El núcleo (kernel) de Linux

¿Qué hace el kernel?

Probablemente haya escuchado a la gente hablar acerca de compilar el kernel o construir un kernel, pero ¿qué es exactamente el kernel y qué es lo que hace? El kernel es el centro de su ordenador. Es la base del sistema operativo completo. El kernel actúa como un puente entre el hardware y las aplicaciones. Esto significa que el kernel es (normalmente) la única pieza de software responsable de ordenar alrededor de los componentes de hardware de su ordenador. Es el kernel quien instruye a las unidades de discos duros para buscar un determinado flujo de datos. Es el kernel quien indica a su tarjeta de red que transmita cambios rápidos de voltaje. Así como es el kernel quien también escucha al hardware. Cuando la tarjeta de red detecta un ordenador remoto que envía información, reenvía esa información al kernel. Esto hace que el kernel sea la pieza más importante de software en su ordenador y la más compleja.

Trabajar con módulos

La complejidad de un kernel de Linux actual es asombrosa. El código fuente para el kernel pesa casi 400MB sin comprimir. Hay miles de desarrolladores, centenares de opciones, y si todo estuviese construido junto, el kernel pronto pasaría los 100MB de tamaño. Con el fin de mantener el tamaño del kernel bajo (así como la cantidad de RAM necesaria para el kernel), la mayoría de las opciones del kernel se construyen como módulos. Usted puede pensar en estos módulos como controladores (drivers) de dispositivo que pueden ser insertados o eliminados de un kernel en ejecución cuando uno quiera. Verdaderamente, muchos de ellos no son drivers de dispositivo en absoluto, pero contienen soporte para cosas tales como protocolos de red, medidas de seguridad e incluso sistemas de archivos. En resumen, casi cualquier pieza del kernel de Linux se puede construir como un módulo cargable.

Es importante darse cuenta de que Slackware manejará automáticamente la carga de la mayoría de los módulos. Cuando su sistema arranca, ***udevd***(8) se inicia y comienza a explorar el hardware de su sistema. Para cada dispositivo que encuentra, carga el módulo adecuado y crea un nodo de dispositivo en `/dev`. Esto suele significar que no tendrá que cargar ningún módulo para poder utilizar su ordenador, pero a veces esto es necesario.

Entonces, ¿qué módulos se cargan actualmente en su ordenador y cómo lo hacemos para que se carguen y descarguen? Afortunadamente tenemos un conjunto completo de herramientas para manejar esto. Como habrá adivinado, la herramienta para enumerar los módulos es ***lsmod***(8).

```
darkstar:~# lsmod
Module                  Size  Used by
nls_utf8                1952  1
cifs                  240600  2
i915                  168584  2
drm                   168128  3 i915
i2c_algo_bit            6468  1 i915
tun                    12740  1
```

```
... many more lines ommitted ...
```

Además de mostrarle qué módulos están cargados, muestra el tamaño de cada módulo y le dice qué otros módulos lo están utilizando.

Hay dos aplicaciones para cargar módulos: ***insmod***(8) y ***modprobe***(8). Ambos cargarán módulos y reportarán cualquier error (como cargar un módulo para un dispositivo que no está presente en su sistema), pero ***modprobe*** es preferido porque puede cargar cualquier dependencia del módulo. Usar cualquiera de las dos es sencillo.

```
darkstar:~# insmod ext3
darkstar:~# modprobe ext4
darkstar:~# lsmod | grep ext
ext4                  239928  1
jbd2                   59088  1 ext4
crc16                   1984  1 ext4
ext3                  139408  0
jbd                    48520  1 ext3
mbcache                 8068  2 ext4,ext3
```

La eliminación de módulos puede ser un proceso complicado, y una vez más tenemos dos programas para eliminarlos: ***rmmod***(8) y ***modprobe***. Para eliminar un módulo con modprobe, necesitará usar el argumento *-r*.

```
darkstar:~# rmmod ext3
darkstar:~# modprobe -r ext4
darkstar:~# lsmod | grep ext
```

Compilar un kernel y por qué hacerlo así

La mayoría de los usuarios de Slackware nunca necesitarán compilar un kernel. Los kernels gigante (huge) y genérico (generic) contienen prácticamente todo el soporte que necesitará.

Sin embargo, es posible que algunos usuarios necesiten compilar un kernel. Si el ordenador contiene hardware de última tecnología, un kernel más nuevo puede ofrecer mejor soporte. A veces puede estar disponible un parche del kernel que corrige un problema que está experimentando. En estos casos, una compilación del kernel está probablemente justificada. Los usuarios que simplemente quieren la última y la mejor versión o quienes crean que usar un kernel compilado personalizado les dará un mayor rendimiento, sin duda puede mejorar, pero es poco probable que realmente se noten cambios importantes.

Si aún piensa que compilar su propio kernel es algo que quiera o necesite hacer, esta sección debería guiarle a través de los numerosos pasos. Compilar e instalar un kernel no es tan difícil, pero hay una serie de errores que se pueden hacer a lo largo del camino, muchos de los cuales pueden evitar que el ordenador arranque y causar una gran frustración.

El primer paso es asegurarse de que tiene el código fuente del kernel instalado en su sistema. El paquete fuente del kernel está incluido en el conjunto de discos *“k”* en el instalador de Slackware, o puede descargar otra versión de http://www.kernel.org/. Tradicionalmente, la fuente del núcleo se

encuentra en /usr/src/linux, un enlace simbólico que apunta a la versión específica del kernel utilizada, pero esto no está de ninguna manera grabado en piedra. Puede colocar el código fuente del kernel prácticamente en cualquier lugar sin tener problemas.

```
darkstar:~# ls -l /usr/src
lrwxrwxrwx  1 root root   14 2009-07-22 19:59 linux -> linux-2.6.29.6/
drwxr-xr-x 23 root root 4096 2010-03-17 19:00 linux-2.6.29.6/
```

La parte más difícil de cualquier compilación del kernel es la configuración del kernel. Hay cientos de opciones, muchas de las cuales pueden ser compiladas opcionalmente en módulos. Esto significa que hay miles de formas de configurar un kernel. Afortunadamente, hay algunos trucos útiles que pueden mantenerle a salvo de encontrarse con demasiados problemas. El archivo de configuración del kernel es .config. Si usted es muy valiente, puede editar manualmente este archivo con un editor de texto, pero le recomiendo encarecidamente que utilice las herramientas integradas del kernel para manipular .config.

A menos que esté muy familiarizado con la configuración de kernels, usted debe siempre empezar con una sólida base de configuración y modificarla. Ésto le impide omitir una opción importante que podría obligarlo a compilar el kernel una y otra vez hasta que lo haga bien. Los mejores archivos .config del kernel para empezar son los utilizados por los kernels predeterminados de Slackware. Puede encontrarlo en su disco de instalación de Slackware o en su espejo (mirror) favorito en el directorio kernels/.

```
darkstar:~# mount /mnt/cdrom
darkstar:~# cd /mnt/cdrom/kernels
darkstar:/mnt/cdrom/kernels# ls
VERSIONS.TXT  huge.s/  generic.s/  speakup.s/
darkstar:/mnt/cdrom/kernels# ls genric.s
System.map.gz  bzImage  config
```

Puede reemplazar el archivo .config predeterminado fácilmente, copiando o descargando el archivo config' para el kernel que desea utilizar como base. En este caso estoy usando el kernel generic.s de Slackware recomendado para una base, pero es posible que desee utilizar el archivo de configuración huge.s. El kernel genérico construye más cosas como módulos y así crea una imagen del kernel más pequeña, pero por lo general requiere el uso de un initrd.

```
darkstar:/mnt/cdrom/kernels# cp generic.s/config /usr/src/linux/.config
```

El archivo del kernel de Slackware carece del *"punto"* mientras que el archivo del kernel lo incluye. Si olvida, o simplemente copia el config a /usr/src lo que sea, se usará el archivo .config que ya estaba presente en su lugar.

Si desea utilizar la configuración para el kernel que se está ejecutando actualmente como su base, usted puede ser capaz de encontrarlo en /proc/config.gz. Este es un fichero especial relacionado con el kernel que incluye la configuración completa del kernel en un formato comprimido, y requiere que su kernel haya sido construido para soportarlo.

```
darkstar:~# zcat /proc/config.gz > /usr/src/linux/.config
```

Ahora que hemos creado una configuración de base sólida, es hora de hacer cualquier cambio de configuración que queramos. Todo el proceso de compilación del kernel desde la configuración y la

compilación se realiza con el comando ***make***(1) y argumentos especiales para este. Cada argumento realiza una función diferente.

Si está actualizando a una versión más reciente del kernel, definitivamente querrá usar el argumento *oldconfig*. Esto pasará a través de su base `.config` y buscará los elementos que faltan, lo que normalmente indica que la nueva versión del kernel contiene opciones adicionales. Dado que las opciones se añaden en prácticamente todas las versiones del kernel, esto es generalmente algo bueno para hacer.

```
darkstar:/usr/src/linux# make oldconfig
scripts/kconfig/conf -o arch/x86/Kconfig
*
* Restart config...
*
*
* File systems
*
Second extended fs support (EXT2_FS) [M/n/y/?] m
  Ext2 extended attributes (EXT2_FS_XATTR) [N/y/?] n
  Ext2 execute in place support (EXT2_FS_XIP) [N/y/?] n
Ext3 journalling file system support (EXT3_FS) [M/n/y/?] m
  Ext3 extended attributes (EXT3_FS_XATTR) [Y/n/?] y
    Ext3 POSIX Access Control Lists (EXT3_FS_POSIX_ACL) [Y/n/?] y
    Ext3 Security Labels (EXT3_FS_SECURITY) [Y/n/?] y
The Extended 4 (ext4) filesystem (EXT4_FS) [N/m/y/?] (NEW) m
```

Aquí puede ver que el nuevo núcleo que estoy compilando ha añadido soporte para un nuevo sistema de ficheros: ext4. *oldconfig* ha pasado por mi configuración original, ha mantenido todas las opciones antiguas exactamente como estaban configuradas, y me ha preguntado qué hacer con las nuevas opciones. Normalmente es seguro elegir la opción predeterminada, pero es posible que desee cambiarla. *oldconfig* es una herramienta muy útil para presentarle sólo las nuevas opciones de configuración, lo que la hace ideal para usuarios que simplemente tienen que probar la última versión del kernel.

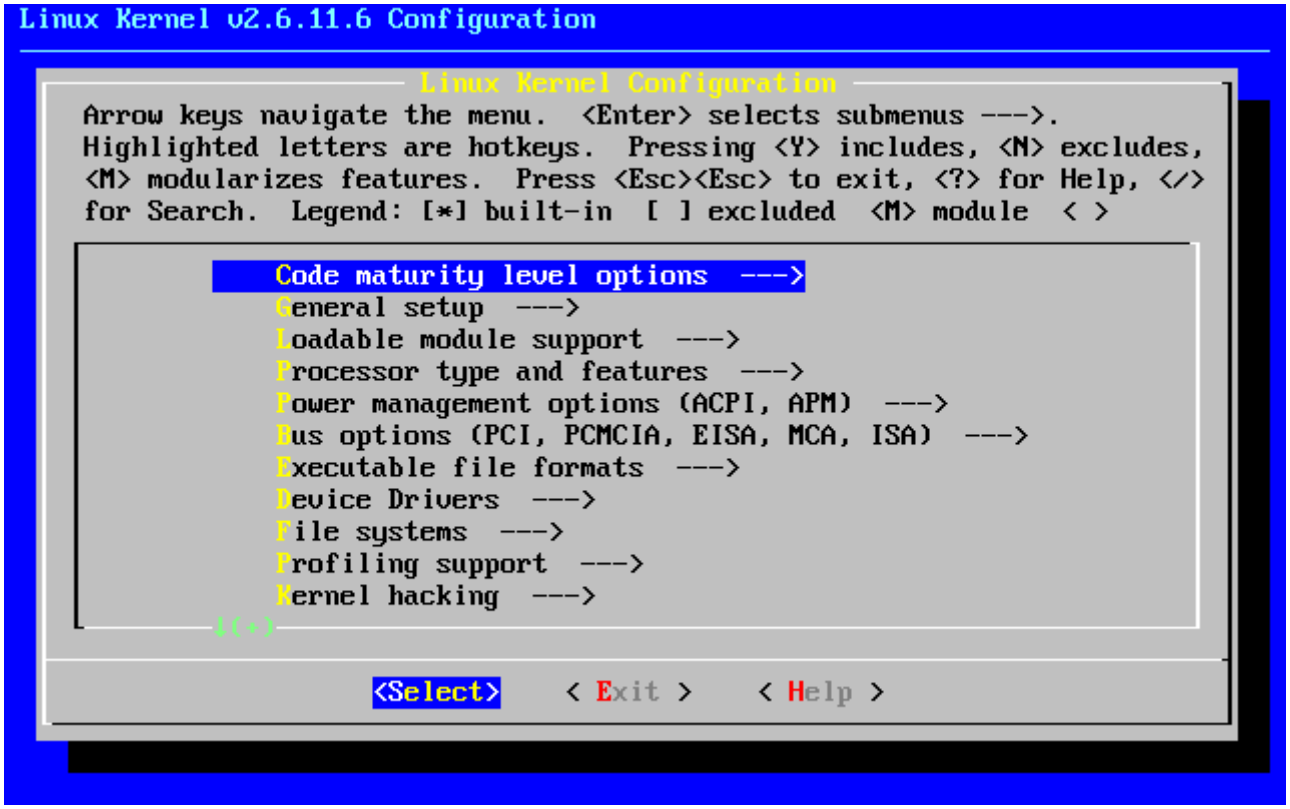
Para tareas de configuración más serias, hay múltiples opciones. El kernel de Linux puede ser configurado de tres maneras principales. La primera es *config*, que pasará por todas y cada una de las opciones, una por una, y le preguntará qué le gustaría hacer. Esto es tan tedioso que apenas alguien lo usa ya.

```
darkstar:/usr/src/linux# make config
scripts/kconfig/conf arch/x86/Kconfig
*
* Linux Kernel Configuration
*
*
* General setup
*
Prompt for development and/or incomplete code/drivers (EXPERIMENTAL) [Y/n/?]
Y
Local version - append to kernel release (LOCALVERSION) [] -test
Automatically append version information to the version string
```

```
(LOCALVERSION_AUTO) [N/y/?] n
Support for paging of anonymous memory (swap) (SWAP) [Y/n/?]
```

Afortunadamente, hay dos maneras mucho más fáciles de configurar su kernel, *menuconfig* y *xconfig*. Ambas crean un programa guiado por menú que le permite seleccionar y deseleccionar opciones sin necesidad de tener que pasar por cada una de ellas. *menuconfig* es el método más comúnmente utilizado, y el que recomiendo. *xconfig* sólo es útil si está intentando compilar el kernel desde una interfaz gráfica de usuario en ***X***. Ambas son tan similares, sin embargo, que sólo vamos a documentar *menuconfig*.

Ejecutando make menuconfig desde una terminal le presentará la interfaz amigable basada en curses que se muestra a continuación. Cada sección del núcleo tiene su propio submenú, y puede navegar con las teclas de dirección.



Si está compilando un kernel que es la misma versión que la del kernel común de Slackware, debe establecer la opción *"Local version"*. Esto se encuentra en el submenú *"General setup"*. No establecerlo resultará en que su kernel compilará sobrescribiendo todos los módulos utilizados por los kernels de stock. Esto puede hacer rápidamente que su sistema no inicie.

Una vez que haya terminado de configurar el kernel, es el momento de comenzar a compilarlo. Hay muchos métodos diferentes para esto, pero el más fiable es utilizar *bzImage*. Cuando pase este argumento a ***make***, la compilación del kernel comenzará y verá muchos datos desplazarse a través de la terminal hasta que el proceso de compilación haya finalizado o se haya encontrado un error fatal.

```
darkstar:/usr/src/linux# make bzImage
scripts/kconfig/conf -s arch/x86/Kconfig
  CHK     include/linux/version.h
  CHK     include/linux/utsrelease.h
  SYMLINK include/asm -> include/asm-x86
```

```
  CALL    scripts/checksyscalls.sh
  CC      scripts/mod/empty.o
  HOSTCC  scripts/mod/mk_elfconfig
  MKELF   scripts/mod/elfconfig.h
  HOSTCC  scripts/mod/file2alias.o
... many hundreds of lines ommitted ...
```

Si el proceso termina en un error, debe comprobar su configuración del kernel en primer lugar. Los errores de compilación normalmente son causados por un error del archivo `.config`. Asumiendo que todo haya ido bien, aún no hemos terminado del todo, ya que tenemos que construir los módulos.

```
darkstar:/usr/src/linux# make modules
  CHK     include/linux/version.h
  CHK     include/linux/utsrelease.h
  SYMLINK include/asm -> include/asm-x86
  CALL    scripts/checksyscalls.sh
  HOSTCC  scripts/mod/file2alias.o
... many thousands of lines omitted ...
```

Si tanto el kernel como los módulos compilados terminan con éxito, estamos listos para instalarlos. La imagen del kernel necesita ser copiada en un lugar seguro, normalmente el directorio `/boot`, y usted debería darle un nombre único para evitar sobrescribir cualquier otra imagen del núcleo situada allí. Tradicionalmente las imágenes del kernel se llaman `vmlinuz` con el lanzamiento del kernel y la versión local añadida.

```
  darkstar:/usr/src/linux# cat arch/x86/boot/bzImage > /boot/vmlinuz-
release_number-local_version
  darkstar:/usr/src/linux# make modules_install
```

Una vez completados estos pasos, tendrá una nueva imagen del kernel situada debajo de `/boot` y nuevos módulos del kernel situados debajo del directorio `/lib/modules`. Con el fin de utilizar este nuevo kernel, necesitará editar `lilo.conf`, crear un initrd para ello (sólo si necesita cargar uno o más de estos módulos del kernel para arrancar), y ejecutar ***lilo*** para actualizar el cargador de arranque. Cuando reinicie, si todo sale según el plan, debería tener la opción de arrancar con su kernel recién compilado. Si algo saliese mal, puede ser que pase algún tiempo arreglando el problema.

Navegación de capítulos

Capítulo anterior: Mantener un registro de las actualizaciones

Fuentes

- Fuente original: http://www.slackbook.org/beta

- Escrito originalmente por Alan Hicks, Chris Lumens, David Cantrell, Logan Johnson

* Traducido por Sergio

slackbook, kernel