There are no translations available.
En este artículo se explicará tanto la posibilidad de añadir nuevos módulos a nuestro núcleo como la opción de compilar un kernel nuevo entero...

# Configuración y compilación del kernel de GNU/Linux. Módulos

## 1. Introducción

Este artículo no es ni mucho menos una guía detallada de compilación del kernel de Linux (para eso es mejor recurrir a documentación específica sobre nuestra distribución y la versión concreta del kernel que pretendemos compilar). Nuestro propósito es más bien informar de las distintas alternativas disponibles para añadir nuevas funcionalidades al sistema, especialmente en lo que se refiere al soporte de nuevo hardware. Con esta intención, se explicará tanto la posibilidad de añadir nuevos módulos a nuestro núcleo como la opción de compilar un kernel nuevo entero.

Aunque la exposición se realizará de forma completamente general, se han incluido algunos ejemplo concretos para que el lector vea la aplicación práctica. Los ejemplos se corresponden con la versión estable de Ubuntu en el momento de escribir este artículo (7.04 “feisty”), no obstante, son perfectamente aplicables también a Debian, aunque para otras distribuciones debería consultarse la correspondiente documentación.

A lo largo del artículo se utilizará el término Linux para referirnos exclusivamente al kernel del sistema operativo (que designaremos de manera más apropiada como GNU/Linux).

## 2. Razones por las que no deberíamos compilar nuestro propio kernel

Basta con una sencilla búsqueda en google para comprobar que existen infinidad de guías, tutoriales y documentos de todo tipo sobre la compilación del núcleo en sistemas GNU/Linux: desde información general hasta documentación específica para una distribución (incluso para determinadas versiones de las mismas) y/o versiones del kernel.

Para mi sorpresa, la mayoría de enlaces empiezan ! ANIMANDO AL USUARIO A HACERLO !, argumentado lo sencillo que resulta y la suerte que tenemos de utilizar tal o cual distribución que cuenta con determinada utilidad o paquete específico. Como mucho, encontramos algún aviso sobre la posibilidad de que nuestro sistema no arranque bien o una eventual pérdida de datos

Desde mi punto de vista es una irresponsabilidad animar a los usuarios, especialmente a los más inexpertos, a compilar sus propios kernels sin advertirles previamente de los riesgos que corren, que sin duda van mucho más lejos que la dificultad para arrancar de nuevo el sistema.

Efectivamente, seguir una "receta" paso a paso para compilar el núcleo puede ser relativamente sencillo para casi cualquier usuario de nivel medio. Sin embargo, el problema no reside en el grado de dificultad técnica, sino en las consecuencias que puede tener. En este sentido, deberíamos hacer las siguientes reflexiones:

-

Las actualizaciones, especialmente las de seguridad, son, sin duda, la mayor preocupación de los responsables de los sistemas operativos actuales. Mediante ellas podemos mantener nuestro sistema convenientemente actualizado y protegido de vulnerabilidades (¡incluso los usuarios de Windows son conscientes de esto!).

-

Una distribución GNU/Linux es un sistema extraordinariamente complejo, formado por muchas piezas de software diferentes desarrolladas de manera independiente por programadores de todo el mundo. Es realmente difícil mantener la compatibilidad y el equilibrio entre ellas, por lo que dentro de cada distribución, por cada aplicación o paquete encontramos uno o más responsables (mantenedores) que se encargan de mantener seguros y actualizados los correspondientes programas, asegurarse de cuales son las dependencias correctas y comprobar la compatibilidad dentro del conjunto de la distribución. Por este motivo, sea cual sea nuestra distribución favorita, la recomendación general es actualizar el equipo con regularidad y no salirse de la distribución que se esté usando. Compilar por nuestra cuenta uno sólo de esos programas pone en peligro la estabilidad del sistema completo y además supone la renuncia a las correspondientes actualizaciones, es decir, nos convierte en mantenedores del mismo y nos pone por delante de los expertos programadores responsables del paquete correspondiente.

-

Si entendemos que utilizar un programa (o una versión del mismo) no incluido en la distribución es peligroso, el caso del núcleo es especialmente arriesgado. Debemos tener en cuenta que el kernel es la pieza más delicada del sistema, ya que gestiona y accede de manera privilegiada a todos los recursos disponibles. Precisamente por este motivo, los responsables del núcleo de cada distribución suelen ser programadores de reconocido prestigio dentro de la comunidad, y

se mantienen en permanente contacto con los desarrolladores del kernel Linux, incluido el mismísimo Linus Torvalds.

-

Por último, cambiar de kernel introduce el riesgo de que nuestro equipo no funcione del todo bien, o incluso que ni siquiera arranque, por lo que debemos prepararnos previamente para este tipo de eventualidades.

-

¿Cuál es el motivo que nos ha llevado a interesarnos por la compilación del núcleo? No hay problema si sólo estamos experimentando o queremos aprender, pero uno de los motivos más frecuentes es conseguir que funcione determinado dispositivo hardware que no está soportado en el kernel que estamos usando. Bien, si este es el caso, es importante saber que disponemos de otras opciones, y que deberíamos agotar el resto de vías posibles antes de ponernos a compilar nuestro propio kernel.

Conclusión, si no somos un experto hacker del kernel dispuesto a asumir el riesgo de bloquear nuestro sistema, exponernos a vulnerabilidades, y capaz de aceptar la responsabilidad de estar pendientes de actualizaciones y parches ... será mejor que no nos lancemos al vacío, y que sigamos leyendo este artículo. Nuestra intención no va a ser disuadirle, sino informarle tanto para que entienda los riesgos a los que se expone, como para que conozca las distintas alternativas y las ventajas e inconvenientes de cada forma de proceder.

## 3. Soporte de dispositivos hardware y módulos del kernel

Como ya hemos dicho, conseguir soporte para nuevos dispositivos hardware es una de las razones más frecuentes para "meterle mano" al kernel. En este sentido, es muy importante saber que la arquitectura del núcleo de Linux permite cargar y descargar extensiones al sistema operativo en forma de módulos sin tener que detener o reiniciar el sistema. Aunque estos módulos pueden ser de distintos tipos, nos centraremos en el caso de los controladores de dispositivo (drivers). Estos controladores permiten el acceso al hardware que se encuentra conectado al sistema, por lo que son especialmente importantes para el caso que nos ocupa.

Los módulos aportan dos ventajas fundamentales. Por un lado, la posibilidad de carga y

descarga bajo demanda disminuye los requerimientos de memoria (sólo necesitamos cargar los módulos correspondientes al hardware conectado), y nos liberan de la necesidad de reiniciar el sistema cada vez que se añade un dispositivo. Por otra parte, evitan que tengamos que reconstruir un nuevo kernel monolítico cada vez que queremos añadir el soporte para un dispositivo nuevo.

Por ejemplo, para mostrar los modulos que tenemos cargados, ejecutaremos: lsmod

Las principales utilidades para trabajar con módulos son:

lsmod , modinfo, insmod , modprobe , rmmod , depmod y update-modules.

En feisty, se incluyen en el paquete  module-init-tools ,

A medida que avanza el desarrollo del kernel, se van incluyendo nuevos módulos, de manera que cada versión nueva reconoce y es capaz de hacer funcionar más dispositivos. Sin embargo, cambiar de versión del núcleo no es, ni mucho menos, la  solución mágica  para soportar un nuevo hardware concreto. Aunque alguien haya desarrollado un módulo específico, no todos llegan a incluirse en las versiones oficiales del kernel, o puede darse alguna otra circunstancia desfavorable (que se incluya pero al cabo de mucho tiempo, que la versión elegida no nos sirva, etc).

En lugar de esperar a que aparezcan nuevas versiones de Linux (y  rezar  para que incluyan determinados módulos), la alternativa es compilar los nuevos módulos contra el kernel actual de nuestra distribución. Esta opción nos permite mantenernos dentro de la distribución, a la vez que ampliamos el soporte hardware.

## 4. Compilar nuevos módulos del kernel

Primero, necesitamos obtener las cabeceras (headers) y quizás el código fuente completo correspondiente al núcleo que estamos utilizando.

En las distribuciones derivadas de Debian, para instalar los headers podemos ejecutar:

apt-get install linux-headers-`uname -r`

Para instalar el fuente completo del kernel, lo mejor es instalar el metapaquete 'linux-source' :

apt-get install linux-source

Es recomendable instalar las mismas utilidades que se utilizan para construir el kernel (los detalles varían de una distribución a otra)

En Ubuntu y en Debian, nuestra recomendación es:

apt-get install build-essential kernel-package libncurses5

apt-get installlibncurses5-dev libqt3-mt-dev

apt-get install libc6-dev dialog module-assistant

Por último, necesitamos el código fuente de los módulos que queremos compilar. (Obsérvese que dicho código puede necesitar dependencias adicionales para su compilación, por lo que conviene empezar leyendo la documentación que se se suministre). El primer sitio donde debemos buscar es entre los paquetes de nuestra propia distribución, utilizando las utilidades específicas de la misma.

*Module Assistant*

Para empezar, en los sistemas Ubuntu/Debian, deberíamos utilizar el programa module-assistant. Se trata de una herramienta fácil de manejar, y que permite compilar módulos adicionales seleccionando opciones en unos sencillos menús. Sólo funciona con aquellos módulos que han sido empaquetados y preparados expresamente por los correspondientes mantenedores para poder ser utilizados desde este programa.

Ejecutándolo (directamente como root o con 'sudo m-a') podemos elegir la opción 'SELECT' del menú principal, con la que se nos muestra la lista de paquetes de módulos disponibles para module-assistant en nuestra distribución. Una vez seleccionados, la opción 'BUILD' lo compilará (incluso nos ofrecerá la posibilidad de descargare instalar el paquete con el fuente, si es que no lo habíamos hecho ya).

El resultado de module-assistant es uno o más paquetes .deb en el directorio /usr/src con un nombre similar a:

*nombre_del_módulo - versión_del_kernel - versión_del_módulo _ arquitectura .deb*

que pueden instalarse con el comando:

dpkg -i {nombres_de_paquetes}

Si no encontramos el módulo en nuestra distribución, deberíamos ampliar la búsqueda incluyendo más repositorios, ya que siempre resultará más sencillo de compilar si está empaquetado.

El orden de preferencia debería ser: repositorios adicionales para nuestra misma distribución, 'backports' oficiales o semi-oficiales, versiones más modernas de nuestra distribución, distribuciones derivadas de la nuestra, repositorios semi-oficiales mantenidos por

programadores de prestigio reconocido en la comunidad, etc.

En Ubuntu, empezaremos comprobando que nuestro fichero /etc/apt/sources.list tiene también habilitadas las líneas correspondientes a universe, multiverse y backports. Para feisty, serían:

deb http://archive.ubuntu.com/ubuntu/ feisty main restricted

deb http://security.ubuntu.com/ubuntu feisty-security main restricted

deb http://archive.ubuntu.com/ubuntu/ feisty-updates main restricted

deb http://archive.ubuntu.com/ubuntu/ feisty universe multiverse

deb http://es.archive.ubuntu.com/ubuntu/ feisty-backports main restricted universe multiverse

Tanto la herramienta gráfica synaptic como el comando apt-cache search permiten realizar búsquedas en el sistema de paquetes de Ubuntu/Debian. Para las búsquedas, es interesante saber que la mayoría de paquetes para module-assistant contienen la cadena "-source" en el nombre.

Para buscar en otras versiones de Ubuntu, podemos visitar http://packages.ubuntu.com/ ( http://packages.debian.org/
para Debian) donde encontraremos distintos criterios de búsquedas.

Para Debian, disponemos también de http://www.backports.org , con backports "casi oficiales".

Aunque la probabilidad no es muy alta, existe también la posibilidad de encontrar paquetes para nuestra distribución directamente en la página del programador o la web del proyecto del módulo que nos interesa. En caso de encontrarlos allí, hay que tener en cuenta que, aún tratándose de paquetes  no oficiales , están construidos directamente por los responsables de los programas, por lo que es más que razonable confiar en la calidad de los mismos. Periódicamente deberíamos visitar la web para comprobar si han empaquetado alguna actualización.

*Compilación directa desde el código fuente*

La última opción sería bajar directamente el código fuente desde la página del proyecto, y seguir sus instrucciones para compilarlo 'manualmente'. La dificultad en este caso reside en conseguir satisfacer las dependencias de compilación, ya que la documentación las especificará de manera genérica y puede no ser trivial encontrar cuales son los paquetes específicos de nuestra distribución.

También en este caso, es importante visitar la página con frecuencia para detectar nuevas versiones, parches. etc ...

Para buscar, es recomendable utilizar los repositorios de proyectos más populares, como http://sourceforge.net/
y
http://
alioth
.debian.org/
, o portales que mantienen recopilaciones actualizadas como
http://freshmeat.net/
.

## 5. Conseguir un nuevo kernel

En algunos casos es posible conseguir imágenes binarias de versiones más modernas del kernel que puedan utilizarse en nuestra distribución sin tener que compilarlas nosotros mismos.

Podemos intentar conseguir estos núcleos buscando en los mismos repositorios adicionales que habíamos considerado para la obtención de módulos: repositorios semi-oficiales, backports, distribuciones derivadas, etc.

Si nos funcionan, usar directamente estos paquetes binarios es muy rápido y sencillo. No obstante, la contrapartida es que no tenemos ninguna garantía de actualización, ya que no se trata de paquetes oficiales mantenidos para nuestra distribución. Esto puede suponer un severo inconveniente a medio/largo plazo.

También podríamos probar con los binarios del kernel de la siguiente versión de nuestra misma distribución, aunque, si este es el caso, quizás deberíamos considerar seriamente la posibilidad de actualizar el equipo completo y cambiar de versión de la distribución.

Sea cual sea su origen, antes de arrancar con una nueva versión del kernel, debemos tomar una serie de precauciones, que pasamos a desarrollar.

Quizás  no funcione todo bien  con el nuevo núcleo, de manera que, como es lógico, no debemos desinstalar el anterior, y debemos configurar nuestro gestor de arranque para poder utilizar ambos indistintamente. No está de más tener un livecd con herramientas básicas a mano por si hay que realizar algún tipo de arranque de emergencia: Kanoppix, Damn Small Linux, Ubuntu Desktop, etc.

NOTA (específica para Ubuntu):

Usuario root: en Ubuntu toda la administración se realiza mediante 'sudo', de manera que una instalación normal no puede usarse la cuenta de root, ya que se encuentra 'bloqueada' por no haberse asignado pasword. Antes de empezar las pruebas es muy recomendable asignarle password. Para ello, nos convertimos primero en root con:

```
sudo passwd
```

(deberemos introducir nuestro password de login) y después ejecutaremos 'passwd' que nos solicitará de forma interactiva un nuevo password para root, así como su correspondiente confirmación.

Prácticamente todas las distribuciones utilizan un ramdisk de inicio. Debemos asegurarnos de que se ha generado el correspondiente al nuevo kernel, y que el gestor de arranque lo ha tenido en cuenta (a menudo esto es automático en la mayoría de distribuciones).

Si ya estamos utilizando módulos personalizados o compilados por nuestra cuenta (por ejemplo con module-assistant), no funcionarán con el nuevo kernel hasta que no los compilemos contra él. Podríamos perder el acceso a dispositivos críticos (red, vga o incluso el disco duro !!!), por lo que debemos identificar previamente que hardware puede verse afectado y considerar configuraciones alternativas (por ejemplo, editar el fichero xorg.conf y cambiar el driver a 'vesa').

## 6. Compilar un nuevo kernel

Por último, abordaremos la opción de compilar nuestro propio kernel completo a partir del código fuente. Antes de empezar, conviene tomar las mismas precauciones que en el apartado anterior, para no llevarnos desagradables sorpresas cuando arranquemos con el nuevo núcleo.

Tenemos dos opciones para obtener el código fuente. Usar una versión empaquetada para nuestra distribución (que incorporará ciertos parches y modificaciones), o descargar el fuente directamente de kernel.org (lo que se denominan "vanilla kernels , ya que no han sido modificados por nadie).

Siempre que se pueda, es recomendable elegir un kernel con parches para nuestra distribución, pero en cualquier caso, la forma de proceder es muy similar.

Ejemplo Ubuntu (kernel empaquetado).

En el caso de Ubuntu, podríamos descargar el código fuente del kernel de una versión más

moderna, desde http://packages.ubuntu.com . Tanto en Ubuntu como en Debian, los paquetes con el fuente del núcleo se denominan: linux-source- numero_de-versión . Por ejemplo, en el momento de escribir ese artículo, en Ubuntu disponemos de linux-source-2.6.22.

Una vez instalado, dispondremos del código fuente como un fichero comprimido en /usr/src. Para descomprimirlo, podríamos ejecutar los siguientes comandos (como root):

cd /usr/src

tar -xvjf linux-source- versión .tar.bz2

ln -s linux-2.6.22 linux

Ejemplo Ubuntu (vanilla kernel).

Si optamos por la segunda opción (vanilla kernel), podemos descargar el fichero .tar.bz2 de la última versión desde http://kernel.org , dejarlo en /usr/src y descomprimirlo con el mismo comando.

La última versión estable se encuentra disponible, con o sin parches, por ejemplo, en este momento, la última versión estable es la 2.6.21. La versión estable con los últimos parches (2.6.21.3) está en:

http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.21.3.tar.bz2

Si lo descargamos en /usr/src, después, desde dicho directorio ejecutaremos:

tar -xvjf linux-2.6.21.3.tar.bz2

ln -s linux-2.6.21 linux

Si queremos experimentar con la versión inestable, es preferible bajar la versión "limpia" del kernel, por ejemplo para el 2.6.21:

http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.21.tar.bz2

Una vez descomprimida y creado el enlace /usr/src/linux, consultaremos la url:

http://www.kernel.org/pub/linux/kernel/v2.6/testing/

para bajar el último parche , en este momento:

http://www.kernel.org/pub/linux/kernel/v2.6/testing/patch-2.6.22-rc3.bz2

NOTA: la diferencia en el número de versión no es una errata. Las versiones inestables se consideran como "pre-releases" o versiones "alfa", de manera que, por ejemplo, el parche 2.6.22.rc3 se aplica sobre la versión 2.6.21 del kernel para obtener la "release candidate 3" del kernel 2.6.22.

Una vez descargado el parche en /usr/src, ejecutaremos:

cd linux

bzip2 -dc /usr/src/patch-2.6.22-rc3.bz2 | patch -p1

Por último, configuraremos y compilaremos el nuevo núcleo. En general, es una buena idea utilizar nuestra configuración actual como punto de partida para generar el nuevo kernel. Los detalles dependen de la distribución, por lo que de nuevo remitimos al lector a que consulte la documentación específica.

En Ubuntu o Debian, ejecutar:

cp /boot/config-`uname -r` /usr/src/linux/.config

Para configurar el kernel, ejecutaremos:

make menuconfig

Una vez estamos en la configuración, selecionaremos la opción  Load an Alternate Configuration File y elegiremos  .config para establecer nuestros valores actuales.

Ahora podemos hacer los cambios que queramos. Para que nuestra configuración tenga efecto, al salir contestaremos  Yes a la pregunta  Do you wish to save your new kernel configuration?

Ya por último, ejecutaremos los comandos:

make-kpkg clean

make-kpkg --initrd --append-to-version=-custom kernel_image kernel_headers

Al cabo de un tiempo más o menos largo, dependiendo fundamentalmente de nuestro procesador, encontraremos dos paquetes en nuestro directorio /usr/src: (en nuestro ejemplo)

*linux-headers-2.6.22-rc3-custom_2.6.22-rc3-custom-10.00.Custom_i386.deb*

*linux-image-2.6.22-rc3-custom_2.6.22-rc3-custom-10.00.Custom_i386.deb*

Que instalaremos con:

dpkg -i linux-headers-2.6.22-rc3-custom_2.6.22-rc3-custom-10.00.Custom_i386.deb

dpkg -i linux-image-2.6.22-rc3-custom_2.6.22-rc3-custom-10.00.Custom_i386.deb

Sólo nos queda reiniciar y probar nuestro flamante nuevo kernel.