3. Arranque y Parada

3.1. Introduccion

Introducción

El proceso por el cual un computador arranca y pone a disposición de los usuarios el sistema operativo es conocido como "booting" (bootstrapping). Es un periodo de especial vulnerabilidad, donde cualquier error de configuración puede hacer que el computador no funcione, además requiere conocimientos sobre algunos aspectos del sistema, como el sistema de ficheros, los demonios, la ejecución de macros, o la configuración del kernel, siendo además muy dependiente del hardware que esté debajo, lo que lo hace especialmente dependiente de la máquina en si y poco general.

A pesar de todo, se siguen unos pasos comunes que se pueden resumir de la siguiente manera:

1. Carga e inicialización del núcleo.
2. Detección y configuración de dispositivos.
3. Creación de procesos espontáneos (se llaman procesos espontáneos a los que no han sido creados por otros procesos a través de la llamada fork. Estos procesos varían en nombre y número en diferentes sistemas, pero todos tienen en común al proceso init encargado de la carga del núcleo).
4. Posible operación del administrador en modo monousuario.
5. Ejecución de las macros de arranque.
6. Operación normal multiusuario.

Estas fases, además, pueden ser realizadas de forma automática o manual. La usual es la primera, donde a partir de un dispositivo como puede ser el disco o algunas veces la cinta, un pequeño programa residente en alguna zona especial de estos dispositivos se encarga de leer el código del núcleo y volcarlo a memoria. La segunda se utiliza cuando hay algún problema en el arranque del computador, en cada paso el sistema se detiene para devolver el control al operador y normalmente se ejecuta la inicialización en modo monousuario.

En el caso de un sistema personal el proceso comienza con el POST (Power On Self Test) que chequea el sistema y pasa el control a la BIOS, la cual habrá configurado el usuario para que cargue el sistema operativo desde algún lugar. Esto, que en principio es versatilidad, puede ser una fuente de inseguridad ya que alguien (intruso) podría insertar un disquete para montar otro sistema. Podemos evitar esto botando directamente desde disco y haciendo que la BIOS arranque con una clave.

Una vez localizado el dispositivo, se ejecutará el programa de carga del sector de arranque (ver siguiente sección) que sabrá como cargar el sistema operativo.

3.2. Cargador LILO

El cargador

Como ya hemos mencionado, esta parte del sistema es independiente del Linux/UNIX y variará fuertemente de un sistema operativo a otro, por eso, en este apartado, nos referiremos de forma fundamental al cargador más habitual en el sistema Linux (puede servir para otros sistemas operativos) que es el LILO (LInux LOoader). Este es el nombre que recibe para la arquitectura más normal que es la INTEL, pero también existe en otras como la Alpha de Digital (MILO) o la SPARC de Sun (SILO). Existen otros gestores de arranque para Linux menos utilizados como:

- GRUB, GRand Unified Bootloader, muy potente y flexible (ver arriba referencias).
- Loadlin, para cargar Linux desde MSDOS, incluido en el directorio `/dosutils`.
- SYSLinux, suele ser utilizado por distribuciones de disquetes o CD instalables.
- NTLDR, para Windows NT.
- Poof, Para iMac.
- Otros programas comerciales como System Comander, Boot Manager, BootMagic o BootStar.
- O incluso sin disco a través de la red (si nos lo permite la BIOS [la BIOS es el sistema básico de entrada/salida, es un chip situado en la placa madre de los PC que da una interfaz para el manejo de dispositivos como el disco, el teclado o la pantalla, LINUX no utiliza la BIOS, ya que por eficiencia y para evitar las limitaciones de la misma trabaja directamente con los dispositivos, pero el LILO si lo hace]).

Si hemos optado por LILO, podremos colocarlo de diversas maneras: en el MBR hacia donde salta la BIOS automáticamente, en el primer sector del disco, típicamente accesible por el programa cargador del MBR (o IPL), o en un disquete, si las BIOS tienen la secuencia A: C:. Todo dependerá de la situación en que estemos, si sólo tenemos Linux en nuestro sistema lo lógico será instalarlo en el MBR, si el sistema es dual y nos interesa tener el cargador de otro sistema deberemos instalarlo en la partición raíz o en el disquete.

El comando de instalación de LILO se encuentra en el fichero `/sbin/lilo` (ejecutándolo lo instalamos en el lugar que indiquemos), teniendo como fichero de configuración `/etc/lilo.conf` (le decimos dónde y cómo lo vamos a instalar y cómo debe cargar el sistema operativo). La configuración del LILO se puede hacer, o bien desde el *prompt* del programa o editando el fichero de configuración, en cualquier caso, el cargador LILO usará las opciones que aparecen posteriormente, pero para su modificación hay que tener en cuenta cuales son las particiones que hemos creado (vienen reflejadas en el fichero `/etc/mtab`).

Ficheros relacionados con LILO:
/sbin/lilo
/etc/lilo.conf
/boot/boot.b
/boot/map
/boot/vmlinuz-version

Existen dos tipos de opciones (se pueden ver todas haciendo: `man lilo.conf`), las generales válidas para cualquier sistema colocadas al principio del fichero y las particulares para un sistema operativo (imagen) precedidas por su correspondiente etiqueta o título. Entre las globales podemos destacar:

- **boot=dispositivo**. Estamos diciendo que vamos a instalar LILO en el MBR de la partición (dispositivo) indicada. Se le indicará en que dispositivo (por ejemplo una partición del disco duro) se debe arrancar (o instalar) el sector de arranque (MBR) y por tanto donde está LILO, por defecto, será la partición donde se encuentre el sistema raíz de archivos LINUX. LILO siempre deberá estar en el primer disco IDE (`/dev/hda`) o SCSI (`/dev/sda`) o en un disquete (`/dev/fd0`). En el primer caso sólo se puede instalar en el MBR o en particiones primarias o extendidas pero no lógicas. La partición donde esté instalado debe ser de LINUX, no de otro sistema operativo.
- **install=fichero**. Se instala el archivo especificado como sector de arranque, sino se usará `/etc/lilo/boot.b`. Si hemos creado la partición especial de arranque boot se deberá usar `boot/boot.b`.
- **linear**. Genera direcciones lineales de sectores en vez de direcciones de tipo sector/cilindro/cabeza (para acceder a un disco, la BIOS utiliza direcciones conocidas como 3D o CHS [cilindros, cabezas, sectores] que obviamente dependen de la geometría del mismo, para manejar discos grandes, los fabricantes han introducido otro tipo de direccionamiento conocido como LBA, en que se accede a los discos de forma lineal) cuando se utiliza LBA.
- **message=fichero**. Para mostrar un mensaje antes de que se presente el prompt de boot. La longitud del fichero debe ser como máximo de 65535 bytes. Si se coloca 0xFF (Ctrl. L) en el mensajese borrará la pantalla.
- **verbose=nivel**. Será un número entre 1 y 5. Cuanto mayor es el número mayor información es presentada en pantalla.
- **backup=fichero**. Copia el sector de arranque en el fichero indicado.
- **delay=número**. Número de décimas de segundo que se esperará para arrancar la primera imagen (sistema operativo) indicada.
- **prompt**. Se exige que se escriba algo en el arranque de LILO.
    - **timeout=número**. Será el número de décimas de segundo de espera para escribir algo en el arranque. Si es un cero LILO esperará indefinidamente. Superado este tiempo se pasará a la opción por defecto indicada por default.
- **default=imagen**. Será el sistema operativo (imagen) por defecto (ver siguiente párrafo). Si no se coloca será la primera.
- **vga=modo**. Indica el modo gráfico de la pantalla en el boot. Puede ser `normal`, extendido (`ext`) o a petición (`ask`).
- **password=contraseña**. Se pueden proteger las imágenes de arranque. Esto es interesante para que nadie pueda venir con un disquete rearrancar el ordenador y cargar lo que el quiera. Combinado con el password de la Bios podemos proteger el sistema totalmente. Debido a que el fichero de configuración es visible, y que el password no va encriptado, deberemos poner en el mismo la protección 600, es decir lectura y escrituta sólo a root.

Si se modifica cualquiera de las opciones, se deberá ejecutar de nuevo LILO, usualmente con `lilo -v`, no importa hacerlo de nuevo si no estamos seguros de haber cambiado `lilo.conf` y no haberlo hecho antes. Esto es así ya que LILO cuando arranca no sabe nada del sistema de ficheros y lo que se hace al ejecutar el comando LILO es tomar referencias absolutas del disco para después poder botar.

Las opciones particulares para los sistemas operativos se clasifican en dos, las de LINUX encabezadas por la opción **image** y las de otros sistemas por **other**. Su dos encabezaminetos son:

- **image=núcleo**. Se indica dónde está la imagen del núcleo del sistema operativo que vamos a cargar con LILO.
- **other=dispositivo**. Es el equivalente de root en Linux y serviría para indicar cual es la partición donde se encuentra el sistema operativo.

En ambos casos, se pueden poner opciones comunes al principio y redefinirlas en alguna sección donde difieran. Las opciones más utilizadas son:

- **label=nombre**. Se define el nombre (etiqueta o título) de la imagen del sistema operativo. Si no se indica, se utilizará el nombre (sin camino) puesto en **image** o **other**.
- **alias=nombre**. Se define un segundo nombre para el sistema.
- **single-key**. Permite arrancar este sistema con una sola tecla, para ello la etiqueta o el alias deben ser de una sola letra.
- **password=contraseña**. Se define un acceso protegido para esta imagen.
- **restricted**. Se puede usar con **password** para aplicarlo sólo cuando se introduzca algo por teclado.

En el caso Linux se indicará la imagen a cargar, que normalmente estará en el directorio `/boot` con nombre `vmlinuz` seguido de la versión del núcleo (el kernel a cargar por LILO deberá estar en los primeros 1024 cilindros del disco IDE en la primara interfaz: hda o hdb, en las BIOS anteriores al 98, de ahí lo de la partición `/boot`, en otro caso se puede usar la opción LBA de la BIOS, siempre que esté permitido, y poner al LILO en modo lineal), opciones habituales específicas para Linux son:

- **root=dispositivo**. Indica que partición (dispositivo) será montada como raíz del sistema.
- **read-only**. Indica que la imagen no se debería cambiar, lo cual es natural.
- **append=cadena**. Se define una cadena a añadir a los parámetros pasados al núcleo del sistema. Por ejemplo será necesario cuando en el computador tengamos más de 64MB. de memoria: cadena = "mem 128M", por ejemplo.

En el caso de otros sistemas operativos podremos utilizar algunas de estas opciones:

- **table=dispositivo**. En algunos sistemas operativos es necesirio indicar que partición contiene la tabla de particiones, como por ejemplo la familia de MS-DOS, ya que de ella se toma información sobre el disco.
- **change**. Con diferentes subopciones permite modificar la tabla de particiones, como por ejemplo activar o desactivar una partición o cambiarlo de estado (tipo).
- **map-drive**. Se puede "cambiar" la BIOS para que por ejemplo un dico secundario pueda aparecer como primario y viceversa, ya que podemos tener un sistema que esté en este disco pero no pueda botar de él.

A continuación aparece una configuración normal del fichero /etc/lilo.conf cuando sólo existe un sistema operativo LINUX pero se pueden cargar dos núcleos:

```
     # opciones globales a todos los sistemas
boot=/dev/hda
map=/boot/map
```

```
     install=/boot/boot.b
prompt
timeout=50
message=/boot/message
linear
default=linux

     read-only
# opciones particulares para linux
image=/boot/vmlinuz-2.2.16-2
label=linux
root=/dev/hda6

     # opciones particulares para linux
image=/boot/vmlinuz-2.2.14-2
label=viejo
root=/dev/hda7
```

Si hubiera un sistema de doble arranque podría aparecer:

```
     # opciones particulares para sistema Microsoft
other=/dev/hda1
label=dos
table=/dev/hda
```

Además de estas opciones, cuando arranca el LILO le podemos dar mandatos on-line para comunicarle eventos especiales como que ponga al LINUX en modo monousuario (ver niveles de ejecución en un apartado posterior) o decirle que arranque desde otro dispositivo, las más usuales son:

- **Linux Rescue**. Inicia al Linux en modo monousuario, de tal manera que se puedan hacer reparaciones en el sistema.
- **Linux Single**. Del mismo efecto que la anterior intentando arrancar desde disco. Estas opciones deberían estar protegidas con **password** y usando la opción **restricted**.
- **Root=dispositivo**. Parecida a la entrada de lilo.conf, permite al usuario arrancar el sistema desde otros dispositivos como por ejemplo un CD-ROM.
- **Vga=modo**. Permite modificar el modo de la consola.

Normalmente se usan estos comandos on-line cuando estamos en situaciones de pérdida de LILO, por ejemplo cuando hemos instalado LILO en el MBR y después hemos instalado Windows y lo hemos borrado. En estos casos es vital tener siempre un disquete de arranque (el disquete de arranque se crea cuando se instala el sistema, después, como se indicó en el capítulo anterior, siempre se puede hacer con el comando `mkbootdisk`) y una vez que nos presenta el inductor de arranque (prompt de boot) escribir: **linux root=/dev/hda7** en nuestro caso (en esa partición estaba el sistema raíz). Una vez arrancado el sistema tendremos que ejecutar **lilo -v** para volver a instalar el LILO.

También resulta muy útil hacer una copia del MBR por si ocurre alguna desgracia, lo podemos conseguir con el conocido comando `dd`:

```
dd if=/dev/hda of=<archivo> bs=512 count=1
```

Para restaurar el IPL salvado sólo tenemos que ejecutar el comando:

```
dd if=<archivo> of=/dev/hda bs=446 count=1
```

Si no tuvieramos disquete de arranque (muy mal hecho), deberíamos proceder a la instalación, pero en vez de llevarla a cabo, pararnos cunado nos pida el boot e indicar el dispositivo de root. Si tenemos un disquete de rescate tampoco tendríamos problemas y procederíamos de la misma manera. Una vez que estemos en el sistema tendremos que ejecutar LILO y rebotar.

Un último aspecto que hay que tener en cuenta es si algún día se quiere cambiar el cargador de arranque LILO por otro, en este caso habrá que ejecutar **lilo -u**. Si lo hubieramos colocado en el MBR también habrá que eliminarlo de allí con **fdisk** /**mbr**.

Si se necesita más información sobre todas las opciones de LILO (por ejemplo ponerlo en modo gráfico) siempre (al igual que en otros aspectos de Linux) se puede recurrir al manual del sistema y a las páginas web de la distribución utilizada o de Linux en general (la página de Red Hat tiene un buscador para cualquiera de los sites de Linux). Allí encontraremos soporte software y hadware, faqs (preguntas frecuentes) y Howtos (Comos), además de foros y servicios de noticias donde podemos hacer consultas a otros usuarios de la red y Linux. En las referencias de este capítulo están varias direcciones de interés.

Como último recurso puedes recurrir a `Linuxconf` en sus sección de LILO.

3.3. Ejemplos Cargador LILO

El cargador

Ejemplos no probados tomados de la red.

1. Tengo un disco duro con sólo Linux, en la partición */dev/hda1*, y la imagen del núcleo es */boot/vmlinuz*. Queremos instalar Lilo como

gestor de arranque principal.

```
     boot=/dev/hda                  # Instalar Lilo en el MBR del primer disco IDE
image=/boot/vmlinuz     # Imagen del núcleo de Linux
  root=/dev/hda1                # Partición donde está el sistema raíz de Linux
  read-only                     # Opción recomendable para particiones Linux ext2
```

2. Tengo Linux en la partición primaria */dev/hda2*. Quiero instalar Lilo en el sector de arranque de esa partición porque uso otro gestor de arranque principal. La imagen del kernel es */boot/vmlinuz*.

```
     image=/boot/vmlinuz     # Imagen del núcleo de Linux
  root=/dev/hda2                # Partición del sistema raíz y donde se va a instalar Lilo
  read-only                     # Opción recomendable para particiones Linux ext2
```

3. Tengo el sistema raíz de Linux en la unidad lógica */dev/hda5*. Quiero instalar Lilo en el sector de arranque de la partición extendida, que es la */dev/hda4*, porque uso otro gestor de arranque principal.

```
     boot=/dev/hda4                 # Instalar Lilo en el sector de arranque de la partición extendida
image=/boot/vmlinuz     # Imagen del núcleo de Linux
  root=/dev/hda5                # Unidad lógica donde está el sistema raíz de Linux
  read-only                     # Opción recomendable para particiones Linux ext2
```

4. Tengo un disco duro con Ms-Dos en */dev/hda1*, Windows 95/98 con FAT32 en */dev/hda2*, NT en */dev/hda3* y Linux en la unidad lógica */dev/hda5* de la partición extendida */dev/hda4*. Además tengo un segundo disco duro con Ms-Dos o Windows 95/98. Quiero usar Lilo como gestor de arranque principal y se imprimirá un mensaje inicial.

```
     # Opciones globales

boot=/dev/hda                   # Instalar Lilo en el MBR del primer disco IDE
message=/boot/message   # Mensaje inicial
single-key                              # Permitir seleccionar el OS con la pulsación de una sola tecla
change-rules                    # Inicio de definiciones de tipos de partición
   type=DOS32 normal=0x0b       # Definición del tipo de partición FAT32
   type=NTFS normal=0x07        # Definición del tipo de partición NTFS

# Sección de la imagen de Linux

image=/boot/vmlinuz     # Imagen del núcleo de Linux
   label=linux                  # Nombre identificativo
   alias=1                              # Nombre alternativo
   root=/dev/hda5               # Unidad lógica donde está el sistema raíz de Linux
   read-only                    # Opción recomendable para particiones Linux ext2

# Sección para el Ms-Dos

other=/dev/hda1         # Partición donde está el Ms-Dos
 label=dos                      # Nombre identificativo
 alias=2                        # Nombre alternativo
 table=/dev/hda         # Opción necesaria para los OS de Microsoft
 change                         # Inicio de la sección de modificación del MBR
   partition=/dev/hda1          # Modificar la partición Ms-Dos
    activate                            # Activar el flag de arranque
    set DOS16_big_normal        # Hacer visible la partición
  partition=/dev/hda2           # Modificar la partición Win 95/98
    deactivate                          # Desactivar el flag de arranque
    set DOS32_hidden            # Ocultar la partición
  partition=/dev/hda3           # Modificar la partición del NT
    deactivate                          # Desactivar el flag de arranque
    set NTFS_hidden                     # Ocultar la partición

# Sección para el Win 95/98

other=/dev/hda2         # Partición donde está el Win 95/98
  label=win                     # Nombre identificativo
  alias=3                       # Nombre alternativo
  table=/dev/hda        # Opción necesaria para los OS de Microsoft
  change                                # Inicio de la sección de modificación del MBR
   partition=/dev/hda1          # Modificar la partición del Ms-Dos
     deactivate                         # Desactivar el flag de arranque
     set DOS16_big_hidden       # Ocultar la partición
   partition=/dev/hda2          # Modificar la partición del Win 95/98
     activate                           # Activar el flag de arranque
     set DOS32_normal           # Hacer visible la partición
   partition=/dev/hda2          # Modificar la partición del NT
     deactivate                         # Desactivar el flag de arranque
     set NTFS_hidden            # Ocultar la partición

# Sección para el NT

other=/dev/hda3         # Partición donde está el NT
 label=nt                       # Nombre identificativo
 alias=4                        # Nombre alternativo
 table=/dev/hda         # Opción necesaria para los OS de Microsoft
 change                                 # Inicio de la sección de modificación del MBR
  partition=/dev/hda1   # Modificar la partición del Ms-Dos
   deactivate                   # Desactivar el flag de arranque
```

```
        set DOS16_big_hidden # Ocultar la partición
   partition=/dev/hda2   # Modificar la partición del Win 95/98
    deactivate                  # Desactivar el flag de arranque
    set DOS32_hidden     # Ocultar la partición
   partition=/dev/hda2   # Modificar la partición del NT
    activate                    # Activar el flag de arranque
    set NTFS_normal             # Hacer visible la partición

# Sección para el otro disco duro

other=/dev/hdb1          # Partición del segundo disco donde está el OS
 label=disco2            # Nombre identificativo
 alias=5                         # Nombre alternativo
 table=/dev/hdb          # Opción necesaria para los OS de Microsoft
 map-drive=0x80          # Intercambia el primer y segundo disco en la Bios
  to=0x81                        # para así poder arrancar del segundo disco
 map-drive=0x81          # Entonces, para la Bios, el segundo disco será el
  to=0x80                        # primero, y el primero será el segundo
```

3.4. Inicialización

Inicialización

Una vez que el kernel está cargado y sabe cuanta memoria tiene (en la mayoría de los casos el kernel utiliza una cantidad fija de memoria para sus tablas internas), chequea el entorno de la máquina para saber que hardware está disponible (se supone que cuando se construye un kernel, el usuario le ha dado esa información como hemos visto en el capítulo anterior) y lo inicializa. Si la información pasada al kernel es insuficiente, el kernel probará los drivers de los dispositivos y en caso necesario requerirá información de los mismos. Si éstos no responden, sus dispositivos serán desactivados. Ocurrirá lo mismo si los dispositivos se conectan después de la inicialización, no serán accesibles hasta que el sistema bote de nuevo (esto no es del todo cierto en núcleos que soporten módulos, como es el caso del LINUX actual, ya que un módulo puede soportar un driver y es reconocido después de la inicialización. Los módulos se pueden cargar o descargar con los comandos `insmod` y `rmmod` o directamente por el núcleo a través de su demonio kerneld).

Una vez que la configuración hardware está realizada, el núcleo crea los procesos espontáneos en el espacio del usuario, cuyo número y nombre dependerá del sistema en cuestión; en el caso de sistemas BSD serán tres: el **swapper** (0), el proceso **init** (1) y el **pagedaemon** (2); en sistemas ATT (System V): el **sched** (0), el **init** (1) y varios manejadores de memoria.

En este punto, la misión de inicialización del núcleo ha terminado, pero ninguno de los procesos que se encarga de operaciones básicas como aceptar logins y terminales ha arrancado todavía, al igual que el resto de demonios del sistema. Toda está inicialización del sistema es función del proceso **init** de PID número 1, que es el único proceso que realmente se ejecuta en el espacio de usuario, ya que el resto de procesos espontáneos son copias de parte del núcleo que se separan por razones de planificación.

Ficheros y Directorios relacionados con init:
/etc/rc*
/etc/init.d
/etc/rcx.d

Si se ha indicado al núcleo del sistema que la inicialización va a ser en modo monousuario (un de los niveles de ejecución), éste al crear a **init** se lo indicará, con lo cual este proceso sólo ejecutará una shell de tipo `sh`. A través de ésta shell el superusuario o root podrá ejecutar la mayoría de los comandos que ejecutaría con el sistema completo, con la salvedad de que sólo la partición de root es montada y que la mayoría de los demonios no están creados, con lo que no funcionarán servicios básicos de red como el correo o comandos que no residan en `/bin`, `/sbin` o `/etc`. El comando que con mayor frecuencia se utilizará será el **fsck** que sirve para chequear el sistema de ficheros y repararlo, éste se ejecuta de forma automática en el bote normal (otro nivel de ejecución), pero en este caso se deberá hacer manualmente.

Una vez que el proceso **init** está cargado y ejecutándose, el siguiente paso será la puesta en marcha de las macros de arranque. Para ejecutarlas el proceso init arranca una **sh** que las interpretará, la ubicación, contenido y nombre de las macros es muy dependiente (de nuevo) del sistema. En los sistemas BSD están guardadas en el directorio `/etc` y empiezan por rc. En los sistemas ATT las macros están en el directorio `/etc/init.d` y están linkadas (con **ln**) al directorio `/etc` con nombres rcx.d donde x es un número que indicará el nivel de ejecución (ver posteriormente).

Las funciones típicas de estas macros serán:

- Dar nombre al computador.
- Poner la zona horaria.
- Chequear el sistema de ficheros.
- Montar las particiones.
- Borrar ficheros de /tmp.
- Configurar la red.
- Arrancar demonios y servicios de red.
- Activar las cuentas y las cuotas.

Las macros suelen ser muy "expresivas", técnicamente verbosas, y nos irán diciendo en cada momento lo que están haciendo, lo cual es muy útil cuando hay problemas en el arranque.

Una vez ejecutadas, el sistema está listo para aceptar a los usuarios. Para que esto ocurra el proceso **init** ha creado a un proceso **getty** por

cada terminal. Si se recibe una entrada, este proceso creará a su vez a un proceso **login** que verificará el nombre y la contraseña del usuario y si todo va bien se creará una **shell** de comunicación con el mismo.

3.5. Inicialización ATT/ REDHAT

Inicialización

Debido a que esta es otra de las partes del sistema que varía bastante entre sistemas operativos, en este apartado nos vamos a centrar en la inicialización del Red Hat que sigue fundamentalmente el esquema de ATT (System V).

La gran diferencia con sistemas de tipo BSD es que en ATT el proceso ***init*** soporta varios niveles de ejecución (el proceso ***init*** de puede ejecutar en línea con el número de nivel de ejecución, por ejemplo init 0 apagará el sistema) que habilitan diferente recursos, de esta manera las macros son divididas en partes más pequeñas que son ejecutadas dependiendo del nivel de ejecución.

Fichero o directorio	Tipo	Función
/etc/inittab	F	Indica que hacer en cada nivel de ejecución
/etc/rc.d	D	Directorio de macros de arranque
/etc/rc.d/rc.sysinit	F	Macro común de inicio
/etc/rc.d/rc	F	Macro que toma el nivel de arranque
/etc/rc.d/rc.local	F	Macro para características particulares
/etc/rc.d/rcx.d	D	Directorios de arranque para cada nivel de ejecución
/etc/rc.d/init.d	D	Directorio de ubicación de servicios

Los niveles de ejecución están planificados (le dicen a ***init*** lo que debe hacer en cada uno) en el fichero ***/etc/inittab***. Si este fichero está defectuoso o no existe, la única forma de botar será en modo monousuario (si el ordenador no arranca por culpa de un /etc/inittab mal definido o no permite la conexión porque hay un `/etc/passwd` corrupto o simplemente la contraseña de root ha sido olvidada, basta con arrancar en modo monousuario tecleando `linux single`, si no está protegido, en la línea de comandos de LILO). El fichero de texto está compuesto por líneas de un formato parecido a este:

```
id:ejecucion:acción:proceso
```

donde **id** es la identificación de la entrada dentro de ***inittab***, **ejecucion** será el nivel o los niveles de ejecución asociados a esa entrada, **proceso** será una línea ***sh*** que indicará a ***init*** que proceso ejecutar cuando se especifica alguno de los niveles correspondientes a la entrada y **acción** especifica lo que deberá hacer ***init*** con el proceso, las más usadas son: ***wait*** esperar a que acabe, ***off*** ignorar la entrada, ***once*** ejecutarle sólo una vez, ***respawn*** volverle a ejecutarlo si muere y la más importante ***sysinit*** que indicará a ***init*** que debe preguntar al operador.

Los niveles de ejecución (en Red Hat hay hasta 10 niveles pero del 7 al 9 están indefinidos) son:

0 — **Halt**. El nivel 0 se utiliza para cerrar el sistema. Las tareas a realizar serán fundamentalmente y por ese orden: terminar todos los procesos, desactivar la zona de intercambio de memoria en disco y desmontar los sistemas de ficheros.

1 — **Monousuario**. Este es el estado administrativo usado por los system managers para hacer mantenimiento de software. Los sistemas de ficheros están montados pero la red está inactiva.

2 — **Multiusuario**. Este es el nivel de ejecución multiusuario, pero con los servicios de red parcialmente inactivos, como por ejemplo el NFS.

3 — **Multiusuario**. Este suele ser el nivel de ejecución predeterminado (primera línea de inittab). Todos los servicios están activos incluyendo los de red.

4 — **Sin usar**. Este es un directorio vacío que se usa para que sea determinado por el administrador.

5 — **Multiusuario**. Es muy parecido al tres con la diferencia que el servicio ***named*** (servidor de nombres) está activo y que además es el elegido cuando se utilizan sesiones gráficas basadas en el sistema X.

6 — **Reinicialización**. Es parecido al nivel cero, con la diferencia de que la lógica de init.d indicará si el sistema está iniciándose o terminando.

Obviamente el sistema no puede permanecer en los niveles 0 y 6, se usan para pararlo o rebotarlo. El aspecto de un fichero ***inittab*** típico puede ser (el fichero ***inittab*** se puede editar para hacer cambios, aunque resulte muy peligroso y muy poco usual):

```
#
# inittab       This file describes how the INIT process should set up
#               the system in a certain run-level.
#
# Author:       Miquel van Smoorenburg, <miquels@drinkel.nl.mugnet.org>
#               Modified for RHS Linux by Marc Ewing and Donnie Barnes
#
# Default runlevel. The runlevels used by RHS are:
```

```
      #   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not have networking)
#   3 - Full multiuser mode
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
#
id:3:initdefault:
# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit
#
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
#
# Things to run in every runlevel.
ud::once:/sbin/update
#
# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
#
# When our UPS tells us power has failed, assume we have a few minutes
# of power left.  Schedule a shutdown for 2 minutes from now.
# This does, of course, assume you have powerd installed and your
# UPS connected and working correctly.
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"
#
# If power was restored before the shutdown kicked in, cancel it.
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"
#
# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
# Run xdm in runlevel 5
# xdm is now a separate service
x:5:respawn:/etc/X11/prefdm -nodaemon
```

Lo primero que veremos en este tipo de ficheros es el nivel por defecto que se escogerá al arrancar sin indicar nada, en este caso el 3 con la línea:

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

A continuación con:

```
si::sysinit:/etc/rc.d/rc.sysinit
```

indicaremos una serie de servicios comunes a todos los niveles de ejecución dados en el fichero `/etc/rc.d/rc.sysinit`. Funciones típicas de esta macro son: configurar variables para el PATH, la red, el intercambio de memoria, el hostname, montar el sistema de ficheros raíz y borrar algunos ficheros no necesarios. Entrar en más detalles se sale de este curso básico.

Después, arrancaremos una macro llamada ***rc*** residente en el directorio `/etc/rc.d` a la que se la pasa el nivel de ejecución. Ésta, además de configurar el nivel de ejecución actual y anterior, irá a un directorio de formato rcx.d, donde x es el nivel de ejecución en el directorio `/etc/rc.d`, donde aparecerán una serie de enlaces a ficheros. Estos ficheros son servicios que están en el directorio `/etc/rc.d/init.d` y también están en forma de macro. Los enlaces empiezan por S o K seguidos de un número y un nombre, y nos indicarán que servicios arrancar (*start*) o terminar (*kill*) por la macro ***rc*** y en que orden (por el número, como las líneas de BASIC). Si encuentra una S, la macro ***rc*** mandará un *start* al servicio y si es una K un *stop*. Debemos tener en cuenta que cualquier macro (incluidas las de los servicios) puede ser ejecutada manualmente desde la línea de comandos. Así, un comando como:

```
/etc/rc.d/init.d/httpd stop
```

detendría el servidor de httpd (de web). Podemos ver todos los servicios simplemente haciendo un `ls` del directorio. Normalmente cada servicio está asociado a un demonio.

La macro ***rc*** (en este ejemplo se ve con claridad la complejidad que puede alcanzar una macro en UNIX) tendrá un aspecto parecido a las siguientes líneas (en lo que se refiere a parar y arrancar servicios):

```
      ...
# Is there an rc directory for this new runlevel?
if [ -d /etc/rc$runlevel.d ]; then
       # First, run the KILL scripts.
        for i in /etc/rc$runlevel.d/K*; do
                # Check if the script is there.
                [ ! -f $i ] && continue
                # Don't run [KS]??foo.{rpmsave,rpmorig} scripts
                [ "${i%.rpmsave}" != "${i}" ] && continue
                [ "${i%.rpmorig}" != "${i}" ] && continue
                [ "${i%.rpmnew}" != "${i}" ] && continue
                # Check if the subsystem is already up.
                subsys=${i#/etc/rc$runlevel.d/K??}
                [ ! -f /var/lock/subsys/$subsys ] && \
                    [ ! -f /var/lock/subsys/${subsys}.init ] && continue
                # Bring the subsystem down.
                if egrep -q "(killproc |action )" $i ; then
                        $i stop
                else
                        action "Stopping $subsys: " $i stop
                fi
        done
```

```
             # Now run the START scripts.
        for i in /etc/rc$runlevel.d/S*; do
                # Check if the script is there.
                [ ! -f $i ] && continue
                # Don't run [KS]??foo.{rpmsave,rpmorig} scripts
                [ "${i%.rpmsave}" != "${i}" ] && continue
                [ "${i%.rpmorig}" != "${i}" ] && continue
                [ "${i%.rpmnew}" != "${i}" ] && continue
                # Check if the subsystem is already up.
                subsys=${i#/etc/rc$runlevel.d/S??}
                [ -f /var/lock/subsys/$subsys ] || \
                   [ -f /var/lock/subsys/${subsys}.init ] && continue
                # If we're in confirmation mode, get user confirmation
                [ -n "$CONFIRM" ]  &&
                  {
                    confirm $subsys
                    case $? in
                      0)
                        :
                      ;;
                      2)
                        CONFIRM=
                      ;;
                      *)
                        continue
                      ;;
                    esac
                  }
                # Bring the subsystem up.
                if egrep -q "(daemon |action )" $i ; then
                        $i start
                else
                         if [ "$subsys" = "halt" -o "$subsys" = "reboot" -o "$subsys" = "single" -o
"$subsys" = "local" ]; then
                             $i start
                        else
                             action "Starting $subsys: " $i start
                        fi
                fi
        done
fi
```

Y la macro del servicio más sencillo, que es killall tendría un aspecto:

```
     #!/bin/bash
# Bring down all unneeded services that are still running (there shouldn't
# be any, so this is just a sanity check)
for i in /var/lock/subsys/*; do
        # Check if the script is there.
        [ ! -f $i ] && continue
        # Get the subsystem name.
        subsys=${i#/var/lock/subsys/}
        # Bring the subsystem down.
        if [ -f /etc/init.d/$subsys.init ]; then
            /etc/init.d/$subsys.init stop
        else
            /etc/init.d/$subsys stop
        fi
done
```

Una vez que la macro ***rc***, llamada adecuadamente por el proceso ***init*** a través del fichero ***inittab***, ha terminado su trabajo, se pasará a ejecutar el comando `/sbin/update`, que crea un demonio del kernel que manda los buffers "sucios" (los que han sido escritos) a disco (en muchas ocasiones esto suele estar incluido en la propia ***rc***).

También `/etc/inittab` describe como el sistema debe gestionar la traducción de Ctrl-Alt-Delete en algo como el comando `/sbin/shutdown -t3 -r now` (rebotará el sistema ahora haciendo que ***init*** espere 3 segundos). Y a continuación `/etc/inittab` indica que debe hacer el sistema si falla el fluido eléctrico, esto sólo tendrá sentido cuando esté conectado un sistema de alimentación initerumpida UPS.

Después el fichero `/etc/inittab` se replica para ejecutar un proceso ***getty*** para cada consola virtual. En este caso los niveles 2-5 tienen seis consolas, el nivel 1, que es monousuario, sólo tiene una consola y como es natural, los niveles 0 y 6 no tienen consolas virtuales. Cuando estamos trabajando en una consola normal de Linux (y también otros sistemas), tenemos varias consolas virtuales en el mismo dispositivo de salida (pantalla) a las que se accede a través de la combinación de teclas Ctrl-Alt-tecla de función, normalmente la quinta (F5) está reservada como terminal X (gráfico) y el resto son de texto.

En el nivel de ejecución 5, también se ejecuta una macro llamando ***/etc/X11/prefdm***. El script ***prefdm*** configura la variable PATH de forma adecuada para el entorno gráfico y ejecuta el gestor de pantalla preferido (*gdm* si escogemos GNOME, *kdm* si KDE, o *xdm* en otro caso, AnotherLevel) basándose en el fichero `/etc/sysconfig/desktop`, si en éste no aparece nada, cogerá el primero indicado.

A continuación, en Red Hat (a semejanza de los UNIX de la familia BDS como en Freebsd) se ejecuta opcionalmente la macro ***rc.local***, donde se coloca la parte de la inicialización que no se ha querido poner en las macros anteriores de tipo System V. Suele usarse para obtener el nombre del sistema operativo y la arquitectura del computador, adaptando el fichero `/etc/issue`. En algunos casos se puede colocar ***setserial*** en ***rc.local*** o ***rc.serial*** si se tienen puertos serie que inicializar. También se pueden poner arranques de demonios adicionales o inicializar una impresora. Pero el `/etc/rc.d/rc.local` por defecto, simplemente crea un bonito mensaje de bienvenida (/etc/issue) con la versión del kernel y el tipo de ordenador.

Otras configuraciones posibles, al estilo de `/etc/rc.config.d` usados en los sistemas HP-UX, se obtienen modificando los ficheros que se encuentran en el directorio `/etc/sysconfig` que usualmente contienen definiciones de variables para "sintonizar" el sistema. En la siguiente tabla aparecen las más frecuentes:

Fichero	Función
apmd	Manejo del demonio de energía (power).
clock	Tipo de reloj que se tiene.

desktop	Tipo de escritorio.
harddisks	Tipo de disco.
hwconf	Información usada por Kudzu (proceso de arranque).
i18n	Formato local de fechas, lenguajes, etc.
init	Manera en que los mensajes de init son enseñados.
keyboard	Tipo de teclado.
mouse	Tipo de ratón para las X.
network	Opciones globales de la red: hostname, gateway, etc.
pcmcia	Para arrancar demonios relacionados con la PCMCIA.
sendmail	Opciones del demonio de correo sendmail.

También existen dentro de él otros directorios de macros como `console` o `network-scripts`.

3.6. Herramientas

Existen tres comandos relacionados con el arranque del sistema que nos permitirán modificar los servicios a arrancar o parar sin necesidad de crear enlaces en los ficheros `rcx.d`. El primero es de tipo gráfico y se llama ***tksysv*** (propio de Red Hat), que puede ejecutarse tanto desde la línea de comando gráfico como desde el panel de control, tiene su propia ayuda y es muy sencillo de usar, fue creado por Donnie Barnes y está escrito en `tcl/tk`, de ahí el nombre "tk System V".

Existe una versión con un interfaz de carácter con ventanas que se llama ***ntsysv***, que hace la misma labor, fue realizado por Erik Troan.

Por último, el comando más básico, del que provienen los anteriores (y éste a su vez de IRIX) es ***chkconfig***. La definición de la utilidad ***chkconfig*** es: “proveer de una simple herramienta de línea de comandos para mantener la jerarquía de directorios `/etc/rc.d`”. De esta manera libera a los administradores de sistemas de tener que manipular directamente numerosos enlaces simbólicos en /etc/rc.d.

Se puede tener información completa a través del comando ***man***, pero la opción más habitual es:

```
chkconfig [--level levels] name <on|off|reset>
```

donde indicaremos el nivel de ejecución el nombre del servicio y lo que queremos hacer con él.

3.7. Cierre del Sistema

Ya sabemos como se inicia un sistema UNIX, nos queda ver, para concluir este capítulo, como se cierra y como podemos cambiar el nivel de ejecución.

Para apagar un sistema Red Hat LINUX (en general cualquier sistema UNIX), hay que ejecutar el comando **`shutdown`**, jamás deberemos apagar el sistema directamente como si se tratara de un sistema MS-DOS, esto no es una manera aceptable de parar el sistema, ya que corremos el riesgo de perder información de archivos e incluso estropear algunos discos viejos sin *autoaparque*, además de la seguridad de perder tiempo en la inicialización posterior del sistema, que será mucho más lenta debido al obligatorio chequeo del sistema de ficheros. Pero existen otras formas para realizar esta parada que podemos describir aquí:

- Usando **`shutdown`**. Es la forma ortodoxa de parar un sistema. Podemos encontrar más detalles en las páginas de manual de `shutdown`, ya que es un comando que depende en alguna medida del tipo de UNIX que tenemos, incluso su ubicación (en Red Hat está en `/sbin`), pero básicamente lo que hará es parar el sistema en algún periodo de tiempo (puede ser inmediato), mientras tanto podrá mandar mensajes de advertencia a los usuarios que estén conectados. Ambos parámetros se pueden especificar.
  La parada la efectuará enviando señales de tipo SIGTERM a los procesos que se estén ejecutando (lo que les permite salir de un modo seguro) y cambiado a ***init*** de nivel de ejecución: 0 para parar el sistema, 6 para rebotarlo o 1 para pasar a modo monousuario. Esto se consigue con las opciones `-h`, `-r` y (sin opciones). Existen otras opciones menos usados como `-a` (para permitir que sólo usuarios dados

en `/etc/shutdown.allow` puedan echar el sistema abajo), `-f` para chequear el sistema de ficheros con `fsck`, `-c` para cancelar un `shutdown` en espera o `-k` para mandar sólo el mensaje a todos los usuarios pero no pararlo. Las opciones en general son:

```
shutdown [-t sec] [-arkhncfF] time [warning-message]
```

- Usando los comandos BSD `reboot` y `halt`. Son equivalentes a `shutdown -r now` y `shutdown -h now`. Es una mala costumbre utilizarlos, ya que no todos los sistemas operativos tipo LINUX y similares soportan esta característica.

- Mandando al proceso ***init*** una señal de tipo TERM. El resultado de enviar a ***init*** esta señal suele ser impredecible y un poco sucio. Normalmente ***init*** reacciona matando todos los procesos de usuario, los demonios, los ***getty*** y poniendo el sistema en modo monousuario.

- Cambiando de nivel de ejecución. El comando ***telinit*** indica al proceso ***init*** que cambie de nivel de ejecución, suele usarse con un argumento de tipo numérico (nivel de ejecución) de 0 a 6 o más brevemente S o s para pasar a modo monousuario. También se le puede indicar que vuelva a leer el fichero `/etc/inittab` con la opción `-q`. Utilizar esta vía no nos ofrecerá ningún periodo de espera, ni ningún mensaje de advertencia, por eso es muchísimo mejor utilizar `shutdown -iS` (en Red Hat simplemente `shutdown`).

- Matando a ***init***. En la mayoría de los computadores esto producirá un rebote automático, incluso en algunos *kernel* se entrará en modo pánico. Es tan desaconsejable como apagar el computador.

3.8. Consejos

Consejos

Para acabar el capítulo daremos unos consejos finales que no se deberán eludir:

- Utilizar sólo ***root*** para tareas administrativas, no por defecto. El mismo Linus borró accidentalmente todo el sistema de ficheros por error, aunque después pudo recuperar los ficheros uno a uno debido a sus conocimientos del sistema, pero recordar que son miles de ficheros.

- Tener disponible copias de seguridad. Ver capítulo quinto.

- Tener discos de emergencia, al menos uno que incluya el kernel que se está utilizando. Si se puede otro con los ficheros imprescindibles.

- No apagar el ordenador, usar siempre `shutdown`.

- No preocuparse por la fragmentación del disco en LINUX y utilizar sistemas de ficheros de tipo extendido (`e2fs`).

- Si hay problemas con el arranque, conocer bien las herramientas de chequeo de discos (ver capítulo quinto).