

Inicio del sistema

Comenzaremos el análisis de la administración de sistemas Linux por el primer proceso que se ejecuta en cuanto se enciende la PC: la inicialización del sistema.

El proceso de inicialización	16
Descripción general	16
Gestores de arranque	19
Niveles de ejecución	24
Personalización del inicio en modo texto	29
Resumen	31
Actividades	32

EL PROCESO DE INICIALIZACIÓN

Los administradores de sistemas prestamos especial importancia a todo lo que pasa en el proceso de inicialización de nuestros sistemas operativos. Este proceso es el pilar básico de todo lo que sucede sobre el sistema, ya que en este punto todas sus necesidades y configuraciones se cargan en memorias. Durante el proceso de iniciación se cargan, en primer lugar, los controladores de hardware. Ellos, a su vez, darán pie a la carga de servicios que hagan uso de estos controladores. También se cargan programas residentes, servidores, sistemas de registro de actividades, y mucho más.

En este capítulo vamos a analizar de qué manera se desarrolla el proceso de inicialización de Linux y aprenderemos a configurarlo al máximo.

DESCRIPCIÓN GENERAL

Como todos sabemos, el núcleo del sistema operativo es aquel componente que se encarga de interactuar casi directamente con la parte física de nuestro sistema (el hardware). Entre otras cosas, el núcleo gestiona el control de los procesos, la asignación de los datos en la memoria, la planificación de las tareas, el acceso físico a dispositivos de almacenamiento, y mucho (pero créanme que mucho) más.

Linux es el núcleo del sistema GNU (de allí que el nombre completo del sistema sea GNU/Linux) y, como tal, es parte vital del mismo. Ahora bien, analizando el archivo (los invito a dar una mirada al directorio **/boot**) encontraremos que se trata de un archivo binario, sin propiedades de ejecución, y que, a simple vista, inspira humildad si consideramos su tamaño. Esto es así porque el núcleo de nuestro sistema es el fruto binario de la compilación de cientos de miles de líneas de código. A la fecha de redacción de este libro, el código fuente del núcleo Linux pesa cerca de 60 MB comprimido. Imaginemos el tamaño una vez descomprimido. Tal vez nos preguntemos: ¿Y cómo es que se convierte en algo tan pequeño? Sencillo, durante el proceso de compilación, el encargado de esta tarea selecciona los componentes que serán parte del núcleo. En este punto, por ejemplo, se selecciona el soporte a diferentes sistemas de archivos, el soporte a dispositivos especiales de hardware, y mucho más. De ahí que cuando se compila el núcleo Linux simplemente se compila lo que el usuario necesitará en su máquina, y no la totalidad del código fuente.

Precisamente en ese momento de configuración del soporte del núcleo, el operador decide si determinado componente estará dentro o fuera del mismo. Que un componente esté dentro del núcleo quiere decir que conformará parte del archivo final. Cuantos más componentes incluyamos dentro del núcleo, mayor será su tamaño.

Ahora bien, los componentes que son configurados para no ser partes del núcleo se llaman **módulos**. Los módulos son como planetas que giran alrededor del núcleo y están conectados directamente a él. La ventaja es que es posible tener un mayor control sobre ellos, ya que son independientes del núcleo.

Durante la fase de inicialización del sistema, nuestro entorno pasa por dos ámbitos totalmente diferentes. El primero de ellos es el “espacio de núcleo”, en donde el propio núcleo y todos los componentes que hayan sido incluidos dentro de él tienen acceso directo a la memoria del sistema. Es decir que pueden hacer lo que quieran sin necesidad de andar pidiendo permisos a algún gestor de procesos o de memoria.

Cuando encendemos nuestra computadora, lo primero que se carga es el gestor de arranque, luego, el núcleo, y éste entra directamente a ejecutarse en “espacio de núcleo”. Una vez que todos los componentes que requieren acceso directo al hardware han sido cargados, el sistema pasa a modo “espacio de usuario”, en donde se comienzan a cargar los componentes que han sido configurados como módulos (satélites o planetas del núcleo) y cualquier otro programa que utilice el usuario. Una vez que el sistema entró en “espacio de usuario”, ya no se puede salir de él. ¡En realidad, ya nadie quiere salir de él! El espacio de usuario provee un sistema de memoria protegida a los programas, en el cual cada uno de estos opera dentro de su propio ámbito de memoria, sin molestar a los demás y de manera totalmente independiente. Si un programa quedara colgado o a la espera de una respuesta que nunca llegará, esto no molestará en lo más mínimo a la ejecución del resto de los programas ni del sistema mínimo.

Nuestro objetivo en esta primera parte del capítulo es identificar las diferentes fases de ejecución del proceso de inicialización del sistema operativo. Es muy fácil decir que, cuando vemos el menú de inicio (de LILO o GRUB, que veremos en detalle más adelante), lo que estamos viendo es en realidad el gestor de arranque. Pero ¿cuándo comienza la ejecución en espacio de núcleo? ¿Y cuándo comienza la ejecución en espacio de usuario? Podemos simplificar este análisis afirmando que una vez que el gestor de arranque obtiene lo que necesita para cargar el núcleo (un **timeout**, o una tecla **ENTER** que le dé la orden de empezar a ejecutar el sistema operativo), el sistema se ejecuta en espacio de núcleo. Lo que veremos en pantalla es algo similar a la **Figura 1**.



EL SISTEMA A LA MANO

Es recomendable que leamos los capítulos prácticos del libro con una computadora con GNU/Linux cerca. ¡Hay muchos datos útiles e interesantes que probar!



MEMORIA PROTEGIDA

El sistema en el cual cada programa ocupa su propio espacio de memoria y no molesta al resto se llama “memoria protegida”, y es lo que hace que GNU/Linux sea tan estable.

```

Alternarama:~# dmesg lmore
Linux version 2.4.18-686 (herbert@gondolin) (gcc version 2.95.4 20011002 (Debian prerelease)) #1 Sun Apr 14 11:32:47 EST 2002
BIOS-provided physical RAM map:
 BIOS-e820: 0000000000000000 - 000000000009fc00 (usable)
 BIOS-e820: 000000000009fc00 - 00000000000a0000 (reserved)
 BIOS-e820: 00000000000f0000 - 0000000000100000 (reserved)
 BIOS-e820: 0000000000100000 - 0000000000ddf0000 (usable)
 BIOS-e820: 0000000000ddf0000 - 0000000000ddf8000 (ACPI data)
 BIOS-e820: 0000000000ddf8000 - 0000000000de00000 (ACPI NVS)
 BIOS-e820: 0000000000ffef0000 - 0000000000ffff0000 (reserved)
 BIOS-e820: 0000000000ffff0000 - 0000000100000000 (reserved)
On node 0 totalpages: 56816
zone(0): 4096 pages.
zone(1): 52720 pages.
zone(2): 0 pages.
Local APIC disabled by BIOS -- reenabling.
Found and enabled local APIC!
Kernel command line: auto BOOT_IMAGE=Linux ro root=1601
Initializing CPU#0
Detected 334.095 MHz processor.
Console: colour VGA+ 80x25
Calibrating delay loop... 666.82 BogoMIPS
--More--
    
```

Figura 1. Si deseamos ver esta información en nuestros sistemas, no tenemos más que escribir el comando *dmesg*.

Como podemos ver, la información comienza con datos del BIOS, luego avanza con datos de memoria, CPU, buses de comunicación, dispositivos de almacenamiento, etc. Es decir, la información se desarrolla “desde dentro hacia fuera” y es fiel a lo que sucede realmente en el sistema cuando el núcleo se carga en memoria. Cuando decimos “desde dentro hacia fuera” significa que se desenvuelve un desarrollo jerárquico que tiene como único fin poder montar la partición del sistema y empezar a cargar los servicios realmente útiles para el usuario. No es posible cargar una partición si antes no se definió cuál es el disco duro. No es posible cargar un disco duro si antes no se definió el tipo de bus de conexión que emplea. No es posible definir un tipo de bus de conexión sin antes definir el tipo de procesador y BIOS.

Una vez que sucedió todo esto, entonces el gestor de inicialización del núcleo abandona el espacio de núcleo para entrar en el “espacio de usuario”, del cual ya nunca saldrá y en el cual se ejecutarán todos los servicios, drivers y otros módulos que se hayan compilado como satélites del núcleo. Lo primero que se hace cuando el sistema entra en espacio de usuario es “cambiar” al nivel de ejecución configurado por defecto. Los

III TODO ES CONFIGURABLE

Todas estas tareas que realiza el sistema operativo cuando se inicia son totalmente configurables y modificables por el administrador del sistema. Ésta es otra de las indiscutibles ventajas del sistema operativo GNU/Linux. A lo largo de este manual, descubriremos cómo lograrlo.

niveles de ejecución son configuraciones que definen el entorno y la forma en la cual funcionará el sistema. Son ejemplos de niveles de ejecución el modo monousuario (en el cual sólo puede ingresar en el sistema un usuario a la vez), el modo multiusuario (con soporte de múltiples accesos locales y vía red) y el modo gráfico, el cual ofrece toda una serie de herramientas visuales para operar el sistema. La configuración de los diferentes niveles de ejecución es un tema que será abordado más adelante, en este mismo libro.

Finalmente, cuando todo lo que estaba configurado para ser inicializado en el nivel de ejecución definido ha sido cargado, nos topamos con la famosa pantalla de “login”, en donde tenemos la posibilidad de ingresar un nombre de usuario y una contraseña. Ahora que tenemos una idea más clara de cómo funciona el proceso de inicialización del sistema, vamos a analizar el funcionamiento y la configuración de cada una de estas etapas.

GESTORES DE ARRANQUE

Los gestores de arranque son el primer componente que se ejecuta cuando encendemos el hardware. Su misión principal es la de cargar el núcleo del sistema en memoria. Hoy en día, los gestores de arranque nos ofrecen otras posibilidades, como la de presentar un menú en pantalla para acceder a diferentes sistemas operativos que puedan estar instalados en el sistema. Los gestores de arranque, además, proveen métodos que nos permiten comunicarnos con el kernel justo en su momento de ejecución, para poder definir o modificar la forma en que se iniciará el sistema. En el ámbito de Linux en plataformas x86, hay dos gestores de arranque que reinan: **LILLO** y **GRUB**. Veamos las características de cada uno.

Configuración de LILLO

LILLO (*Linux Loader*) es el gestor de arranque clásico de GNU/Linux. Viene incluido en casi todas las distribuciones desde tiempos memorables, aunque muchas de

III ¿CUÁL ES MEJOR?

Muchos administradores pelean por fanatismos a favor de LILLO o GRUB. La cuestión es muy sencilla: los dos realizan las mismas tareas. La diferencia es que GRUB ofrece más posibilidades de configuración. Pero, así como brinda más posibilidades, es más complicado. Está en nosotros decidir cuál instalar, pero si sólo tenemos uno o dos sistemas, lo más recomendable es optar por LILLO.

ellas ahora están optando por pasarse a GRUB, ya que es mucho más robusto y completo. De todas formas, vamos a aprender a configurar este sistema que aún habita en millones de computadoras en el mundo.

LILO se configura desde un archivo ubicado en el directorio `/etc/lilo.conf` que, al igual que todos los otros archivos de configuración, es un archivo de texto que puede ser editado con cualquier editor. Una vez que lo hayamos modificado, debemos escribir el comando `lilo`. La función de este comando es la de leer la configuración definida por el usuario y hacer “reales” esos datos. Esto quiere decir que, en la MBR o en el sector de arranque de la partición definida, se escribirán los datos definidos por el usuario.

Los archivos de configuración de LILO poseen una estructura muy sencilla. Veamos uno a modo de ejemplo:

```
boot = /dev/hda
prompt
timeout = 50
default = Linux
map = /boot/map
install = /boot/boot.b

image = /boot/bzImage
label = Linux
root = /dev/hda2

other = /dev/hda1
label = dos
table = /dev/hda
```

Claramente, distinguimos tres partes dentro de estos archivos. En la primera se definen los valores generales de configuración. Podemos ver como primera línea la configuración de dónde será instalado LILO. Si lo instalamos en la MBR (tal cual está definido en nuestro ejemplo), entonces LILO será el que se ocupe de cargar no



MBR

MBR (*Master Boot Record*) es el primer sector del disco duro, y se lo utiliza para alojar el sector de arranque del sistema operativo. En este capítulo veremos cómo configurar, desde Linux, las opciones relacionadas con este sector.

sólo Linux, sino el resto de los sistemas operativos de nuestra computadora. Esto tiene beneficios y contras. Como beneficios, podemos mencionar que la configuración de todo el arranque de nuestra PC será gestionada por un programa de GNU/Linux, lo cual puede inspirar cierta seguridad frente a virus y otros componentes malignos. Como contras, nos encontramos con que no todos los sistemas operativos son amigos de LILO (como Windows 2003) y requieren tener su propio gestor de arranque. Por lo tanto, se recomienda instalar al menos una versión de LILO en el sector de arranque de la partición de GNU/Linux. Esto nos permitirá instalar, en la MBR, otro gestor de arranque que sea compatible tanto con Windows como con GNU/Linux. Si sólo ejecutamos Linux en esa computadora, entonces podemos optar por instalar LILO en la MBR del disco duro.

Comencemos ahora a analizar, línea por línea, lo que está definido en nuestra sección de configuración general del archivo `/etc/lilo.conf`:

- **prompt**: con esta directiva le indicamos a LILO que presente un prompt en pantalla, dando la posibilidad al usuario de elegir el sistema operativo a iniciar.
- **timeout = [numero]**: aquí podemos definir el tiempo máximo de espera a que el usuario haga una selección. Este tiempo debe ser expresado en décimas de segundo.
- **default = [etiqueta]**: aquí definimos cuál será el perfil de iniciación que se ejecutará por defecto, si el usuario no realiza ninguna elección en particular durante el menú.

Luego vienen definiciones de ubicaciones de archivos específicos del núcleo Linux, y entonces sí se comienzan a definir todos los sistemas operativos que están instalados en el sistema y que podrán ser cargados por LILO.

Los sistemas GNU/Linux requieren tres directivas para ser configurados:

- **label = [cadena]**: para definir una etiqueta que haga referencia a ese sistema.
- **image = [ruta]**: con la ruta completa al núcleo Linux.
- **root = [ruta]**: con la ruta completa a la partición en la cual está instalado nuestro sistema operativo.

Los sistemas no-GNU/Linux son aún mucho más sencillos de configurar. Sólo debemos definir lo siguiente:

- **other = [ruta]**: la ruta completa a la partición donde está actualmente instalado nuestro “otro” sistema operativo.
- **label = [cadena]**: para definir una etiqueta que haga referencia al sistema operativo.
- **table = [ruta]**: con la ruta completa al dispositivo de disco duro.

Configuración de GRUB

GRUB (*Grand Unified BootLoader*) es un gestor de arranque mucho más avanzado que LILO, ya que provee un set de herramientas mucho más completo. Esto, para muchos, quiere decir que la configuración será un proceso más complejo. Como administradores de sistemas, sólo nos interesa saber cómo funciona y cómo se configura para poder afinar un poco más nuestras computadoras.


 <h1 style="margin: 0;">GNU GRUB</h1>	
<p>GNU GRUB</p> <ul style="list-style-type: none"> • Home • Introduction • GRUB 2 • GRUB Legacy • Documentation • Development <p>Support</p> <ul style="list-style-type: none"> • FAQ • Bug Reports <p>Links</p> <ul style="list-style-type: none"> • Link Explanations • PUPA • Etherboot • Original Site • GRUB/98 • French Translation of GRUB manual. • A GRUB logo by Karol Krenski 	<p style="text-align: center;">Introduction</p> <p>GNU GRUB is a Multiboot boot loader. It was derived from GRUB, GRand Unified Bootloader, which was originally designed and implemented by Erich Stefan Boleyn.</p> <p>Briefly, <i>boot loader</i> is the first software program that runs when a computer starts. It is responsible for loading and transferring control to the operating system <i>kernel</i> software (such as the Hurd or the Linux). The kernel, in turn, initializes the rest of the operating system (e.g. GNU).</p> <p>Currently, a new breed GRUB is under development, GRUB 2. This means what was known as GRUB (i.e. version 0.9x) has become GRUB Legacy.</p> <p style="text-align: center;">GRUB Legacy</p> <p>GRUB Legacy is not actively developed any longer. Only bugfixes will be made so that we can continue using GRUB Legacy until GRUB 2 becomes stable enough. If you want more features in GRUB, it is a waste of time to work on GRUB Legacy, because we never accept any new feature. Instead, it is better to take part in the development of GRUB 2.</p> <p>GRUB Legacy follows these requirements below:</p> <ul style="list-style-type: none"> • Compliant with the Multiboot Specification • Basic functions are easy for an end-user to use. • Rich functionality for OS experts/designers

Figura 2. En el sitio oficial de GRUB (www.gnu.org/software/grub/) encontraremos más información sobre la configuración de este sistema.

La configuración de lo que se ve en el menú del sistema se realiza por medio del archivo `/boot/grub/menu.lst`. En este archivo se define lo mismo que se define en el archivo `/etc/lilo.conf` que utiliza LILO, con ligeras diferencias de sintaxis.

Veamos cómo sería un archivo de ejemplo de GRUB, utilizando como base el sistema configurado en el archivo `/etc/lilo.conf`.



¡NO NOS OLVIDEMOS DE USAR EL COMANDO LILO!

Mucha gente se olvida de escribir el comando **lilo** una vez que modificó el archivo **lilo.conf**. Esto resulta ser de vital importancia. Si omitimos este paso, el sistema no sufrirá ningún cambio y las cosas seguirán funcionando como estaban.

```

default 0
timeout 8
gfxmenu (hd0,2)/boot/message

title Linux
    kernel (hd0,2)/boot/vmlinuz root=/dev/hda2
    initrd (hd0,2)/boot/initrd

title Windows
    root (hd0,1)
    chainloader +1

```

Como vemos, tenemos la misma estructura de tres partes que en **lilo.conf**. Primero, definimos opciones generales del sistema. Luego, nos encontramos con una configuración de un perfil de Linux y otra para Windows.

Analicemos la sección general:

- **default [#]:** con esto se le indica a GRUB que, si el usuario no realiza ninguna selección, entonces inicie por defecto el primer sistema operativo configurado. En nuestro caso, iniciará por defecto el sistema Linux.
- **timeout [#]:** aquí definimos cuál será el perfil de iniciación que será ejecutado por defecto, si el usuario no realiza ninguna elección en particular durante el menú.
- **gfxmenu (hd0,2)/boot/message:** aquí definimos la ubicación de un mensaje de bienvenida en el menú principal.

Ahora podemos comenzar a configurar cada uno de los sistemas operativos instalados en nuestro equipo. Veamos la configuración del sistema GNU/Linux:

- **title Linux:** esta directiva es análoga a la directiva **label** en el archivo **/etc/lilo.conf**.
- **kernel (hd0,2)/boot/vmlinuz root=/dev/hda2:** aquí definimos la ubicación del archivo del kernel. A diferencia del archivo **/etc/lilo.conf**, en este caso hacemos

III A DIFERENCIA DE LILO...

Una diferencia importante es que GRUB no requiere que el usuario ejecute un comando en particular luego de modificar el archivo de configuración. El sistema lo leerá automáticamente cuando la computadora sea reiniciada.

referencia a las particiones de forma (hd#, #), en donde el primer número indica el número de dispositivo, y el segundo, el número de partición.

- **initrd (hd0,2)/boot/initrd**: aquí especificamos la ubicación del archivo **initrd** de nuestro núcleo, de la misma forma que lo hicimos anteriormente.

Veamos cómo es la configuración de un sistema operativo no-GNU/Linux:

- **title Windows**: esta directiva es análoga a la directiva **label** en el archivo **/etc/lilo.conf**.
- **root (hd0,1)**: definición de la partición en la cual está instalado el sistema operativo.
- **chainloader +1**: indicamos que el sistema operativo posee su propio gestor de arranque instalado en la partición anteriormente definida.

NIVELES DE EJECUCIÓN

Como mencionamos anteriormente, los niveles de ejecución son configuraciones de perfiles del sistema. Cada nivel de ejecución posee su propio set de scripts que definen qué procesos se ejecutarán cuando éste se inicia y qué procesos se detendrán cuando muere. Por ejemplo, el nivel de ejecución 0 (apagado) posee definido un set de scripts que se encarga de cerrar todos los procesos y los descriptores de archivos abiertos, desmontar las particiones y, por último, apagar el sistema. El nivel de ejecución 3 (multiusuario) presenta un set de scripts que se ocupa de cargar en memoria todos los servicios necesarios para que el sistema operativo pueda ser operado por muchos usuarios simultáneos conectados local o remotamente.

Cada vez que iniciamos nuestro sistema, se inicializa un nivel de ejecución. Esta inicialización está a cargo del proceso **init**, el primero que se ejecuta en espacio de usuario. Ahora bien, esto no quiere decir que el nivel configurado por defecto será el nivel que se utilice permanentemente en el sistema. En cualquier momento podemos cambiar de nivel de ejecución con el comando **init**, seguido de un número de nivel de ejecución. Por ejemplo, **init 6** sería análogo al comando **reboot**, ya que el nivel 6 es el nivel de reiniciación del sistema. En realidad, el comando **reboot** no hace más que ejecutar el comando del ejemplo anterior para reiniciar el sistema.

Los niveles de ejecución son siete en total, y generalmente están configurados del siguiente modo:

- 0 - Sistema apagado.
- 1 - Sistema monousuario.
- 2 - Sistema multiusuario sin NFS.
- 3 - Sistema multiusuario con soporte de red.

- 4 - Sin usar.
- 5 - Modo multiusuario con operación gráfica.
- 6 - Reiniciar el sistema.

El proceso **init** basa su ejecución en un archivo localizado en **/etc/inittab**, en el cual se encuentran detallados los diferentes niveles de ejecución (*run levels*) y sus procesos.

A continuación, veremos un archivo **/etc/inittab** de ejemplo. Cabe destacar que los comentarios han sido traducidos a nuestro idioma.

```
#
# inittab      Este archivo describe como el proceso init debe configurar
#             el sistema en un nivel de ejecucion determinado.
#
# Autor:      Miquel van Smoorenburg, <miquels@drinkel.nl.mugnet.org>
#             Modificado para RHS Linux por Marc Ewing y Donnie Barnes
#
# runlevels predeterminados. Los runlevels que utiliza RHS son:
# 0 - Parado (no configure este como predeterminado)
# 1 - Modo Monousuario
# 2 - Multiusuario, sin NFS (Lo mismo que el 3 si no tiene una red)
# 3 - Modo multiusuario completo
# 4 - sin usar
# 5 - X11
# 6 - reiniciar (no configure este como predeterminado)
#
id:3:initdefault:

# Inicializacion del sistema

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

10:0:wait:/etc/rc.d/rc 0
11:1:wait:/etc/rc.d/rc 1
12:2:wait:/etc/rc.d/rc 2
13:3:wait:/etc/rc.d/rc 3
14:4:wait:/etc/rc.d/rc 4
15:5:wait:/etc/rc.d/rc 5
16:6:wait:/etc/rc.d/rc 6
```

```

# Cosas que se ejecutan en cada nivel de ejecucion.
ud::once:/sbin/update

# Atrapar el CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now

# Cuando nuestro UPS nos dice que la alimentacion ha fallado, asumimos que
# tenemos unos minutos de alimentacion mas. Registramos
un cierre de sistema
# de 2 minutos por ahora.
# Esto obviamente asume que usted tiene un UPS conectado y trabajando
# correctamente.
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"

# Si la alimentacion se reestablecio antes de que se ejecute
shutdown, cancelamos.
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"

# Ejecutar gettys en los niveles de ejecucion estandars.
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

# Ejecutar xdm en el runlevel 5
# xdm ahora un servicio separado.
x:5:respawn:/etc/X11/prefdm -nodaemon

```

La primera línea sin comentar le indica a **init** que el nivel de ejecución configurado por defecto es el 3. Al igual que la primera, las demás líneas tienen todas el mismo formato: valores separados por signos “:”.

El primero es el valor identificatorio del proceso (utilizado internamente por **init**), que deberá ser un valor numérico. El segundo es la secuencia de números correspondientes a los diferentes niveles en los que se quiere ejecutar ese proceso. El tercer parámetro es la acción a realizar con dicho proceso. Por ejemplo, si en ese campo se encuentra la cadena **wait**, **init** ejecuta el proceso configurado y espera a que termine. El último valor de la línea es el proceso a ejecutar en cuestión.

Las acciones que pueden realizarse sobre los procesos son las siguientes:

- **respawn**: si el proceso no existe, lo crea y no espera a que termine (continúa con la ejecución del **inittab**). Si el proceso existe, no hace nada. Cuando el proceso muere, se vuelve a ejecutar.
- **wait**: ejecuta un proceso y espera a que termine.
- **once**: crea el proceso y continúa con la lectura del **inittab**. Cuando muere, no vuelve a ser ejecutado.
- **boot**: crea el proceso solamente durante la etapa de booteo. No espera a que termine, y cuando muere, no vuelve a ser ejecutado.
- **bootwait**: se crea el proceso cuando se realiza el cambio entre modo monousuario y modo multiusuario. Si el nivel de ejecución configurado por defecto es el 2, entonces se creará justo después del booteo.
- **powerfail**: ejecuta el proceso solamente cuando **init** recibe una señal de falla de alimentación (**SIGPWR**).
- **powerwait**: ejecuta el proceso solamente cuando **init** recibe una señal de falla de alimentación y espera a que concluya antes de continuar con los demás procesos.
- **off**: si el proceso se está ejecutando, envía una señal de advertencia (**SIGTERM**) y espera 20 segundos antes de enviarle la señal de muerte (**SIGKILL**). En caso de que el proceso no exista, esta acción es ignorada.
- **initdefault**: define el nivel de ejecución en que se iniciará el sistema por defecto.
- **sysinit**: los procesos que lleven esta acción se ejecutarán antes de que **init** intente acceder a la consola para iniciar la sesión de **login**.

Personalización de los servicios de arranque

Hasta ahora sabemos cómo configurar el gestor de inicialización del núcleo y el selector del nivel de ejecución inicial (**init**). El próximo paso será, entonces, configurar los servicios que se ejecutarán en cada nivel de ejecución. Obviamente, cuando instalamos una distribución de GNU/Linux, se generan las configuraciones correspondientes a los siete niveles de ejecución. La cuestión es que esta configuración no siempre es acorde a nuestras necesidades. Por ejemplo, un problema típico es que cuando instalamos una distribución de las “hogareñas”, se ejecutan automáticamente servicios como Samba, Apache o Postfix (que son servidores de alto rango) sin que nosotros lo hayamos pedido. Esto se traduce en una pérdida de performance para nuestro sistema y un aumento en el porcentaje de inseguridad general (recordemos, cuantos más servicios abiertos, mayores serán los problemas de seguridad que deberemos afrontar).

Una de las maneras que disponemos para gestionar los servicios de nuestro sistema es hacer uso de los programas de configuración que acompañan a las distribuciones. Por ejemplo, la famosa distribución Mandrake incluye una sección dedicada a los

servicios en su Centro de Control. Los usuarios de SuSE, en cambio, deberán usar YaST para configurar los servicios correspondientes a su distribución. En RedHat, los usuarios disponen de un comando llamado **setup** que permite, entre otras cosas, configurar los servicios del arranque del sistema. En conclusión, siempre encontraremos algún gestor de servicios en todas las distribuciones. Pero... ¿cómo funcionan ellos? ¿Qué es lo que hacen realmente? Vamos a analizarlo.

En el directorio **/etc/rc.d** de nuestra distribución encontraremos diferentes subdirectorios, cada uno de ellos dedicados a un nivel de ejecución diferente. De esta forma, el directorio **/etc/rc.d/rc3.d** contiene información de los servicios del nivel de ejecución 3; el directorio **/etc/rc.d/rc5.d** contiene información de los servicios del nivel de ejecución 5, y así sucesivamente.

Allí dentro lo que encontramos son básicamente enlaces simbólicos hacia scripts ubicados en el directorio **/etc/rc.d/init.d** (o **/etc/init.d** en algunas distribuciones). Estos enlaces poseen nombres raros a simple vista pero que, en realidad, revelan gran cantidad de información sobre el servicio.

Como mencionamos anteriormente, el nivel de ejecución 3 es un nivel que se utiliza para iniciar el sistema (y no para apagar, como el 6 o el 0), entonces, obviamente, incluirá enlaces a scripts que ejecutan servicios, módulos y otras aplicaciones. Por esta razón, notamos que gran parte de los archivos comienzan con una letra **S** (mayúscula), y otras, con una **K**. Los enlaces que comienzan con **S** se encargan de ejecutar programas. Los enlaces que comienzan con **K** se encargan de matar programas (pensemos que los enlaces ubicados en **rc6.d** serán casi en su totalidad archivos que comienzan con **K**). Luego, se especifica un número que corresponde al orden de ejecución. Por último, una palabra nos indica el servicio que ejecuta dicho enlace.

Ahora bien, dijimos también que esos enlaces apuntan hacia un directorio ubicado en **/etc/rc.d/init.d** (o **/etc/init.d**) que incluye scripts que lanzan o detienen servicios. Es importante que sepamos que estos scripts pueden también ser usados desde una línea de comandos. Todos ellos poseen tres parámetros básicos que nos permitirán lanzar, matar o relanzar dicho servicio.



CHINO BÁSICO

Queda claro que los programas que anteriormente mencionábamos nos permiten gestionar los servicios de forma sencilla (el Centro de Control de Mandrake o el setup de Red Hat), o sea, no hacen más que crear o eliminar archivos de enlaces en estos directorios. Por consiguiente, una forma de eliminar servicios del nivel 3 es borrar directamente estos archivos de enlaces.

El siguiente es un ejemplo válido para iniciar el servidor NFS:

```
/etc/rc.d/init.d/nfs start
```

Si quisiéramos detenerlo, sólo tendríamos que ejecutar:

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

Y para relanzarlo (matar y volver a ejecutar):

```
/etc/rc.d/init.d/nfs restart
```

El archivo de autoejecución de GNU/Linux

Muchos usuarios se encuentran con la necesidad de ejecutar un comando cada vez que encienden la computadora. Por ejemplo, al utilizar el comando **ifconfig** para configurar la placa de red, notamos que cuando apagamos la computadora y volvemos a encenderla, la configuración se perdió. Esto se debe a que el comando no guarda registro de la configuración en ningún archivo.

Como solución, muchas distribuciones incluyen un archivo en **/etc/rc.d/rc.local**. Este archivo no es más que un simple script que se ejecuta justo después de que todos los servicios del nivel de ejecución definido se han ejecutado, y justo antes de que se presente la pantalla de login al usuario. Dentro de ese script (que generalmente las distribuciones lo incluyen vacío para que el usuario lo complete con lo que necesita), podemos incluir cualquier comando que se ejecute cuando encendemos nuestro sistema.

Existen otras distribuciones que ya no incluyen ese archivo y que ahora ofrecen un directorio llamado **/etc/rc.boot**, en el cual todos los scripts que pongamos serán ejecutados al iniciar el sistema, exactamente en el mismo momento que el **rc.local** (éste es un directorio en el cual debemos crear archivos de scripts que ejecuten los comandos que deseamos).

PERSONALIZACIÓN DEL INICIO EN MODO TEXTO

Una vez que se han cargado todos los servicios del nivel de ejecución correspondiente, podemos también, como administradores, personalizar la pantalla de login (bienvenida al sistema) en modo texto. El archivo que se encarga de almacenar el mensaje de bienvenida (que se imprime en pantalla justo antes de la palabra **login**)

es `/etc/issue`. Éste es un simple archivo de texto que puede ser editado a gusto y que se mostrará a todos los que generen conexiones locales. Existe otro archivo que nos permite especificar un mensaje de bienvenida para todas las personas que ingresan en nuestro sistema de forma remota (por ejemplo, vía telnet o ssh); se trata de `/etc/issue.net`, y se utiliza generalmente para informar a la persona que inició la conexión sobre la utilidad de dicho servidor.

Ahora bien, existe otro archivo que nos permite mostrar un mensaje en pantalla justo después de que nuestros usuarios se registraron en el sistema (esto es, escribieron un nombre de usuario y clave válidos), y ese archivo es `/etc/motd`. Su nombre proviene de las siglas de la frase “*Message of the Day*” (mensaje del día) y se lo utiliza para distribuir información que sólo queremos que llegue a los usuarios reales del sistema.

```

tty2 87x23
Trying 10.0.0.10...
Connected to 10.0.0.10.
Escape character is '^]'.
Password:
Last login: Sun Aug 15 19:13:38 2004 from 10.0.0.1 on pts/2
Linux Alternarama 2.4.18-686 #1 Sun Apr 14 11:32:47 EST 2002 i686 GNU/Linux

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
hfarena@Alternarama:~$ █
    
```

Figura 3. El archivo `/etc/motd` se muestra justo luego de que el usuario ingresó nombre de usuario y contraseña.

Configuración del intérprete de comandos de cada usuario

Cuando el usuario ingresa en el sistema, el gestor de registración (el programa login) se encarga de ejecutar el intérprete de comandos configurado para dicho usuario. Esto se define en el archivo `/etc/passwd`, y es el último parámetro de cada registro de usuario. En el ejemplo, el intérprete de comandos para el usuario Facundo es el famoso **BASH**.

```
facundo:x:500:100:facundo:/home/facundo:/bin/bash
```

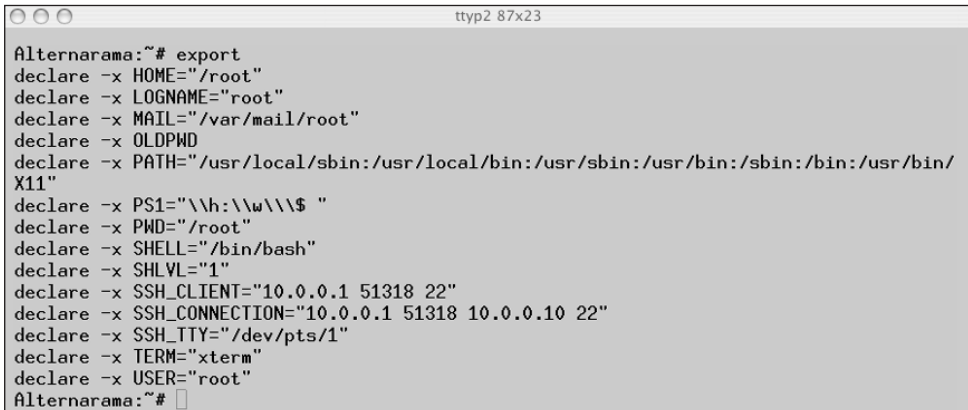
En este ejemplo, el intérprete de comandos del usuario Roberto es el famoso gestor de archivos **Midnight Commander**.

```
roberto:x:508:100::/home/roberto:/usr/bin/mc
```

Resumiendo, el último parámetro de cada registro de usuario en el archivo **password** corresponde al primer programa que se cargará justo después que el usuario ingresa nombre de usuario y clave válidos. El programa más popular es el intérprete de comandos **BASH** (*Bourne Again SHell*); éste lee un archivo de configuración ubicado en el directorio personal de cada usuario, para generar un entorno de trabajo. ¿Qué es un entorno de trabajo? Sencillo: una configuración de la línea de comandos y un set de variables de entorno útiles para dicho usuario. Este archivo se llama **.bashrc** (es un archivo oculto, ya que comienza con un punto), y, como mencionamos anteriormente, está ubicado en el directorio personal de cada usuario. Es posible editarlo con cualquier editor de textos y crear o eliminar configuraciones de variables de entorno según lo deseado. Para definir variables de entorno nuevas, simplemente usamos:

```
export [nombredelavariabile]= [valor]
```

Existe un archivo que nos permite definir variables de entorno válidas para todos los usuarios de BASH, cuya estructura es exactamente igual al recién visto, y se ubica en **/etc/bash.bashrc**.



```
Alternarama:~# export
declare -x HOME="/root"
declare -x LOGNAME="root"
declare -x MAIL="/var/mail/root"
declare -x OLDPWD
declare -x PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/bin/X11"
declare -x PS1="\h:\w\$ "
declare -x PWD="/root"
declare -x SHELL="/bin/bash"
declare -x SHLVL="1"
declare -x SSH_CLIENT="10.0.0.1 51318 22"
declare -x SSH_CONNECTION="10.0.0.1 51318 10.0.0.10 22"
declare -x SSH_TTY="/dev/pts/1"
declare -x TERM="xterm"
declare -x USER="root"
Alternarama:~#
```

Figura 4. El comando *export* nos muestra un listado de todas las variables de entorno definidas en el sistema.

RESUMEN

En este capítulo hemos comenzado mediante una completa descripción de cómo funciona el proceso de inicialización de nuestra computadora, para luego analizar, paso por paso, de qué manera personalizarlo para que se adecue a nuestras necesidades. Más adelante volveremos a tocar algunos puntos relacionados con la inicialización.



TEST DE AUTOEVALUACIÓN

1 ¿Qué es un gestor de arranque?

2 ¿Cuáles son los gestores de arranque más populares en GNU/Linux?



3 ¿Qué diferencia hay entre la ejecución en espacio de núcleo y espacio de usuario?

4 ¿Qué es un nivel de ejecución?

5 ¿Para qué se utiliza normalmente el nivel de ejecución 5?

6 ¿En qué archivos se define el nivel de ejecución por defecto?

7 ¿Para qué sirve el archivo rc.local?

8 ¿Cuál es el programa más utilizado como primer proceso de un usuario?

9 ¿En qué archivo se define el intérprete de comandos de cada usuario?

10 Al ejecutar el comando halt, ¿a qué nivel de ejecución estamos cambiando?