *scripts* lo que tenemos en los directorios `/etc/rc[i].d/` son links simb licos a los verdaderos *scripts* de los servicios disponibles en la maquina que est n ubicados en `/etc/init.d/`. A modo de ejemplo (en un *Debian*) tenemos

```
anacron bind9 gdm iptables klogd sysklogd networking xfs etc.
```

Los nombres de los servicios puede cambiar entre diferentes distribuciones (por ejemplo el servicio DNS se llama **bind9** en *Debian* y **named** en *RedHat*, *Mandrake* o *SuSe*). Uno puede arrancar o detener un servicio a mano simplemente ejecutando el archivo de ese servicio seguido de la opci n **start** o **stop**. Por ejemplo

```
/etc/init.d/iptables start
```

Tambi n se puede cambiar de *runlevel* desde la consola (con usuario *root*) usando el comando **init** o **telinit** seguido de un n mero, por ej:

```
telinit 5
```

La convenci n es que niveles del 2 al 5 son para modo multi-usuario y su uso var a entre las distribuciones, aunque por lo general el nivel 3 corresponde al sistema completo en modo consola y el nivel 5 al modo con servidor gr fico **X**.

Como vimos antes el *runlevel* 0 es para detener la m quina (*halt*), el 6 es para reiniciar (*reboot*); en tanto el *runlevel* 1 hace que la m quina entre en modo *single user*.

Otras entradas importantes en `/etc/inittab` son de la forma

```
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
...
```

donde se especifica que se arranquen los manejadores de las consolas virtuales en todos los niveles normales de usuario (ver `man mingetty`). También debe notarse la acción `respawn` que indica que estos procesos sean relanzados por `init` cada vez que mueren (esto hace que cada vez que se termina una sesión la consola vuelva a presentar el `login:`).

9. Single-user mode

Cuando la maquina corre en modo *single-user* no se levanta ningún servicio, tampoco hay conexión de red y solo se lanza un *shell* del usuario *root* en la consola. Este modo permite realizar tareas administrativas o de recuperación de problemas en el servidor.

En este modo el ÚNICO USUARIO es *root*.

Para entrar al modo *single-user* se puede hacer desde el *prompt* al bootear la máquina

```
boot: linux single
```

o de lo contrario entrando en el *runlevel* 1 con `telinit 1` (o `init S`).

En algunos sistemas entrar en modo *single-user* al bootear la máquina es una manera sencilla de entrar como *root* y *sin password* al sistema, por lo que muchas veces por seguridad se debe proteger con un password esta posibilidad. Esto se puede hacer usando `lilo` o el archivo `/etc/securetty` que lista los dispositivos considerados seguros, incluida la consola (ver `man securetty`).