10. Núcleo

10.1. Introducción

Introducción

La configuración del kernel es un proceso muy delicado y crítico que no se suele hacer muy a menudo y cuando se hace es con todas las precauciones y medidas de seguridad posibles, como un sistema de arranque suplementario (en caso de Linux con un disquete) y una copia de seguridad.

Esta operación merecerá la pena ser efectuada en determinadas ocasiones, que podemos enumerar así:

1. Instalación de un sistema nuevo. Esta situación es obvia, a no ser que el sistema venga montado ya con un sistema UNIX reciente que no deba ser renovado por razones de eficiencia.
2. Instalación de algún “driver” de dispositivo. En la mayoría de los casos, sobre todo en sistemas UNIX antiguos, instalar un nuevo driver requiere parar el sistema y efectuar esa instalación, ya que esto requiere reconfigurar tablas internas del kernel, que deberá ser recompilado. En sistemas más modernos, en el caso del Linux, a partir de la versión 2, existen los módulos cargables, en los que no es necesario parar el núcleo para añadirle nuevas funcionalidades, además tiene la ventaja de que el kernel con módulos es más pequeño, ahorrándose memoria, y que los módulos sólo se cargan cuando se necesitan. Otros autores opinan que introducir módulos en plena ejecución es como “realizar una operación cerebral a un individuo que está manejando maquinaria pesada”.
3. Ajustar las tablas del núcleo. El kernel por eficiencia trabaja con un espacio constante de memoria, teniendo todas sus tablas (número de procesos, segmentos de memoria, dispositivos, etc.) un tamaño fijo. Si queremos ajustar el tamaño de esas tablas (compromiso entre gasto de memoria y cantidad de recursos que pueden ser usados) no nos quedará otro remedio que reconfigurar y recompilar el kernel.

Otra posible causa es porque nuestro kernel se ha quedado anticuado. Sólo merecerá la pena cambiar este núcleo cuando exista una gran diferencia de versión en el número grande (ver capítulo anterior). Si sólo queremos cambiar de revisión existe otro método que es aplicar parches (patch) a partes del núcleo.

10.2. Actualización

Actualizando el núcleo

Para actualizar el núcleo de Linux es necesario que tengamos el código fuente del sistema en el directorio /usr/scr/linux. Si esto no es así, lo instalaremos a través de un paquete RPM (o DEB) como hemos visto en el capítulo anterior.

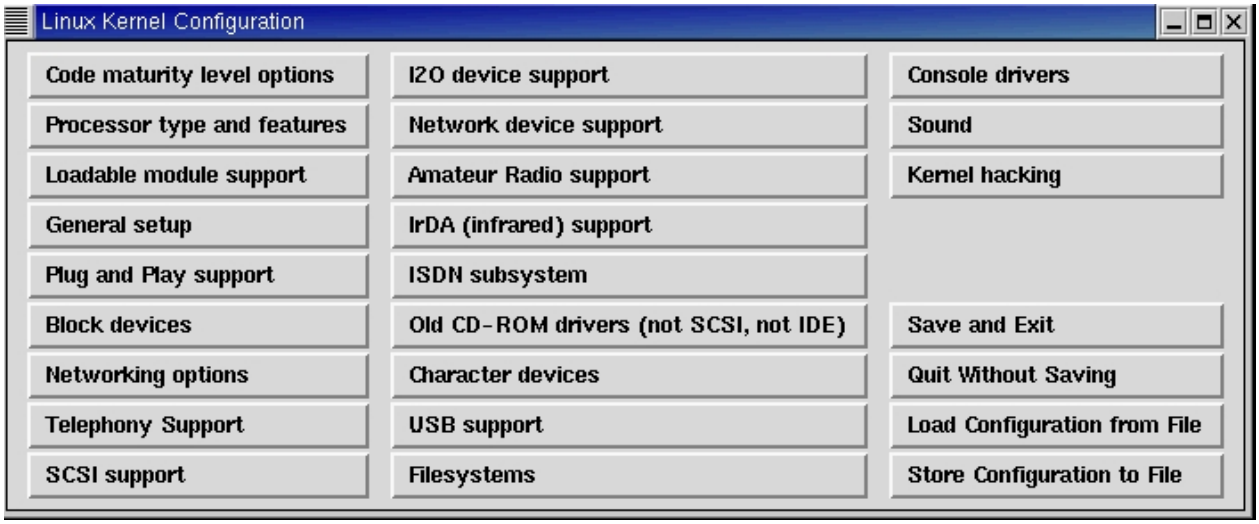
Antes de actualizar el kernel es obvio que tenemos que saber que versión del kernel estamos usando, para ello tenemos el comando uname (opción –r) que nos dará información sobre el sistema en general y sobre la versión en particular.

```
#uname -r
2.2.16-22
```

Después, tendremos que buscar en algún sitio ftp o desde un CD, una nueva versión de las fuentes del núcleo. Estas fuentes pueden estar de la forma clásica comprimidas y empaquetadas (tar.gz) o en la moderna en forma RPM. Una dirección (buscar el fichero adecuado dentro) que contiene el último código fuente es: `ftp://sunsite.rediris.es/pub/linux/kernel/sources` y otra en forma de paquete RPM: `ftp://ftp.rpmfind.net/linux/redhat`.

Otra posible opción, si no está instalado este código fuente del kernel, es la de la propia distribución, siempre existe un CD especial con los códigos fuente de diversos paquetes.

Una vez que tengamos instalado el código fuente podemos proceder de tres maneras dependiendo del entorno que tengamos: instalación basada en texto, instalación con menús o instalación gráfica (ver siguiente figura). En el primer caso habrá que hacer desde el directorio de las fuentes (`/usr/src/linux`) `make config`, en el segundo `make menuconfig` y en el tercero `make xconfig`.



De forma más o menos amigable (la ventaja de las dos últimas es que sólo responderemos a las partes que deseamos cambiar), tendremos que responder opcionalmente a las preguntas que nos plantea la aplicación de configuración del kernel. Para cada pregunta que tengamos que responder existirá un botón de ayuda para auxiliarnos en la decisión y tres opciones de respuesta: y para sí, n para no y m para módulo (ver siguiente sección). Debemos tener en cuenta que cuando pulsamos y esa parte pertenecerá al núcleo cada vez que se inicia el sistema, si decimos n, no estará cargada y si decimos m, se cargará cuando sea necesaria a través de un módulo. Si nos hemos atrevido a configurar el núcleo es porque tenemos suficiente experiencia y madurez para saber que es lo que debe estar cargado o no en el mismo y cual es el balance adecuado entre rapidez en dar un recurso y espacio ocupado en el kernel. Las preguntas serán:

- Grado de madurez del código. Esta parte servirá de indicativo para las siguientes, refiriéndose, por ejemplo, a la instalación de un determinado driver que podría estar incompleto o no, ser seguro o no, etc. El sistema en este caso puede preguntar antes. Se recomienda poner no como elección general.
- Soporte para módulos. Si se quiere usar un kernel que soporte módulos o uno monolítico. Esta parte también tendrá consecuencias sobre las siguientes, ya que si se escoge no, no se podrá elegir la opción m en las siguientes cuestiones. Se recomienda la primera opción.
- Características y tipo de procesador. Indicar si se tiene coprocesador matemático o va a estar emulado por software y si hay un conjunto de procesadores simétrico o no.
- Soporte para pnp. Si queremos soportar dispostivos plug and play.
- Configuración general. Indicaremos si queremos soporte para grandes cantidades de memoria (hay que configurar el lilo), para la red y para el acceso a PCI. Para comunicaciones de tipo System V entre proceso y para el formato de los ejecutables.
- Dispositivos de bloque y carácter. Soporte para distintos tipos de discos, como disquetes, cd-rom ATAPI, discos duros IDE/MFMRLL. Soporte para consolas virtuales y terminales serie y ratones.
- Opciones y soporte de red. Si queremos instalar soporte para sockets, se recomienda decir que sí si se tiene red.
- Soporte de SCSI, telefónico, de radioaficionado. Soporte para distintos dispositivos.
- Soporte de diversos dispositivos: IrDA, USB, I2O, Sonido, etc.
- Tipos de sistemas de ficheros. Nos preguntará si queremos instalar el sistema de cuotas de disco y el automounter, además de los tipos de sistemas de ficheros: amiga, apple, DOS FAT, VFAT, Minix, ISO 9660, Joliet, OS/2, etc. El que nunca deberá faltar es el segundo sistema extendido ext2 así como el soporte de /proc, que es un directorio no residente en disco con información del sistema.

Si estamos interesados en alguna opción más específica, antes de compilar el kernel, siempre se pueden revisar las macros de compilación para poder hacer algún cambio manualmente. Esto requiere unos conocimientos más o menos profundos del sistema.

Una vez que hemos configurado nuestro kernel (se habrán hecho los cambios oportunos en los scripsts a utilizar [existe un directorio de macros cambiadas llamado scripts]), deberemos pasar a compilarlo, esto se hace con las macros de make que aparecen a continuación:

```
mak dep
make clean
```

```
make zImage
```

Con el primer paso analizamos las dependencias de compilación del núcleo, con el segundo limpiamos todos los ficheros objeto que no sirven (con `make mrproper` se puede hacer más profundamente) y con el tercero compilamos ya el núcleo. Dependiendo del sistema que tengamos: procesador y memoria, ésta tarea puede llevar más o menos tiempo (minutos).

Si todo ha ido bien (no siempre es así -por no decir casi nunca-), y no ha habido errores de ejecución en `make`, en el directorio de fuentes donde está nuestra arquitectura (en el caso de Intel i386) `/usr/src/linux/arch/i386/boot/` veremos el fichero `zImage` que es el núcleo comprimido que se deberá cargar en el arranque.

Hasta ahora no hemos hecho nada irreversible, pero llegará el momento de botar con este nuevo núcleo (mover el fichero zImage al directorio /boot) y aquí viene el paso peligroso, ya que si lo hacemos alegremente, podemos conseguir que el sistema no arranque, por ello debemos garantizar que:

- Tenemos un disquete de arranque con el sistema viejo (ver capítulo de inicialización).
- O que hemos construido el nuevo núcleo en un disquete, esto se puede hacer ejecutando el make zImage con: make zdisk. Después meteremos el disquete, botaremos y si no arranca, siempre podremos sacarle y botar como siempre.
- O hemos cambiado el fichero de configuración de lilo `/etc/lilo.conf`, para que dependiendo del comando introducido podamos arrancar con el nuevo o el viejo (se recuerda que hay que ejecutar lilo para que tome el nuevo fichero). Para ello debemos renombrar el fichero vmlinuz de `/boot` por vmlinuz.old, por ejemplo, copiar el nuevo núcleo en vmlinuz y poner en el fichero de configuración lilo un doble arranque con algo parecido a (esto se hace automáticamente con make zlilo):

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

image=/boot/vmlinuz
label=linux
read-only
root=/dev/hda7
image=/boot/vmlinuz.old
label=anterior
read-only
root=/dev/hda7
```

El último paso que hay que realizar antes de botar el sistema es hacer los cambios en los módulos que hayamos incluido, ya que lo que hemos hecho hasta ahora es actualizar el kernel monolítico sin módulos (ver apartado de gestión de dispositivos).

10.3. Módulos

*H*emos dicho que para no cargar el núcleo del sistema con programas que se utilizan pocas veces, desde la versión 2.0 existen los módulos, que el kernel puede enlazar en tiempo de ejecución (no existen en todas las arquitecturas, por ejemplo para Alpha se debe usar un kernel monolítico). Es la forma preferida de usar algunos manejadores como los de dispositivos PCMCIA o cintas. Actualmente la tendencia es “modularizar” todos los controladores posibles, salvo aquellos estrictamente necesarios para arrancar, existiendo un demonio llamado kerneld que se encargará de cargarlos

y descargarlos por nosotros cuando el sistema los necesite.

Existen varios comandos relacionados con el uso de los módulos (todos estos comandos pueden ser ejecutados de forma gráfica desde el panel de control con el kernel configurator):

- **`lsmod`**. Nos dará una lista de los módulos que están cargados en memoria, indicando, el nombre, número de páginas de memoria usadas, cuantas veces ha sido usado y por quien (la misma información que `/proc/modules`). Un listado de **`lsmod`** aparece a continuación:

```
Module Size Used by
nls_cp437 3876 1 (autoclean)
ide-cd 23628 1 (autoclean)
lockd 31176 1 (autoclean)
sunrpc 52964 1 (autoclean) [lockd]
3c90x 22224 1 (autoclean)
awe_wave 158068 0
sb 33876 0
uart401 6224 0 [sb]
sound 57496 0 [awe_wave sb uart401]
soundlow 420 0 [sound]
soundcore 2596 7 [sb sound]
```

- **`insmod`**. Inserta un módulo especifico en el kernel.
- **`rmmod`**. Elimina un módulo del kernel.
- **`depmod`**. Crea un archivo de dependencias del que se sirve **`modprobe`**.
- **`modprobe`**. Carga módulos a partir de una lista generada por depmod.

Todos los cambios hechos con estos comandos o con la herramienta gráfica se aplican al fichero `/etc/modules.conf` que se encarga de leer el demonio **`kerneld`**. Un listado para los módulos anteriormente cargados se puede ver a continuación

```
alias eth0 3c90x
alias parport_lowlevel parport_pc
alias eth1 3c59x
alias sound-slot-0 sb
options sound dmabuf=1
options opl3 io=0x388
alias midi awe_wave
post-install awe_wave /bin/sfxload /etc/midi/GU11-ROM.SF2
options sb io=0x220 irq=5 dma=1 dma16=5 mpu_io=0x330
```

Para arrancar y parar el demonio de kernel debemos irnos a las macros de inicio y utilizar el `/etc/rc.d/init.d/kerneld start` o `stop` para realizar esta tarea.

Una vez que sabemos como manejar módulos, antes de botar el sistema debemos compilar los módulos que hemos configurado, al igual que hicimos con el resto del kernel. Para ello tenemos el `make modules` desde el directorio de fuentes `/usr/scr/linux` y después `make modules_install`. Como siempre deberemos asegurarnos de que los nuevos módulos funcionan, si no así, como en el caso del kernel monolítico, conviene hacer una copia de seguridad de estos módulos, para ello se deberá ir al directorio `/lib/modules` y renombrar el directorio que aparecerá con el nombre de la versión que estamos usando, por ejemplo: `mv /lib/modules/2.2.16-22 /lib/modules/2.2.16-22.trabajo`.

10.4. Parches

Parches

No siempre es necesario hacer una recompilación completa del núcleo para cambiar de versión, en muchos casos para pasar de una versión a otra cercana basta con aplicar los parches necesarios.

Las actualizaciones incrementales del núcleo se distribuyen como parches (no se debe tomar con sentido peyorativo el término “parche", no se trata de un remiendo, lo que hacemos al “parchear" es modificar directamente los fuentes del núcleo, incluyendo las variaciones que se hayan introducido). Por ejemplo, si se tiene la versión 2.2.16 y se ve que existe un parche “patch18.gz”, con ese fichero podrá actualizarse el sistema a la versión 2.2.18 (antes se deberá actualizar también a la 2.2.17).

Como siempre, antes de actuar se debería guardar una copia del árbol de directorios de las fuentes del núcleo actual haciendo “`make clean`” y luego “`tar cvfz antiguas-fuentes.tar.gz linux`” desde el directorio `/usr/src`.

Después procederemos a ejecutar el parche haciendo en `/usr/src`:

```
gzip -cd patch18.gz | patch -p0
```

donde 18 es la versión a actualizar.

Se deberán tener en cuenta los mensajes que se produzcan al ejecutar la línea por los posibles errores que se puedan ocurrir (si usamos la opción –s sólo saldrán los mensajes de error). Si nos actualizamos de varias versiones se puede ejecutar algo parecido a esto:

```
# for i in patch-2.0.2[1234567].gz; do
>zcat $i | patch -p0
>done
```

Si ha habido algún error se creará un fichero con extensión “`.rej`”. Habrá un fichero `config.in.rej` donde vendrán los parches que han originado problemas.