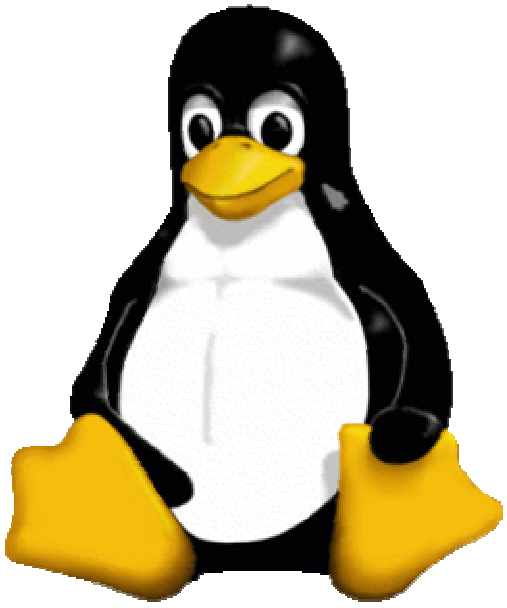
aLeZX

Recompilando el kernel de GNU/Linux

Cambiando las entrañas de Tux



Este artículo habría sido publicado en Los Cuadernos de Hack x Crack, revista a la que le debo mucho. Por este motivo, quiero que, mientras lo leáis, imaginéis que pertenece a ese número que nunca llegó a ver la luz.

A Hack x Crack, gracias.

aLeZX

RECOMPILANDO EL KERNEL DE GNU/LINUX

CAMBIANDO LAS ENTRAÑAS DE TUX

A principios de los años 90, un señor llamado Linus Benedict Torvalds, no contento con los sistemas operativos de la época, comienza a diseñar, como un simple hobby, el núcleo de un sistema operativo que cambiaría la historia de la informática.

Catorce años más tarde seguimos usando ese núcleo, Linux, pero ¡ya va siendo hora de que lo actualicemos!, ¿no?

Recompilemos el kernel de GNU/Linux.

1.- INTRODUCCIÓN

Hola a todos. Más de una vez he visto documentos realmente interesantes acerca de Linux, pero que tenían un inconveniente para los usuarios más noveles: ¡había que recompilar el kernel! En gran cantidad de esos documentos yo esperaba que explicasen qué es eso de recompilar el kernel y cómo se hacía, sin embargo, lo único que veía al respecto era que su explicación se escapaba de la temática del artículo y que lo dejarían para otra ocasión. Al final, después de mucho tiempo pensando que esa tarea iba a ser muy difícil y, dejándolo siempre para otro momento, me decidí a meterle mano. Y ahora mirad, escribiendo un artículo sobre recompilar el kernel de nuestro amigo Tux :D. En fin, que los que no tengáis ni idea de cómo hacer esto, ya no tenéis excusa para comenzar la recompilación ;)

¿Estáis listos? ¡¡PUES MANOS A LA OBRA!!

2.- UN POCO DE TEORÍA ANTES DE EMPEZAR

El kernel es el alma de todo sistema operativo (SO). Es el núcleo, no sólo de GNU/Linux sino de cualquier otro SO (sí, de todos). El kernel se encuentra en la capa más baja del sistema operativo y permite interactuar con el hardware, controlar los datos, la gestión de la memoria...

NOTA: Para más información acerca de los núcleos de los sistemas operativos ya sabéis dónde buscar. Os pongo el enlace para que no os quebréis: www.google.es :P

Pero bueno, lo que a nosotros nos interesa es el kernel de Linux. En verdad, esta forma de expresarlo es incorrecta. ¿Por qué? Pues porque lo que nosotros llamamos sistema operativo Linux, se llama realmente GNU/Linux, siendo Linux, en esa combinación de nombres, el kernel del sistema operativo.

Si buscamos la palabra "compilar" en el Diccionario de la Real Academia Española veremos:

> Preparar un programa en el lenguaje máquina a partir de otro programa de ordenador escrito en otro lenguaje.

Y lo que vamos a hacer no es otra cosa sino esto, con la única diferencia de que el programa que vamos a pasar a lenguaje máquina es el kernel de GNU/Linux y que lo vamos instalar en nuestro sistema ☺.

2.1.- ¿Por qué vamos a tener que recompilar el kernel?

Pues muy simple, los tiempos cambian y los kernels también :D Si queremos mayor compatibilidad de hardware, tener soporte para todos los sistemas de archivos, etc., necesitaremos recompilar el kernel ;).

3.- PREPARÁNDOLO TODO.

En el momento de escribir este documento ya vamos por el kernel 2.6.14, de la rama 2.6. A su vez, también se usa la rama 2.4 (rama par = estable, impar = inestable). Debido a esto, yo usaré la versión 2.6.14, pero vosotros debéis usar la última versión estable, aunque podríais utilizar la que quisieseis. Por eso, tened en cuenta que debéis sustituir 2.6.14 (mi versión) por vuestra versión en los comandos que yo use.

3.1- Consiguiendo las fuentes.

Muy bien, ya empezamos con la parte práctica. Vamos a descargarnos la última versión estable del kernel para estar en la onda ;).

Abrimos nuestro navegador web y nos metemos en "The Linux Kernel Archives", también conocido para nuestro navegador como www.kernel.org (***imagen 1***). Ahora nos vamos a cualquiera de las URLs que aparecen en la parte de arriba de la web (por protocolo FTP o HTTP) y buscamos hasta encontrar todas las ramas de kernels clasificaditas y todo para no perderse. Para los despistados el link directo es: http://www.kernel.org/pub/linux/kernel o ftp://ftp.kernel.org/pub/linux/kernel/

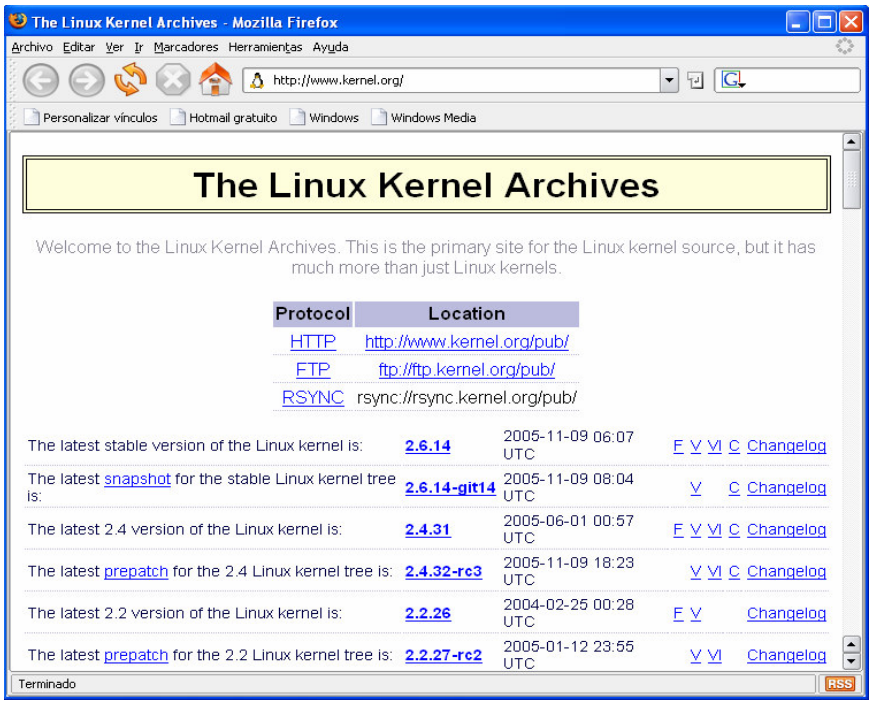


Imagen 1

Le damos en este caso a la 2.6 y nos bajamos el archivo .bz2 de la versión que queráis. Ahora nos toca esperar un ratillo dependiendo de si tenéis un ancho o un "estrecho" de banda xD.

3.2- Descomprimiendo las fuentes.

¿Ya os habéis bajado las fuentes? Vale. Nos vamos al directorio /usr/src:

```
cd /usr/src
```

> NOTA: Teóricamente la compilación se podría realizar en cualquier directorio, pero lo más común es hacerlo en /usr/src. Esto me recuerda un problemilla que he tenido mientras compilaba el kernel en mi PC y es que me quedé sin espacio en el disco duro. Aseguraos de que tenéis suficiente espacio en el disco duro antes de comenzar la recompilación u os llevaréis una desagradable sorpresa. Advertidos quedáis, quien avisa no es traidor ;).

Ahora copiamos el archivo de las fuentes al directorio /usr/src y descomprimimos el *tarball* de la siguiente forma:

```
cp  /media/cdrecorder/linux-2.6.14.bz2  /usr/src/
tar xvfj linux-2.6.14.bz2
```

NOTA: En mi caso, al tener las fuentes en un CD, las copio desde /media/cdrecorder, pero vosotros debéis poner el directorio donde os las hayáis descargado.

Una vez descomprimido el *tarball*, proceso que tarda unos minutos, es muy recomendable crear un enlace simbólico a /usr/src/linux desde las fuentes recién descomprimidas. Esto lo hacemos así:

```
ln -s  /usr/src/linux-2.6.14  /usr/src/linux
```

NOTA: Aunque no voy a entrar en detalles, diremos que un enlace simbólico es una especie de acceso directo, de forma que, para entrar en el directorio en cuestión, en vez de escribir /usr/src/linux-2.6.x.x simplemente podamos escribir /usr/src/linux.

Y ya nos podemos ir preparando para configurar el nuevo kernel ;)

3.3.- Preparándonos para configurar

A continuación, entramos en /usr/src/linux (enlace simbólico) y vemos una serie de archivos y carpetas (no voy a explicar la función de cada carpeta porque si no recuerdo mal ya fue explicado en la revista HxC número 27).

Normalmente, todas las distribuciones de GNU/Linux suelen traer en sus repositorios el GCC (GNU C Compiler), así como otros programas necesarios para la compilación. Así que, si da algún error (***imagen 2***) simplemente debéis leer qué os dice y ya sabréis qué paquete necesitáis, y si no, podéis pasaros por nuestro maravilloso foro (si es que no lo habéis hecho ya) y preguntar por el problema que tenéis. Así de fácil ;).

También sería interesante que supieseis cuál es el hardware de vuestro PC. Esto lo podéis conocer mediante el comando /sbin/lspci.

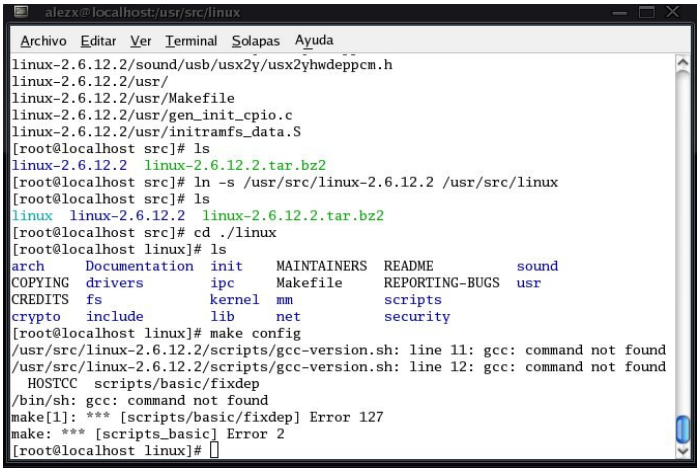


Imagen 2

4.- CONFIGURACIÓN

Muy bien, ahora nos acercamos al proceso más pesado de toda la compilación de un kernel, la configuración. Existen varios métodos de configuración:

- make config: Es un método muy poco recomendable, además de muy lento y cansino, ya que se va mostrando opción por opción.

- make menuconfig: Es el método que usaré yo. ¿Por qué? Porque a pesar de ser en modo texto, me gusta más que el método de configuración a través de las X y se realiza mediante un menú, pudiéndonos mover por la configuración del kernel a nuestro libre albedrío :). Un error muy común al usar este tipo de configuración es el de que faltan las librerías ncurses. Estas librerías son necesarias para este tipo de configuración. Se suelen encontrar en los repositorios de cada distribución, así que es muy simple instalarlas (libncurses5-dev) ;) (***imagen 3***)

- make xconfig: Es un método muy semejante al anterior, con la única diferencia de que se usa el servidor X para manejarlo.

- make gconfig: Como auténtica primicia para los usuarios de GNOME (:D) existe este otro método, que no usaremos por respeto a los usuarios de KDE y otros escritorios ;) (¡para que después digan! :P) (***imagen 4***)

- make oldconfig: Lo que hace esta opción es coger la configuración que trae nuestro kernel actual y preguntar por las opciones nuevas. Es útil en algunos casos, aunque no es muy recomendable, ya que se salta la mayor parte de las opciones (usando la configuración del anterior kernel). (No usar para pasar de la rama 2.4 a la 2.6)

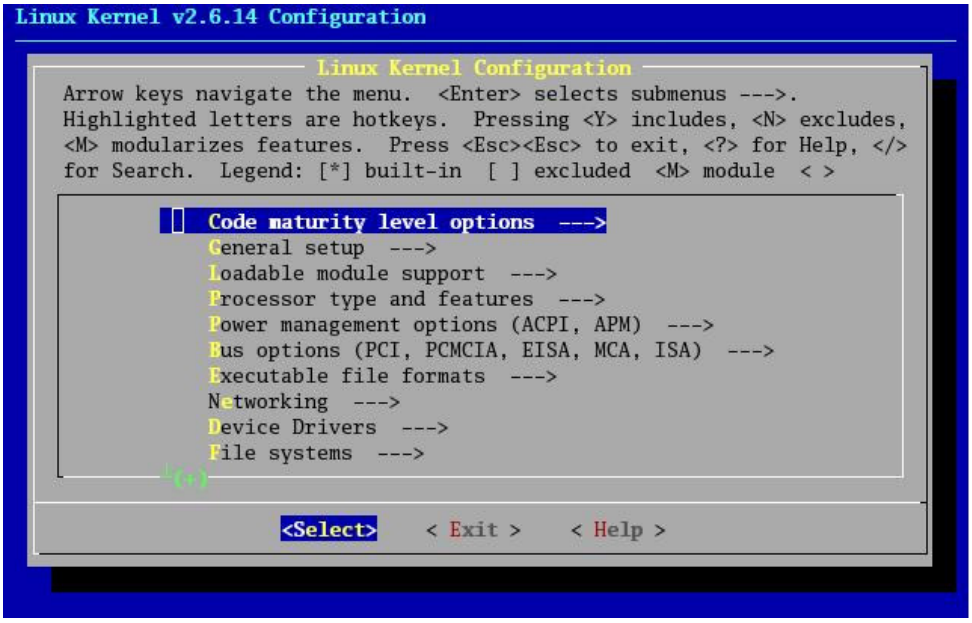


Imagen 3

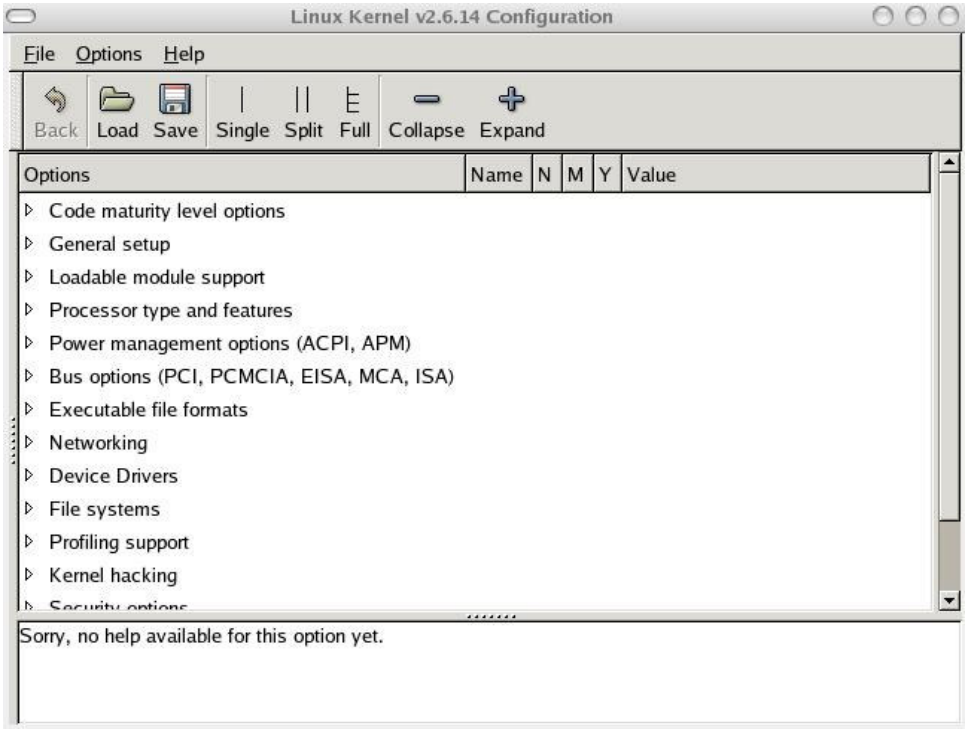


Imagen 4

NOTA: El proceso configuración del kernel es demasiado extenso como para explicarlo sobre el "papel", además de que necesitaría una inmensidad de páginas y a ver quién va a ser el listo que se lee el artículo enterito ☺. Por lo tanto, lo que voy a hacer es realizar un resumen de todas las secciones de configuración del kernel, para, al final, explicar detalladamente las opciones que crea más interesantes y de importancia más relevante. De todas formas, en la red hay numerosos documentos que explican paso a paso todas las opciones, así que, si tenéis alguna duda, podéis preguntar al amigo Google, o preguntar en nuestro maravilloso foro. Por cierto, en el propio programa de configuración hay una opción llamada "Help" con la que veréis un texto en perfecto inglés explicando la definición de cada cosa ;).

4.1.- Comenzando la configuración

Ahora escribimos el comando "make menuconfig" (o el que queráis) y ¡comienza la fiesta! (***imagen 3***)

Dentro del menú de configuración del kernel, podemos poner las distintas opciones, como "compiladas dentro del kernel" → [*](opciones que se usarán en todo momento) pulsando la tecla Y sobre la opción; como "módulos" → <*>(opciones que se usarán cuando las necesitemos) pulsando la tecla M sobre la opción; o podemos, simplemente, omitir esas opciones → [] / <> (no se usarán nunca) pulsando la tecla N sobre la opción.

A continuación, voy a resumir todas y cada una de las secciones para que no os perdáis ☺.

1. ***Code maturity level options*** → Está relacionada con los drivers que no están 100% terminados o que están obsoletos (personalmente, selecciono todas las opciones que contiene).

2. ***General Setup*** → Aquí encontramos, como su nombre indica, opciones generales de la configuración del kernel, tales como soporte para la conexión a Internet, hardware PCI, soporte para ACPI, etc.… Debemos seleccionar aquellas opciones que concuerden con nuestro sistema.

3. ***Loadable module support*** → Este menú es bastante interesante y, ya de paso, me va a servir para explicar la parte de los módulos.

   Un kernel está formado por "módulos" compilados dentro del mismo, y módulos externos a él. La ventaja de los módulos externos a él es que no ocuparán espacio en la memoria cuando no los usemos. Por ejemplo, si tenemos un portátil con un puerto

i.Link (puerto 1394) que se usa para transferir fotografías y vídeos desde las cámaras digitales, podremos usar la opción del kernel que da soporte a esto como módulo, módulo que sólo cargaremos en memoria cuando vayamos a transferir fotografías. De lo contrario, si lo incluyésemos en el kernel, ocuparíamos un espacio en la memoria innecesario, ya que no siempre lo estaríamos usando.

4. ***Processor type and features*** → Está relacionado con el tipo de procesador y sus características (en sus opciones hay que seleccionar el tipo de procesador, arquitectura, etc…).

5. ***Power Managent Options*** → Este es un menú bastante importante, sobre todo para los equipos portátiles (aunque también puede ser beneficioso para los de sobremesa). Sirve para dar soporte al ahorro de energía. Si activamos *Power Managent Support,* podremos seleccionar ACPI (Advanced Configuration and Power Interface) o APM (Advanced Power Managent), dependiendo de nuestro PC, así como otras opciones relacionadas.

6. ***Bus Options*** → En este menú hay que tener sumo cuidado si queremos que nuestro PC funcione correctamente, puesto que da soporte a tarjetas PCI, PCMCIA, ISA, etc... Obviamente, si disponemos de puertos para alguna de estas tarjetas, deberemos seleccionarlas.

7. ***Executable file formats*** → Relacionado con los ejecutables de Linux. Aquí yo selecciono todas las opciones dentro del kernel para curarme en salud y para que luego no haya problemas con los programas, pero es cuestión de gustos ;)

8. ***Networking*** → Esta nueva opción es bastante simple. Da soporte para los distintos protocolos de red (Bluetooth, Wireless, IrDA) y otras opciones de red. Todo protocolo usado debe introducirse dentro del kernel. Obvio.

9. ***Device Drivers*** → Este menú da soporte a los distintos dispositivos del sistema. Si estáis ya muy cansados, os recomiendo que lo dejéis para más tarde, porque esta opción se las trae, jeje. Bueno, esta sección está formada por muchos submenús que, a su vez, incluyen más submenús y que, a su vez, incluyen más... esto...sí, a pesar de ser muy largo, este menú es bastante intuitivo en todos los aspectos. Sólo me queda decir que seáis listos al seleccionar los módulos. De todas formas, voy a hacer algunas aclaraciones que me parecen importantes ;):

   a) *ATA/ATAPI/MFM/RLL support* → Si tenéis un disco duro SATA (Serial ATA) NO seleccionéis la opción *Support for SATA,* ya que provoca una incompatibilidad con el driver *libata* del menú SCSI. Os lo digo por experiencia ;)

   b) *SCSI Device Support* → *SCSI low-level drivers* → *Serial ATA (SATA) support:* Esta opción es la que SÍ debéis seleccionar, porque sin ella no podréis acceder al HD (Hard Disk, disco duro) desde Linux.

NOTA: Aunque he dicho que no se podría, esto no es totalmente cierto, ya que después si que se conseguiría creando una imagen Initrd (ver 7)

c) *Networking Support* → *Wireless LAN* → Como ahora está muy de moda todo esto del wireless y demás es altamente necesario seleccionar *Wireless LAN drivers (nom-hamradio) & Wireless Extensions.* Ya de paso, si tenéis alguna de las tarjetas que se muestran abajo podéis seleccionarlas directamente y funcionarán sin problemas, así de fácil ;)

d) *Sound* ---> Si queremos poder escuchar sonido en nuestro flamante GNU/Linux, debemos compilar la opción *Sound card support* dentro del kernel, y también seleccionar la opción ALSA (Advanced Linux Sound Architecture)(más actual) u OSS (Open Sound System)(deprecated) y buscar dentro de ellos a nuestra tarjeta de sonido.

e) Hay más opciones, pero son bastante intuitivas, además de que no es plan aburriros con tantas :P

9. ***File systems* --->** Este menú contiene opciones vitales para que nuestro sistema pueda funcionar a la perfección. Da soporte a los sistemas de archivos. Debemos tener mucho cuidado al seleccionar si compilaremos dentro del kernel o fuera, como módulo. Los sistemas de archivos que usemos (normalmente ext3 o ReiserFS) DEBEN estar compilados dentro del kernel o de lo contrario no podremos arrancar, ya que no podrá ni leer la partición raíz (/) (bueno, esto no es totalmente cierto, ver 7). También debéis seleccionar, si tenéis Windows sobre NTFS, la opción que se refiere a éste, aunque personalmente, no recomiendo que deis soporte de escritura por ahora, porque los drivers no están muy desarrollados aún.

10. El resto de opciones, aunque son muy interesantes e importantes (***Kernel hacking, security options, cryptographic options...)***, no las voy a detallar. Además de ser intuitivas no son de vital importancia para el funcionamiento de nuestro pingüinito.

Y ya está ☺. Ya hemos configurado nuestro kernel. Ahora, fijémonos en las opciones que hay un poco más abajo, *Load an Alternate Configuration File* y *Save Configuration To an Alternate File*. Estas opciones significan Cargar y Guardar respectivamente y sí, sirven:

- Para guardar los cambios realizados hasta el momento, y así poder compilar el kernel descansando de vez en cuando y no teniendo que hacerlo todo seguido.

- Para cargar archivos con la configuración que deseéis darle al kernel. Estos archivos de configuración son como los que podéis encontrar en /boot que traen toda la información de vuestro kernel actual y que os puede facilitar un poco las cosas :D.

Ahora le damos a *Exit* en el menú principal y a la pregunta que aparece le contestáis que SÍ (*Yes).* Y contestadle que "sí" porque como le digáis que "no" es para pegaros una paliza. Os debe quedar algo tal que así: (***imagen 5***)

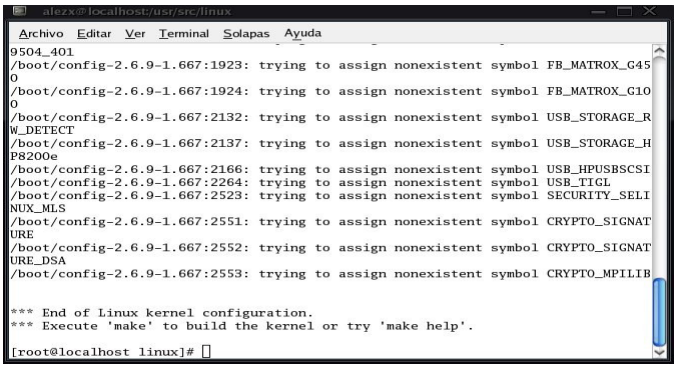


Imagen 5

5.- POR FIN!!! COMPILANDO EL KERNEL

El proceso de compilación se ha simplificado mucho en la rama 2.6. En esta rama, simplemente debéis escribir esto e ir a tomaros un refresco, porque tarda un ratito:

```
make
```

Cuando se termine, escribimos:

```
make modules_install
make install
```

Y ya está, una vez hecho, tendremos configurado incluso LILO/GRUB (***imagen 6***) para dar acceso a nuestro sistema con nuestro nuevo kernel :D.

En vuestro directorio /boot tendréis 2 ó 3 archivos nuevos, normalmente el archivo vmlinuz-2.6.x.x (dependiendo de vuestra versión), que es el archivo de imagen del kernel; el archivo System.map del kernel; y si es necesario también aparecerá el archivo Initrd (dependiendo de la configuración de vuestro kernel, lo creará automáticamente o no), del que hablaré más adelante (ver 7)

También tengo que decir que, al instalar el kernel, el directorio donde quedan las fuentes suele ser /lib/modules/'uname –r'/build (siendo 'uname –r' vuestra versión del kernel).

6.- REINICIANDO, QUE ES GERUNDIO.

Teóricamente ya hemos finalizado todo el proceso de recompilar nuestro kernel .

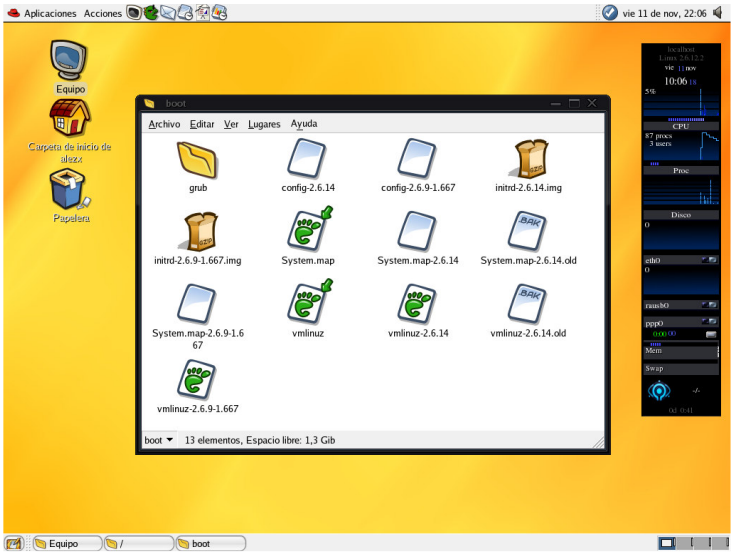


Imagen 6

Ahora vamos a probarlo. Venga, escribimos en la terminal: reboot y esperamos a que reinicie. Una vez que cargue el LILO/GRUB veremos que aparece una nueva opción con el nombre del nuevo kernel. Bien, la seleccionamos y cruzamos los dedos... Ahora pueden ocurrir dos cosas:

a) Nuestro sistema carga sin problemas con el kernel.

b) Kernel Panic. Si aparece un mensaje precedido de “Kernel Panic” y encima has cruzado los dedos es que la suerte no está de tu parte :'(. ¿Qué hacemos? Pues pasar al punto 6.1 ☺.

6.1- Kernel Panics y demás errores

Bueno, este apartado os lo escribo por experiencias propias. Pueden ocurrir por diversos motivos, pero algunos de los más comunes son errores que he repetido en este mismo documento varias veces: no poner el soporte para sistemas de archivos compilado dentro del kernel, así como no compilar tampoco el soporte para discos duros SATA. Si no lo hemos hecho así, el resultado es ¡pánico en el túnel!... digo... en el núcleo :P.

Soluciones: O recompilamos otra vez (aunque no creo que os apetezca) o creamos una imagen Initrd, y eso es lo que vamos a hacer ahora :).

7.- CREANDO UNA IMAGEN INITRD

Cuando colocamos como módulos algunas funciones vitales para el funcionamiento del sistema operativo, como, por ejemplo, el soporte para el sistema de

archivos, debemos crear una imagen Initrd que contiene los módulos de esas funciones vitales y que se carga al iniciar el sistema. Dependiendo de la configuración que le demos al kernel esta imagen puede crearse automáticamente o no. En caso de que la necesitemos (no siempre se necesita) y no se cree, la forma de construirla sería la siguiente:

```
/sbin/mkinitrd [archivo_nueva_imagen_initrd] [version_kernel]
```

Por ejemplo, en mi caso sería, así:

```
/sbin/mkinitrd  /boot/initrd-2.6.14.img  2.6.14
```

Una vez creada hay que añadir una referencia de la imagen Initrd al LILO/GRUB, aunque eso se escaparía de las expectativas de este manual.

8.- NOTAS Y DESPEDIDA

Ya estamos acabando. Ahora deberíamos tener nuestro nuevo kernel compilado y en ejecución. Para comprobar esto escribimos el comando uname -r, que nos dará como resultado la versión de nuestro kernel (***imagen 7***). Si nos diese otra versión, deberíamos ir a la configuración de nuestro gestor de arranque (LILO/GRUB) y asegurarnos de que todo es correcto y de que se hace referencia a nuestro nuevo kernel.

> NOTA: La explicación de dónde se encuentra situada la configuración del gestor de arranque y cómo configurarlo se escapa de las expectativas de este documento. Si tenéis dudas, preguntadle al señor Google o simplemente pasaros por vuestro rincón en el foro de Hack x Crack (http://www.hackxcrack.com/phpBB2/index.php). ¡Os estamos esperando!

El antiguo kernel seguirá estando presente en vuestro PC hasta que digáis lo contrario (en /boot y en la ruta de las fuentes que os indiqué antes). Yo, personalmente, no recomiendo suprimir el antiguo kernel inmediatamente después de compilar uno nuevo. Si queréis borrarlo, podéis hacerlo, pero mejor cuando ya estéis 101% seguros de que el nuevo kernel funciona a la perfección. Y, si lo quitáis, no se os olvide eliminar la referencia hacia ese kernel que hay en el LILO/GRUB, porque que luego hay problemas.

Imagen 7

También quiero dejar claro que, dependiendo de la distro de GNU/Linux que uséis, el proceso de actualización de un kernel puede ser distinto y, normalmente, más simple, pero en este documento explico el método general, que funcionará en cualquier distro.

Y creo que no se me escapa nada más. Mmm... sí, bueno, también se puede actualizar el kernel mediante parches, pequeños archivos que dotan de ligeras funcionalidades a un kernel (resuelven problemas de compatibilidad, etc...), pero, por falta de espacio, no voy a decir nada al respecto de ellos. Quizá, para otra ocasión ;)

Y sin más, señores usuarios del magnífico sistema operativo GNU/Linux, me despido, deseando que este documento haya sido de vuestro agrado y que os haya enseñado (al menos a los usuarios más noveles) cómo funciona esto de los kernels y que, cómo veis, no es tan complejo.

Saludos!!

Por **Alejandro Sánchez Postigo** (también conocido como **aLeZX)**.

Dedicado a mi madre.