Kernel-Como

Brian Ward, bri@blah.math.tu-graz.ac.at
Traducción de Juan José Amor, jjamor@ls.fi.upm.es
Revisión 2.1, 22 de Diciembre de 1996

Lo que sigue es una guía detallada de la configuración del núcleo, cubriendo detalles de compilación y de actualizaciones. El traductor ha intentado respetar el documento original, aunque no ha podido evitar añadir cosas de su propia cosecha, sobre todo en lo que respecta a opciones aún no documentadas.

Índice General

1 Introducción

Esta es la versión 0.75 del documento "*Kernel-HOWTO*" original. La de la traducción es la 2.1 ¿Debe Vd. leer este documento? Bien, veamos si tiene alguno de los siguientes síntomas:

- "¡Jo! El `dosemu-0.99.9` requiere el núcleo `1.8.193` y yo estoy aún con el `1.2.3`"
- Hay un manejador (*driver*) que sólo está en los núcleos más recientes y Vd. lo necesita.
- Vd. realmente no tiene ni idea de cómo se compila el núcleo.
- Vd. lo ha intentado pero no le ha funcionado.

1.1 ¡Lea esto primero!

Algunos de los ejemplos aquí puestos suponen que se tiene el `tar` de *GNU*, así como `find` y `xargs`. Son casi estándares, pero por si acaso... También se asume que conoce la organización del sistema de ficheros; si no lo conoce, copie la salida del comando `mount` en condiciones normales (o el fichero `/etc/fstab` si puede acceder a él). Esta información es importante, y no cambiará a menos que reparticione el disco, añada uno nuevo, reinstale el sistema o similar.

La última versión de *producción* del núcleo que se conoce ahora mismo es la `2.0.10`[1], de manera que nuestros ejemplos irán para esa versión.

[1] `2.0.27` es la última versión estable al día de hoy, 22 de Diciembre de 1996.

Se intenta que tampoco dependa mucho de esto, pero si Vd. usa una versión distinta habrá siempre algunas diferencias, ya que el núcleo está en continuo desarrollo. Esto no debe crear problemas, si acaso un ligero despiste al principio.

Hay dos versiones de las fuentes del núcleo de Linux: las de *producción* y las de *desarrollo*. Las versiones de *producción* empezaron a numerarse en la `1.0.x` y siempre se nombran con número par. Así, se pasó de las `1.0.x` a las `1.2.x` y de ésta a la `2.0.x` (por no ir a la `1.4.x ;)`).

Estas versiones se consideran *estables* y *libres de errores* (hasta el momento). Las versiones de *desarrollo* se numeran en impar (`1.1.x, 1.3.x, 2.1.x`...) y son de prueba, con lo que posiblemente tengan muchos fallos.

1.2 Sobre el estilo

`El texto con el aspecto de esta frase` se corresponderá con las cosas que deban aparecer en la pantalla, un fichero o algo que pueda ser tecleado como opciones de comandos (si está utilizando la versión ASCII de este documento, no verá diferencia alguna). Los comandos son entrecomillados (con ' ') lo que causa un problema con la puntuación: al acabar frases con un comando, el estilo americano pone el punto dentro de las comillas, lo que confundirá a algún lector creyendo que hay que teclear también el punto (algo de sentido común, aunque en realidad pretenden que el "*sentido común*" sea solo el estilo americano de puntuación). Esto se soluciona no poniendo tal punto, aunque no sea correcto ortográficamente. Por ejemplo, pondremos '`make config`' y no '`make config.`'

2 Preguntas y respuestas importantes

2.1 Pero, ¿qué hace el núcleo?

El núcleo de *Unix* actúa como intermediario entre sus programas y el ordenador. En primer lugar, gestiona la memoria de todos los programas o procesos, y se asegura de que se reparten los ciclos del procesador. Además, proporciona una interfaz portable para que los programas `hablen` fácilmente con su hardware.

Realmente, el núcleo hace más cosas, pero las anteriores son las más importantes.

2.2 ¿Por qué puedo necesitar actualizar el núcleo?

Los núcleos nuevos normalmente ofrecen la posibilidad de entenderse con más accesorios hardware (o sea, incluyen más manejadores), se ejecutan más rápidamente, son más estables o corrigen errores de otras versiones. Mucha gente se actualiza el núcleo para poder usar nuevos manejadores que necesitan o librarse de "*bugs*" de la versión que usaban.

2.3 ¿Qué hardware nuevo soportan los nuevos núcleos?

Véanse el documento *Hardware-HOWTO*, el fichero `config.in` de las fuentes o simplemente verá lo que hay cuando haga `'make config'`. Esto le mostrará lo soportado por la distribución estándar, pero no todo, pues algunos son módulos cargables en ejecución (como manejadores de PCMCIA) y se mantienen y distribuyen por vías distintas.

2.3.1 Anexo de la revisión.

Actualmente, muy pocos módulos (los *no-oficiales*) no están incluidos en los fuentes *oficiales* del kernel. Prácticamente, todos los controladores son *modularizables*.

He aquí un extracto de lo que aparece en `/usr/src/linux/Documentation/modules.txt`, algo que se **debe** leer:[2]:

```
La mayoria de los sistemas de ficheros:

minix, xiafs, msdos, umsdos, sysv, isofs, hpfs, smbfs, nfs.

 Soporte SCSI de medio nivel (requeridos por controladores SCSI de
 alto y bajo nivel)

La mayoria de los controladores SCSI de bajo nivel: (i.e. aha1542, in2000)
Todos los controladores SCSI de alto nivel: disk, tape, cdrom, generic.

La mayoria de los controladores ethernet: (demasiados para enumerarlos todos, ver:
             /usr/src/linux/Documentation/networking/net-modules.txt)

La mayoria de los controladores de CDROM (propietarios):
        aztcd:     Aztech,Orchid,Okano,Wearnes
        cm206:     Philips/LMS CM206
        gscd:      Goldstar GCDR-420
        mcd, mcdx: Mitsumi LU005, FX001
        optcd:     Optics Storage Dolphin 8000AT
        sjcd:      Sanyo CDR-H94A
        sbpcd:     Matsushita/Panasonic CR52x, CR56x, CD200,
                   Longshine LCS-7260, TEAC CD-55A
        sonycd535: Sony CDU-531/535, CDU-510/515

Y un monton de modulos, como:
        lp: impresora p. paralelo
        binfmt_elf: cargador elf
        binfmt_java: cargador java
        isp16: interfaz cdrom
        serial: interfaz serial (tty)
```

2.4 ¿Qué versión de `gcc` y `libc` necesito?

Linus recomienda una en el fichero `README` de las fuentes del núcleo. Si no la tiene, probablemente tenga que actualizarse las librerías `libc`, algo que tampoco es difícil.

2.5 ¿Qué es un módulo cargable?

Hay partes del código del núcleo que no se enlazan directamente en el núcleo. Se compilan por separado y luego se incorporan al núcleo que ya está corriendo. Es la forma preferida de usar algunos manejadores como los de dispositivos `PCMCIA` o de cintas `QIC-80/40`[3].

2.6 ¿Cuánto espacio en disco necesito?

Depende de su configuración. La versión `2.0.10` del núcleo ocupa, comprimida, 6 *megabytes*, pero al descomprimir ocupará unos 24 MB. Pero aquí no acaba la cosa: para compilar se necesita espacio para

[2] Hoy mismo se liberará una traducción.

[3] Hoy en día la tendencia es modulizar todos los controladores posibles, salvo aquellos estrictamente necesarios para arrancar (`ext2`, `ide`, o `SCSI`, según el caso), aunque siempre se puede hacer una imagen `initrd` y se sigue pudiendo arrancar con `LILO`. `kerneld` se encargará de cargarlos y descargarlos por nosotros cuando el sistema los necesite o no.

ficheros temporales, dependiendo de la configuración que se elija. Por ejemplo, en un 386, con manejador de red de *3Com* y cinco sistemas de ficheros supone 30 MB. Si a esto añadimos las fuentes comprimidas, serán 36 MB. En otro sistema con menos cosas puede ocupar menos. Además, un núcleo nuevo tendrá por lo general muchos más ficheros que el antiguo, con lo que debe asegurarse de que hay espacio de disco suficiente (además, con los precios que tienen ahora los discos, me puedo permitir el recomendarle que se compre un disco nuevo para el Linux).

2.7 ¿Cuánto se tarda en compilar?

Para mucha gente, se tarda "*muchísimo*". La velocidad del sistema y la cantidad de memoria son determinantes. En un *486DX4/100* con 16 MB, se tarda unos 20 minutos en compilar el núcleo v1.2 con cinco sistemas de ficheros, soporte de red y manejadores de tarjetas de sonido. En un *386DX/40* con 8 MB de RAM se tarda una hora y media. Por lo general se recomienda prepararse un café o ver la televisión (o lo que se le ocurra[4]) mientras se compila el núcleo. Otra posibilidad es compilarlo en un PC más rápido que el suyo, de algún amigo.

3 Cómo configurar el núcleo.

3.1 Obtención de los fuentes.

Se pueden obtener las fuentes por FTP anónimo de `ftp://ftp.funet.fi/pub/OS/Linux/PEOPLE/Linus/`, en un "*mirror*"[5] o en otros servidores. Típicamente tienen el nombre `linux-x.y.z.tar.gz`, donde `x.y.z` es el número de versión. Las versiones se encuentran en directorios `v1.1`, `v1.2` y `v1.3`[6]. El número mayor es el de la última versión, en este caso versión de desarrollo. Le sugerimos que utilice un servidor "*mirror*" de `ftp.funet.fi`, por ejemplo:

- EE.UU.: `ftp://tsx-11.mit.edu/pub/linux/sources/system`
- EE.UU.: `ftp://sunsite.unc.edu/pub/Linux/kernel`
- Inglaterra: `ftp://sunsite.doc.ic.ac.uk/pub/unix/Linux/sunsite.unc-mirror/kernel/`
- Austria: `ftp://fvkma.tu-graz.ac.at/pub/linux/linus`
- Alemania: `ftp://ftp.Germany.EU.net/pub/os/Linux/Local.EUnet/Kernel/Linus`
- Alemania: `ftp://ftp.dfv.rwth-aachen.de/pub/linux/kernel`
- Francia: `ftp://ftp.ibp.fr/pub/linux/sources/system/patches`
- Australia: `tp://kirk.bond.edu.au/pub/OS/Linux/kernel`
- España: `ftp://sunsite.rediris.es/pub/linux/kernel/sources`

Si no tiene acceso `FTP`, encontrará una lista de *BBS* con lo mismo en el grupo `comp.os.linux.announce` de USENET News.

[4] O ponerse a hacer cualquier cosa; para eso estamos en multitarea `:-)`
[5] En España contamos con el veloz y magnífico `ftp://sunsite.rediris.es`
[6] Y `2.0` y `2.1`

3.2 Descompresión de los fuentes

Entre en la cuenta `root` y vaya al directorio `/usr/src`. Si instaló las fuentes cuando instaló Linux, habrá aquí un directorio `linux`, con las fuentes antiguas. Si quiere preservar esas fuentes (y tiene espacio de sobra), renombre (con `mv`) ese directorio a `linux-x.y.z` (la versión actual `x.y.z` la puede obtener con el comando `uname -r`). Es importante que antes de descomprimir las nuevas fuentes no haya ningún directorio `/usr/src/linux`.

Ahora, en `/usr/src`, descomprima las fuentes con

```
tar xvfzp linux-x.y.z.tar.gz
```

(si sólo tiene un fichero `.tar`, sin `.gz`, elimine la `z` del comando `tar`). Los contenidos del fichero se expandirán en `/usr/src/linux`. Vaya ahora a ese directorio y lea el fichero `README`. Encontrará una sección "*INSTALLING the kernel*". Siga las instrucciones que allí se digan (aunque en las siguientes líneas le diremos cómo tiene que hacer casi todo esto).

3.3 Configuración del núcleo

Nota: Este texto es parcialmente una repetición de lo que hay en el fichero `README` de las fuentes.

El comando

```
make config
```

invocado en el directorio `/usr/src/linux` inicia un programa *shell-script* de configuración que le preguntará bastantes cosas. Requiere que esté instalado `bash`, con lo que debe existir `/bin/bash`, `/bin/sh` o la variable `$BASH`.

3.3.1 Anexo de la revisión 2.1.

Si los fuentes que posee son superiores al `2.0.x`, existen dos métodos bastante más agradables e interactivos de llevar a cabo la configuración; invocando desde modo texto, en el mismo directorio mencionado anteriormente

```
make menuconfig
```

Iniciará un colorido programa (escrito con `lxdialog`) muy agradable y bastante más funcional que el `make config` a secas, para llevar a cabo la configuración del kernel.

Observará que su uso es bastante intuitivo; las opciones con un $< >$ indican la posibilidad de modularizar; si en ella presionamos `M` la seleccionaremos como módulo. Si presionamos `ESPACIO` aparecerá un `*` indicando que será incluido en el kernel. Finalmente, si presionamos `N` lo deseleccionaremos.

Las opciones marcadas con `[ ]` no son modularizables.

Si Vd. es un *fan* de las X, entonces puede arrancar desde un `xterm` el `script` equivalente, ejecutando

```
make xconfig
```

Aparecerá ante sus ojos un maravilloso y gráfico programa de selección y configuración del kernel bajo *X Window* `;-)`.

Aquí seguro que sabrá manejarse a golpe de ratón sin problema alguno; las opciones modularizables tendrán tres casillas posibles a rellenar: `y` indica que se incluirá en la parte monolítica del kernel; `m` que lo hará como módulo, y `n` que no será seleccionado.

3.4 Continuación

Ahora [7]se trata de contestar las preguntas, normalmente con sí (`y`) o no (`n`). Los manejadores de dispositivos incorporan, además, la opción de módulo (`m`), opción interesante puesto que no incorpora el manejador al núcleo en tiempo de compilación, sino que se hace durante la ejecución del mismo, cuando sea necesario (por ejemplo, cargar el módulo *PPP* solo cuando vayamos a usar el módem).

Algunas otras opciones obvias o poco importantes no son descritas aquí. Vea la sección 3.4.12 para descripciones de otras muchas.

Además, cuando configure el núcleo 2.0.x, podrá poner la opción `?`, lo que le dará una breve descripción de la misma.

3.4.1 Emulación de coprocesador (`Kernel math emulation`)

Si no tiene coprocesador matemático (o un 486), debe contestar `y` a esto. Si tiene coprocesador y contesta `y` la emulación será instalada pero no usada; funcionará, pero le ocupará memoria innecesariamente. Además, he oído que la emulación es lenta, lo que se traducirá en unas pesadas *X-Window*.

3.4.2 Soporte de discos IDE y MFM/RLL normales (`Normal (MFM/RLL) disk and IDE disk/cdrom support`)

Normalmente necesitará este soporte, pues es el de los discos habituales en los PCs. Este manejador no gestiona discos *SCSI*, para eso está la opción específica que luego veremos.

Ahora se le preguntará si quiere usar "`old disk-only`" y manejadores "`new IDE`". Tiene que elegir uno de ellos. La diferencia principal es que el nuevo (`new IDE`) maneja controladoras secundarias y CD-ROMs *IDE/ATAPI*, hasta cuatro dispositivos, pero ocupa 4 KB más que el antiguo. Se supone que los nuevos manejadores mejoran también velocidad y fiabilidad, en especial si dispone de dispositivos *EIDE*.

3.4.3 Soporte de redes (`Networking support`)

En principio, contestaría afirmativamente si la máquina estuviera en una red local o conectada a la Internet con `SLIP/PPP`. Sin embargo, muchos programas, como ocurre con el sistema X Window, requieren el soporte de red, así que lo mejor es contestar `y` siempre. Después se le preguntará si quiere soporte `TCP/IP`: conteste afirmativamente aun no estando seguro.

3.4.4 Limitar memoria a 16 Mb (`Limit memory to low 16MB`)

Hay controladores de DMA en 386 con problemas para direccionar más de 16 MB de RAM. Contestar `y` en el caso (raro) de que tenga uno de éstos.

3.4.5 Comunicación entre procesos *System V* (`System V IPC`)

Una de las mejores definiciones del IPC está en el glosario del libro del Perl. Curiosamente, los programadores de Perl suelen usar IPC para comunicar sus procesos, así como muchas otras aplicaciones (como `DOOM`, sobre todo), por lo que no es buena idea contestar `n` si no se está seguro de lo que hace.

[7]Lo que sigue está dirigido a los usuarios (*¿masoquistas?* del `make config`, si emplea `make menuconfig` o `make xconfig` encontrará mucho más intuitiva la configuración, si bien debe leer a continuación qué significan las opciones.

3.4.6 Tipo de CPU (386, 486, Pentium, PPro) (`Processor type (386, 486, Pentium, PPro)`)

Nota: En anteriores núcleos, la opción era Ӧsar el parámetro `-m486` en el compilador" (`Use -m486 flag for 486-specific optimizations`)

Tradicionalmente, esto era una optimización compatible con todos los chips, aunque más rápido en la CPU elegida (y en todo caso, ocupaba algo más). Actualmente, esto puede no ser cierto, con lo que deberá elegir siempre la CPU que tenga (aunque la versión de 386 funcionará en todas las máquinas).

3.4.7 Soporte SCSI (`SCSI support`)

Si tiene dispositivos SCSI, conteste '`y`'. Se le preguntarán entonces más cosas: soporte de CD-ROM, discos y qué clase de adaptador utiliza. Vea el documento *SCSI-HOWTO* para más detalle.

3.4.8 Soportes de tarjetas de red (`Network device support`)

Si tiene tarjeta de red, o quiere usar `PPP`, `SLIP` o `PLIP` (puerto paralelo usado para conectarse a Internet), conteste afirmativamente. El script le preguntará ahora qué tarjeta tiene y qué protocolo usar.

3.4.9 Sistemas de ficheros (`Filesystems`)

El programa le preguntará si quiere soporte para los siguientes sistemas de ficheros:

`Standard (minix)` - Las nuevas distribuciones no crean sistemas Minix, y mucha gente no las usa para nada, pero puede convenir elegirla. Algunos "*discos de rescate*" usan el sistema Minix como formato de sus disquetes, ya que da menos problemas.

`Extended fs` - Fue la primera versión del sistema extendido, pero ya no se usa. No necesita seleccionarla.

`Second extended` - Este es el sistema más usado. Seguramente lo usará Vd. también. Seleccione '`y`'.[8]

`xiafs filesystem` - Hace mucho pudo ser usado, pero ahora mismo no conocemos a nadie que lo utilice.

`msdos` - Selecciónelo si quiere acceder desde Linux a sus particiones de MS-DOS o quiere montar disquetes de ese sistema.

`umsdos` - Este sistema permite añadir a un sistema MS-DOS las características típicas de Unix como nombres largos o enlaces. Sirve para quienes quieran instalar Linux en la partición DOS, pero nada más.

`/proc` - Uno de los grandes inventos desde la leche condensada. (una idea robada descaradamente a *Bell Labs*, si no me equivoco) No se trata de algo que se guarde en el disco, sino de una interfaz por medio de ficheros con la tabla de procesos del núcleo, usada por programas como '`ps`'. Pruebe a teclear '`cat /proc/meminfo`' o '`cat /proc/devices`'. Algunos shells (`rc`, concretamente) hacen uso de `/proc/self/fd` (en lugar de `/dev/fd`) para E/S. Debe contestar afirmativamente a la pregunta: muchas utilidades de Linux lo necesitan.

`NFS` - Si su máquina está en red y quiere acceder a sistemas de ficheros remotos con NFS, conteste '`y`' a esta pregunta.

`ISO9660` - Típico en los CD-ROMs. Si tiene un CD-ROM, seleccione esta opción.

`OS/2 HPFS` - En el momento de escribir esto, el soporte HPFS es de sólo lectura.[9]

`System V and Coherent` - para particiones de sistemas Coherent y System V (otras variantes de Unix para PC).

[8]Ojo con seleccionarlo como módulo; aunque se puede, el sistema de ficheros en el que esté su partición `/` debe estar incluido el el kernel *no-modularmente*, ya que de lo contrario no podrá arrancar.

[9]Y por lo que se "oye", así se quedará...

Adiciones desde la edición del *Howto* original (Rev. 2.1):

`quota support` - Selecciónelo si quiere administrar el espacio que consumen en disco los usuarios del sistema; necesitará además las utilidades `quotas-1.55`, disponibles en:

`ftp://ftp.funet.fi/pub/Linux/kernel/src/subsystems/quota/all.tar.gz`

Tenga en cuenta que este tipo de control restrictivo sólo funcionará sobre sistemas de ficheros `ext2`.

`mandatory lock support`: Cambia el algoritmo para proporcionar un sistema de bloqueos más seguro, a fin de evitar la corrupción de ficheros por parte de las aplicaciones en los accesos a disco simultáneo. Esto puede interesar probable y únicamente cuando se usen grandes bases de datos o dedique su sistema Linux a servidor de ficheros. Tenga en cuenta que deberá tener servidores (`samba`, `mars-nwe`, etc...) que soporten esta nueva característica.

`vfat`. En conjunción con el sistema de fichero `fat` o `msdos` nos permitirá acceso a las particiones de w95, con soporte de nombres "largos" y demás. En el momento de escribir esto, se desconoce si soportará la "nueva" *fat32* del recién aparecido w95-*OSR2*.

`SMB filesystem support`: En conjunción con el paquete `samba`[10] nos permitirá montar volúmenes compartidos por clientes de red *NetBios*.

`SMB w95 bug workaround` - `:-)` qué decir... corrije un bug que experimentan los volúmenes exportados por servidores w95.

`Amiga FFS support` - Soporte para el sistema de ficheros *Fast File System* de *Amiga*.

`UFS filesystem support` - Soporte para el *Unix FileSystem*, empleado por algunos *nixes BSD, como *SunOS, FreeBSD, NetBSD, NeXTstep*. Permite montar dichas particiones o disquetes en modo de sólo lectura.

Esta opción contiene otras dos opciones adicionales relacionadas con las tablas de partición de los discos según provengan de uno u otro *nix. Escoja el menú en línea `help` para mayor información.

`NCP filesystem support` - Permite montar volúmenes *NetWare*. Necesitará programas cliente para llevarlo a cabo, no obstante.

`root filesystem on NFS` - necesario para estaciones de trabajo sin disco duro, que arranquen por red y monten todo su sistema de ficheros por *NFS*.

`BOOTP support` - Si su caso es el anterior, y arranca por red, este protocolo le permitirá obtener su `IP` dinámicamente, usando el protocolo `BOOTP`. (Siempre que su tarjeta de red incorpore una *ROM* que lo permita).

`RARP support` - Idem de lo anterior, en este caso usando el protocolo *RARP*.

`Sys V and Coherent filesystem support` permitirá montar particiones de otros *nixes System V, como *SCO Unix*, *Xenix* y *Coherent*. Lea la ayuda asociada.

Actualmente, hay un módulo *no oficial* que permite acceso a sistemas de ficheros *NTFS*, usados por wNT, su autor es Martin von Löwis, `loewis@informatik.hu-berlin.de`. Su nivel de soporte es ciertamente rudimentario todavía, pero siempre es bueno saberlo...

más información en `http://www.informatik.hu-berlin.de:80/~loewis/ntfs`

¡Pero no sé qué sistemas de ficheros necesito! De acuerdo, teclee '`mount`'. La salida será como esta:

```
blah# mount
/dev/hda1 on / type ext2 (defaults)
/dev/hda3 on /usr type ext2 (defaults)
none on /proc type proc (defaults)
```

[10] Cuyo COMO está disponible en castellano, mire en la sección 14 para más información.

```
/dev/fd0 on /mnt type msdos (defaults)
```

Observe cada línea; la palabra que sigue a '`type`' es el tipo de sistema usado. En este ejemplo, los sistemas raíz (/) y `/usr` usan el sistema "*second extended*", además usamos sistema "*proc*" y hay montado un disquete MS-DOS.

Para saber qué sistemas de ficheros usa en el núcleo actual puede teclear '`cat /proc/filesystems`' siempre y cuando tenga al menos soporte `proc`.

Otros sistemas de ficheros más raramente usados ocupan también mucho. Quizás le interese meterlos como módulos (vea la correspondiente sección, y vea también la sección 7.2 para saber por qué no se recomiendan núcleos que ocupen mucho).

3.4.10 Manejadores de tipo carácter (`Character devices`)

Aquí se activan los manejadores para impresoras (por puerto paralelo), ratones de bus, ratones *PS/2* (usado también en muchos *trackballs* de los portátiles), algunos manejadores de cintas y otros. Conteste según su hardware.

Nota: "*Selection*" es un programa que le permite usar el ratón para cortar y pegar texto entre consolas virtuales, fuera de las X-Window. Actualmente no es una opción de configuración, sino un estándar.

Nota 2: 'Selection' está bastante obsoleto. Es mucho mejor el programa "`gpm`"[11] puesto que hace cosas como manejar diversos protocolos de ratones, etc.

3.4.11 Tarjeta de sonido (`Sound card`)

Si siente gran necesidad de oír el ladrido del '`biff`', conteste '`y`' y después otro programa de configuración le preguntará acerca de la tarjeta de sonido que tiene. Una observación: cuando le pregunte si quiere instalar la versión completa del manejador, conteste '`n`' y ahorrará memoria, aunque tendrá que elegir qué características desea incluir. Le recomiendo aquí que lea el *Sonido-Como*[12] donde encontrará mucha más información.

3.4.12 Otras opciones de configuración

No todas las opciones se han listado ya que cambian mucho y otras son completamente evidentes (por ejemplo, el soporte `3Com 3C509`). Hay una lista más extensa en el siguiente *URL* de Axel Boldt, `axel@uni-paderborn.de`:

`http://math-www.uni-paderborn.de/~axel/config_help.html`

o por *FTP* anónimo en:

`ftp://sunsite.unc.edu/pub/Linux/kernel/config/krnl_cnfg_hlp.x.yz.tgz`

donde `x.yz` se refiere a la versión del núcleo. En las últimas versiones (`2.0.x`) se incluyen estos documentos entre los ficheros de las fuentes del núcleo (directorio `Documentation/`...).

3.4.13 `Kernel hacking`

Traducción del fichero `README` de Linus:

La configuración "`kernel hacking`" suele dar lugar a un núcleo más grande y/o lento, y puede hacerlo menos estable al configurar algunas rutinas especiales para encontrar errores del núcleo (`kmalloc()`). Para un núcleo de "*producción*" deberá contestar '`n`' a la pregunta.

[11] *General Purpose Mouse support*, *Soporte Genérico para Ratones*

[12] Disponible en castellano, obviamente

3.5 ¿Y ahora qué? (El fichero `Makefile`).

Tras hacer `make config`, un mensaje le dirá que ya está preparado el núcleo y que "revise el fichero `Makefile` para opciones adicionales", etc.

Por tanto, veamos el `Makefile`. Probablemente no tendrá que cambiar nada, pero debe mirarse. Sus opciones podrán cambiarse con el comando '`rdev`' una vez que el núcleo esté compilado.

4 Compilación del núcleo

4.1 Limpieza y dependencias

Cuando se termina de configurar, se le instará a ejecutar '`make dep`' y '`clean`'. Haga '`make dep`', lo que preparará las dependencias en poco tiempo, a menos que su PC sea muy lento. Cuando acabe, haga '`make clean`'. Esto elimina ficheros objetos y demás de la versión anterior. *No olvidar* este paso.

4.2 El momento de compilar

Después de preparar dependencias, puede ejecutar '`make zImage`' o '`make zdisk`' (esta es la parte que tarda más tiempo). '`make zImage`' compilará el núcleo y lo dejará comprimido en `arch/i386/boot/zImage` junto a otros ficheros. Con '`make zdisk`' el nuevo núcleo se copiará además en el disquete que esté puesto en la disquetera "A:". '`zdisk`' es interesante para probar núcleos; si explota (o simplemente no hace nada) se quita el disquete de la disquetera y podrá arrancar el núcleo antiguo. Además sirve para arrancar si borró accidentalmente el núcleo del disco duro. También puede usarlo para instalar nuevos sistemas simplemente volcando el contenido de un disco en otro ("¡¡ todo esto y más !! Ahora, ¿cuánto pagaría?").

Los núcleos recientes están comprimidos, con una '`z`' comenzando su nombre. Un núcleo comprimido se descomprime automáticamente al ser ejecutado.

4.3 Otras opciones del '`make`'

Con '`make mrproper`' hará una limpieza mucho más "intensa". Suele hacer falta cuando se actualiza (parchea) el núcleo. Pero esta opción borra también su fichero de configuración del núcleo, así que guarde una copia del correspondiente fichero `.config` si cree que le interesa.

La opción '`make oldconfig`' intentará configurar el núcleo con un fichero de configuración anterior[13] evitando todo el proceso del '`make config`'. Si no ha compilado anteriormente el núcleo o no tiene un fichero de configuración anterior, no lo elija pues normalmente querrá algo que se salga de la configuración por defecto.

Para compilación con módulos, vea la sección correspondiente.

4.4 Instalación del núcleo

Una vez que tenga un nuevo núcleo que parezca funcionar como desea, será el momento de instalarlo. Casi todo el mundo utiliza `LILO` (LInux LOader) para esto. Con '`make zlilo`' se instalará el núcleo ejecutando `LILO`, quedando listo para rearrancar, pero esto solo funcionará si `LILO` está bien configurado para su sistema: el núcleo es `/vmlinuz`, `LILO` está en `/sbin` y la configuración de `LILO` (`/etc/lilo.conf`) es coherente con lo anterior.

En otro caso, necesitará usar `LILO` directamente. Hay un paquete que lo instala de manera adecuada, pero su fichero de configuración tiende a confundir a la gente. Observe el fichero de configuración (estará en

[13] A partir del `2.0.xx` no es necesario, `make` *recuerda* la última configuración

`/etc/lilo.conf` o en `/etc/lilo/config` para versiones más antiguas), y vea cuál es la configuración actual. El fichero de configuración será como éste:

```
image = /vmlinuz
        label = Linux
        root = /dev/hda1
        ...
```

La línea '`image =`' apunta al núcleo instalado actualmente. Casi siempre es `/vmlinuz`. '`label`' es el identificador usado para seleccionar qué sistema arrancar, y '`root`' es el disco o partición a usar para el directorio raíz. Haga una copia de seguridad de su antiguo núcleo y copie en `/vmlinuz` o donde diga el fichero anterior el fichero zImage que haya generado el proceso de compilación. Ahora, ejecute `LILO` (en sistemas modernos, será simplemente teclear '`lilo`'. En sistemas antiguos, habrá que poner '`/etc/lilo/install`' o '`/etc/lilo/lilo -C /etc/lilo/config`'.)

Si quiere saber más sobre la configuración de `LILO`, o no tiene `LILO`, obtenga la versión más reciente de su servidor `FTP` favorito y siga las instrucciones que le acompañan.

Para arrancar uno de sus antiguos núcleos, copie las líneas anteriores incluyendo '`image = xxx`' al principio del fichero de configuración de `LILO`, y cambie '`image = xxx`' por '`image = yyy`' donde '`yyy`' es el nombre de camino completo al fichero de la copia de seguridad guardada. Ahora, cambie '`label = zzz`' por '`label = linux-backup`' y reejecute `LILO`. Puede ser que necesite poner una línea en el fichero con '`delay=x`' donde x son las centésimas de segundo que `LILO` esperará antes de arrancar con la primera opción, de modo que pueda interrumpirse (con la tecla `SHIFT`) y seleccionarse qué núcleo desea arrancar (tecleando la etiqueta (`label`) asignada).

Inciso de la revisión 2.1

Una forma muy cómoda de llevar todo el tema del `LILO`, si lo tenemos instalado, y las compilaciones, etc, es añadir lo siguiente en el `/etc/lilo.conf`:

```
...
image=/vmlinuz
        label=ultimo
        root=/dev/hd[loquesea]
        read-only
        append = ""
image=/vmlinuz.old
        label=anterior
        root=/dev/hd[loquesea]
        read-only
        append = ""
```

Al compilar, si lo hacemos con la secuencia de comandos

```
# make dep; make clean; make zlilo; make modules; make modules_install
```

el `make zlilo` renombrará la anterior imagen del kernel a `/vmlinuz.old` , dejando la nueva como `/vmlinuz`, e instalará `LILO`, a continuación con lo cual lo hacemos todo *automágicamente*.

La órdenes `make modules; make modules_install` compilarán los módulos que hayamos seleccionado, y los instalarán. No olvidar ejecutar `depmod -a` en cuanto hayamos arrancado con dicho núcleo.

En caso de que estemos compilando por segunda vez una misma versión de núcleo, y hayamos variado el número de módulos a compilar, es posible que la ejecutar `make dep` nos encontremos con un mensaje de error; esto se debe a que los antiguos módulos que no hayamos compilado ahora no son borrados. Pueden borrarse tranquilamente.

5 Parchear el núcleo

5.1 Aplicación de un parche

Las actualizaciones incrementales del núcleo se distribuyen como parches[14] Por ejemplo, si tiene la versión `1.1.45` y ve que existe un parche '`patch46.gz`', con ese fichero podrá actualizarse a la `1.1.46`. Debería antes de nada guardar una copia del árbol de directorios de las fuentes del núcleo actual (haciendo '`make clean`', luego '`tar cvfz antiguas-fuentes.tar.gz linux`' desde el directorio `/usr/src`).

Ahora, supongamos que tiene '`patch46.gz`' en `/usr/src`. Vaya a ese directorio y escriba

```
zcat patch46.gz | patch -p0
```

(o bien

```
patch -p0 < patch46
```

si ya estaba descomprimido). Verá rápida (o lentamente, depende del ordenador) una serie de mensajes que le dicen que se intentan aplicar los cambios, cuáles tienen éxito y cuáles no. Normalmente, esto irá bien y no habrá que preocuparse de tanto mensaje, aunque con la opción `-s` solo saldrán los mensajes de error.

Inciso de la revisión 2.1:

No es infrecuente que haya que borrar el árbol de los fuentes entero y reinstalarlos de nuevo; muchas veces, se tiene un `.tar.gz` con los fuentes más un montón de parches; para evitarse el tener que parchear uno a uno, puede invocar lo siguiente desde la línea de comandos: (asumo que tratamos de parchear p. ej. con parches a partir del `2.0.20` hasta el `2.0.27`, usamos `bash` y estamos en `/usr/src`, teclee `pwd` para cerciorarse).

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

Investigue por su cuenta, verá qué potencia hay en la línea de comandos *nix...

fin del inciso

Para ver qué partes no se han modificado correctamente, busque los ficheros `.rej` en el directorio de las fuentes. Si se usan algunas versiones de `patch` (antiguas, sobre todo) esos ficheros tendrán extensión '`#`'. Con el comando `find` encontrará fácilmente los ficheros:

```
find . -name '*.rej' -print
```

imprime todos los ficheros `.rej` que están en el directorio actual o subdirectorios.

Si todo ha ido bien, haga '`make clean`', '`config`' y '`dep`' como se describió en las secciones 4 y 4.2.

Hay algunas opciones más en el comando `patch`. Con `-s`, como hemos dicho, se suprimen todos los mensajes salvo los errores. Si guarda las fuentes del núcleo en otro lugar que no sea `/usr/src/linux`, con `patch -p1` se parchearán las cosas limpiamente. Otras opciones de interés se encuentran bien documentadas en las páginas `man`.

[14]No tomar con sentido peyorativo el término "*parche*", no se trata de un "*remiendo*", o "*chapuza*", lo que hacemos al "*parchear*" es modificar directamente los **fuentes** del núcleo, incluyendo las variaciones que se hayan introducido.ver el comando `diff` y `patch` para más referencias.

5.2 Si algo va mal

El problema más común es que una ejecución de `patch` intente modificar el fichero '`config.in`' y no parezca quedar bien, porque haya hecho cambios en él de acuerdo con su máquina. Este problema sucede con versiones antiguas. Para corregirlo, busque el fichero `config.in.rej`, y vea qué tiene el parche originado. Los cambios suelen ir marcados con '`+`' o '`-`' al principio de las líneas. Edítelo, recordando si las opciones estaban puestas a `Y` o a `N`, y ejecute

```
patch -p0 < config.in.rej
```

y si no tiene fallos, puede continuar con el resto del proceso. El fichero `config.in.rej` permanecerá, aunque puede borrarlo.

Si encuentra otros problemas, puede que haya aplicado un parche fuera de orden. Si el programa `patch` responde con

```
previously applied patch detected: Assume -R?
```

probablemente estará intentando aplicar un parche anterior a su versión actual. Si responde '`y`', intentará degradar sus fuentes, y normalmente fallará, obligándole a preparar un nuevo árbol de fuentes.

Para anular el efecto de un parche, use '`patch -R`' con el parche original.

Lo mejor ante un problema es reinstalar un árbol de fuentes limpio y empezar de nuevo.

5.3 Limpieza de ficheros `.orig`

Después de algunos parches, los ficheros `.orig` empezarán a abultar mucho. Por ejemplo, si estamos ya con el núcleo 1.1.51 y empezamos con el `1.1.48`, borrar los `.orig` nos ahorrara medio megabyte. Con

```
find .-name '*.orig' -exec rm -f {} ';'
```

se automatizará esa limpieza. En versiones que usen el `#` en lugar de `.rej`, sustituya el `.orig` por un '`~`'.

Una forma mejor de hacer esto es usar `xargs` de *GNU*:

```
find .-name '*.orig' | xargs rm
```

o el método "más seguro pero más pesado":

```
find . -name '*.orig' -print0 | xargs --null rm --
```

5.4 Otros parches

Hay otros parches (los llamaremos "no estándares") que si se aplican, probablemente provocarán que los parches de Linus no funcionen correctamente, teniendo que retroceder, corregir las fuentes o el parche, instalar de nuevo las fuentes, o una combinación de lo anterior.

inciso de la R. 2.1: Existe una página con información centralizada sobre este tipo de parches en:

`http://www.ecsnet.com/html/linux_21_upatch.html`

Este es un buen sitio para buscar si necesita soporte para algún dispositivo esotérico o implementación reciente. Tenga en cuenta, no obstante, que la mayoría serán para el kernel actual de desarrollo, aunque puedan aplicarse algunos (eiste otra página en la misma localización citada anteriormente) en el kernel de producción.

fin del inciso

Por ejemplo, el autor utilizaba el parche '`noblink`' que anula el parpadeo del cursor en las consolas virtuales. Este parche se actualiza (o actualizaba) frecuentemente para los nuevos núcleos. Como ahora muchos manejadores se pueden cargar como módulos, los parches ya son menos necesarios.

6 Paquetes adicionales

El núcleo de Linux tiene más características no documentadas en el código; éstas se utilizan normalmente vía parches no estándares. Algunas características se listan a continuación.

6.1 `kbd`

La consola de Linux tiene más prestaciones que las conocidas, como cambiar las fuentes, cambiar el modo de vídeo, etc. El paquete `kbd` tiene programas que permiten al usuario hacer esto, junto con nuevas fuentes de texto y mapas de teclado, y se encuentra disponible en los mismos sitios donde están las fuentes del núcleo.

6.2 `util-linux`

Rik Faith, `faith@cs.unc.edu` recopiló una gran colección de utilidades de Linux conocidas como '`util-linux`', que ahora mantiene Nicolai Langfeldt, `util-linux@math.uio.no`. Se encuentran en `ftp://sunsite.unc.edu/pub/Linux/system/Misc`, con programas como `setterm`, `rdev` y `ctrlaltdel`, importantes para el núcleo. Como dice Rik, no lo instale sin pensarlo, instalarlo todo sin necesidad puede causarle después algunos problemas.

6.3 `hdparm`

Son programas hechos con el núcleo oficial, para optimizar y jugar con con los parámetros del disco duro, distribuyéndose por separado.

7 Problemas típicos

7.1 `make clean`

Si su nuevo núcleo hace cosas raras, puede ser que se le olvidara hacer '`make clean`' antes de compilar. Los síntomas suelen ser extraños cuelgues, problemas de E/S... asegúrese también de hacer '`make dep`'.

7.2 Núcleos muy lentos o muy grandes

Si su núcleo consume mucha memoria, o la compilación se hace eterna incluso con su nuevo *786DX6/440*, probablemente se deberá a haber elegido demasiados manejadores, sistemas de ficheros, etc. a soportar en el sistema. Si no los va a usar, no los incluya, puesto que consumen memoria. Lo típico en estos casos es que se recurre demasiado al intercambio con el disco, lo que se aprecia en un ruido 'excesivo' del disco duro.

Puede ver cuánta memoria ocupa su núcleo comparando los valores obtenidos al ver el fichero `/proc/meminfo`, o con el comando `dmesg` o el log de los mensajes del núcleo, donde verá algo parecido a esto:

```
Memory: 15124k/16384k available (552k kernel code, 384k reserved, 324k data)
```

En mi 386 (con pocos drivers configurados) sale:

```
Memory: 7000k/8192k available (496k kernel code, 384k reserved, 312k data)
```

7.3 El núcleo no compila

Si no compila, puede ser por un fallo de parcheo, u otro tipo de corrupción en los ficheros fuente. Además, la versión de `gcc` puede no ser correcta o los propios ficheros de `#include`. Asegúrese que los enlaces simbólicos necesarios (descritos en el fichero `README` de Linus) existen. En general, cuando un núcleo estándar no se puede compilar, es porque hay algún problema serio en el sistema, y será necesario reinstalar algunos programas.

También puede suceder que le den errores compilando el núcleo `1.2.x` con un `gcc` *ELF* (`2.6.3` o superior). Se puede intentan arreglar añadiendo las siguientes líneas al fichero `arch/i386/Makefile`:

```
AS=/usr/i486-linuxaout/bin/as
LD=/usr/i486-linuxaout/bin/ld -m i386linux
CC=gcc -b i486-linuxaout -D__KERNEL__ -I$(TOPDIR)/include
```

Ahora, haga '`make dep`' y '`make zimage`' de nuevo.

En algunos casos muy raros el `gcc` romperá con un mensaje similar a "`xxx exited with signal 15`". En este caso la solución puede estar en desactivar la caché de segundo nivel, pensar en un fallo hardware de la memoria... o reinstalar de nuevo el `gcc`.

7.4 El nuevo núcleo no parece arrancar

No se ejecutó `LILO`, o no está bien configurado. A veces, se ponen errores como '`boot = /dev/hda1`' en lugar de '`boot = /dev/hda`'.

7.5 Se olvidó ejecutar `LILO`, y el sistema ya no arranca

¡Vaya problema! Lo mejor que puede hacerse ahora es arrancar con un disquete y preparar otro disquete para arrancar Linux (con `'make zdisk'` se hizo uno). Necesita saber qué sistema de ficheros raíz (`/`) tiene, dónde está y su tipo (por ejemplo, `ext2` o `minix`). En este ejemplo, también hay que saber dónde están los ficheros de `/usr`, en otra partición.

En el siguiente ejemplo, `/` es `/dev/hda1`, y el sistema con las fuentes del núcleo es `/dev/hda3`, montado como `/usr` normalmente. Ambos son sistemas `ext2`. La imagen compilada estará en el sistema de las fuentes.

La idea es que si hay un fichero `zImage` correcto, puede salvarse en un disquete. Otra posibilidad (que funcionará mejor o peor, según el caso) se verá en otro ejemplo.

En primer lugar, arranque con un disquete de instalación o de rescate, y monte el sistema de ficheros que contenga el núcleo a usar:

```
mkdir /mnt
mount -t ext2 /dev/hda3 /mnt
```

Si `mkdir` le dice que el directorio ya existe, no hay problema. Ahora, pase al directorio donde está el núcleo compilado. Vea que

```
/mnt + /usr/src/linux/arch/i386/boot - /usr = /mnt/src/linux/arch/i386/boot
```

Ponga un disco con formato en la unidad "A:" (que no sea el disquete con el que ha arrancado) y copie el núcleo a él:

```
cd /mnt/src/linux/arch/i386/boot
dd if=zImage of=/dev/fd0
rdev /dev/fd0 /dev/hda1
```

vaya a / y desmonte el sistema de ficheros:

```
cd /
umount /mnt
```

Ahora puede rearrancar el sistema desde el disquete. ¡No olvide ejecutar `LILO`!

Como se ha dicho, hay otra alternativa. Si el núcleo está en el directorio raíz (por ejemplo `/vmlinuz`), puede usarlo para un disquete de arranque. Suponiendo las condiciones anteriores, haríamos los cambios siguientes en el ejemplo anterior: `/dev/hda3` por `/dev/hda1` (el sistema raíz), `/mnt/src/linux` por `/mnt`, y '`if=zImage`' por '`if=vmlinuz`'. El resto puede ignorarse.

Usar `LILO` con discos grandes (de más de 1024 cilindros) puede dar problemas. Le recomendamos que lea el *mini-HOWTO* sobre LILO para más información.[15].

7.6 Mensaje de aviso: '`warning: bdflush not running`'

Es un problema muy importante. Desde la versión `1.0` del núcleo (20 de Abril de 1994), hay un programa, '`update`' que, periódicamente, vuelca al disco la caché de *buffers* del sistema. Obtenga las fuentes de '`bdflush`' (está donde se distribuyen las fuentes del núcleo) e instálelas (quizás deba usar mientras lo hace el núcleo antiguo). Después del rearranque, se instalará en memoria como '`update`' y ya no habrá más avisos.

7.7 Salen mensajes sobre símbolos no definidos, y no compila

Esto será probablemente porque tenga un compilador *ELF* (`gcc 2.6.3` y posteriores) y las fuentes `1.2.x` o anteriores. Habitualmente esto se corrige añadiendo las siguientes líneas al archivo `arch/i386/Makefile`:

```
AS=/usr/i486-linuxaout/bin/as
LD=/usr/i486-linuxaout/bin/ld -m i386linux
CC=gcc -b i486-linuxaout -D__KERNEL__ -I$(TOPDIR)/include
```

Esto permitirá compilar el núcleo `1.2.x` con librerías `a.out`.

7.8 No consigo que me detecte mi CD-ROM *IDE/ATAPI*

Mucha gente tiene este problema, siendo las causas muy diversas.

El error más común es tener el dispositivo en una interfaz IDE sin compañía de otro disco, para lo que debe ser configurado como "maestro" (master) o "único" (single), nunca como "esclavo" (slave).

Creative Labs está poniendo interfaces IDE en sus tarjetas de sonido. Sin embargo, esto es un problema cuando se tienen ya dos interfaces IDE en la placa, en la *IRQ 15*. Entonces se suele configurar el IDE de la tarjeta de sonido en la *IRQ 11*, y los núcleos `1.2.x` no saben manejar esto (tener, en la práctica, tres IDE).

[15] Así como el *Discos-Grandes-mini-Como*, disponible en castellano, ver sección 14.

En las versiones `1.3.x` se empieza a intentar soportar esto, pero está en desarrollo y en todo caso no incluye autodetección. Para utilizarlo, pues, deberá hacer algunas cosas.

Si no tiene segunda IDE en placa, cambie los "*jumpers*" de la tarjeta de sonido referentes a la interfaz IDE para que ocupen la *IRQ 15* (segunda interfaz). Esto debería funcionar.

Si por el contrario hay en total tres interfaces, obtenga un núcleo *1.3.x* (por ejemplo, el `1.3.57` lo incluye) y lea el documento `drivers/block/README.ide`, donde encontrará más información.

7.9 Salen mensajes sobre cosas de encaminamiento obsoletas

Consiga nuevas versiones de los programas '`route`' y otros que manipulan tablas de encaminamiento. El fichero `/usr/include/linux/route.h` tiene cambios al respecto.

7.10 La función de cortafuegos no funciona en el núcleo 1.2.0

Actualícese al menos a la `1.2.1`.

7.11 Mensaje: "Not a compressed kernel Image file"

No utilice el fichero `vmlinux` creado en `/usr/src/linux` para arrancar, sino el mencionado `zImage`.

7.12 Hay problemas con la consola al pasarse a la 1.3.x

Ponga '`linux`' en la entrada '`console`' del fichero `/etc/termcap`. Además, deberá hacer una entrada en `terminfo`.

7.13 Algunas cosas no compilan después de la actualización

El núcleo incluye ciertos ficheros `#include` (acaban en `.h`) que se referencian desde `/usr/include`. Típicamente se referencian con la línea:

```
#include <linux/xyzzy.h>
```

Normalmente, hay un enlace simbólico '`linux`' de
`/usr/include` al directorio `/usr/src/linux/include/linux`. Si no existe tal enlace, o apunta a un lugar incorrecto, muchas cosas compilarán mal. Por ejemplo, el problema es obvio si borró las fuentes del núcleo porque le ocupaban mucho. Otro problema puede tener relación con los permisos de los ficheros, tal vez debido a un `umask` de la cuenta `root` que obligue a crear los ficheros ocultos a otros usuarios por defecto. Puede solucionar esto con el comando `chmod`, pero será más cómodo descomprimir las fuentes del núcleo añadiendo la opción `-p` (preservar modo) al comando `tar`:

```
blah# tar zxvpf linux.x.y.z.tar.gz linux/include
```

Nota: Con "`make config`" se crea el enlace `/usr/src/linux` si no existe.

8 Notas sobre la actualización a la versión 2.0.x

La versión `2.0.x` se instala de manera algo diferente a otras versiones, y requiere actualizar software auxiliar. El fichero `Documentation/Changes` de las fuentes contiene información muy importante sobre esto, como lo referente a los paquetes que debe actualizar (`gcc`, `libc`...).

9 Módulos

Los módulos del núcleo cargables le permitirán simplificar la configuración del núcleo y ahorrar memoria. Sirven para añadir dinámicamente nuevos soportes de sistemas de ficheros o manejadores, a un núcleo que ya está corriendo.

9.1 Instalación de las utilidades asociadas

Estas utilidades están disponibles allí donde esté el núcleo, como `modules-x.y.z.tar.gz`. Elija el que corresponda a su núcleo. Descomprímalo con '`tar zxvf modules-x.y.z.tar.gz`', cambie al directorio que crea (`modules-x.y.z`), léase el fichero `README`, y siga las instrucciones de instalación (normalmente, tan simple como '`make install`'). Con ello tendrá las utilidades `insmod`, `rmmod`, `ksyms`, `lsmod`, `genksyms`, `modprobe` y `depmod` en `/sbin`. Si lo desea, puede probar el manejador ejemplo "`hw`" con `insmod`; lea el fichero `INSTALL` para más detalle.

`insmod` inserta un módulo en el núcleo arrancado. Generalmente los módulos son ficheros `.o`; el manejador ejemplo es `drv_hello.o`, con lo que para insertarlo se usaría '`insmod drv_hello.o`'. Para ver los módulos cargados, pruebe `lsmod`. La salida será como ésta:

```
blah# lsmod
Module:#pages:Used by:
drv_hello1
```

'`drv_hello`' es el nombre del módulo, que usa una página (4k) de memoria, y no hay módulos que dependan de él en este momento. Para quitar el módulo, teclee '`rmmod drv_hello`'. Vea que `rmmod` necesita el nombre del módulo, no del fichero. Ese nombre lo obtiene al listar los módulos instalados. Las otras utilidades de módulos se encuentran documentadas en los manuales *on-line*.

Inciso de la revisión 2.1

Tres cosas relativas a módulos:

- Siempre que compile un kernel nuevo, y sus correspondientes módulos (con `make modules; make modules_install` tras los `make dep; make clean; make zImage`[16] lo primero que ha de hacer al arrancar con el mismo, para computar las dependencias entre módulos, es invocar el comando

  ```
  depmod -a
  ```

- Desde la aparición de `kerneld`, el andar insertando y retirando módulos de memoria a mano no se estila mucho, en su lugar es `kerneld` quien se encarga de esto. Para hacer uso de él, asegúrese de responder sí al soporte `kerneld` durante la configuración del núcleo, y llámelo desde los *scripts* de inicialización del sistema, que están típicamente en `/etc/rc.d/rc.x`.

  Sigue pudiendo cargar módulos a voluntad; el comando recomendado es `modprobe modulo`, que tendrá en cuenta las dependencias entre módulos, cargando todos los necesarios (como por ejemplo el módulo `slhc` en el caso de que queramos cargar el módulo `ppp`.)

  Atención: El soporte de compresión CCP (`bsd_comp`) no está incluído en esto debido a problemas de *copyright*, por lo que deberemos cargarlo a mano; o bien ejecutamos la orden `modprobe bsd_comp` cuando queramos cargar `ppp`, o bien lo insertamos **después** de haber cargado `ppp` y `slhc`, con por ejemplo, `modprobe ppp`.

  Lea el fichero `/usr/src/linux/Documentation/modules.txt`, disponible en castellano en `http://www.insflug.org/pub/online/modulos.txt` para profundizar sobre este tema, así como sobre el tercero de los ítems:

- `/etc/modules.conf` o `/etc/conf.modules`. Lea el anterior fichero (`modulos.txt`).

[16] O `make zlilo`, si usa `LILO` como gestor de arranque.

9.2 Módulos distribuidos con el núcleo `1.2.2`

Algunos sistemas de ficheros, manejadores SCSI y de tarjetas de red, así como otras utilidades son cargables como módulos. Para usarlos, lo primero, no incluirlos en el núcleo principal (al compilarlo, no decir que 'sí' al manejador correspondiente durante el '`make config`'). Luego se arranca ese núcleo[17] y se entra de nuevo en `/usr/src/linux` para hacer '`make modules`', lo que compilará módulos para todas aquellas opciones que no se eligieron durante el '`make config`', y los dejará en `/usr/src/linux/modules`. Puede ahora ejecutar '`make modules_install`' que los instalará en `/lib/modules/x.y.z` siendo `x.y.z` la versión del núcleo.

Esto es especialmente útil con los sistemas de ficheros que, como `minix` o `msdos`, no se usan normalmente. Así, cuando se quiera leer un disquete MSDOS, primero se cargaría el módulo con `insmod`[18] y se descargaría al finalizar. Esto ahorrará unos 50K de memoria en la operación normal del núcleo. Notar que el sistema Minix siempre deberá estar dentro del núcleo (no como módulo) si se quiere hacer un disquete de rescate con él.

10 Otras opciones de configuración.

Esta sección contiene información sobre otras opciones de configuración que se ofrecen durante '`make config`'.

10.1 Opciones generales

`Normal floppy disk support` (soporte de floppy normal): es exactamente eso. Lea el fichero `drivers/block/README.fd`, es importante para usuarios de *IBM Thinkpad*.

`XT harddisk support` (soporte de disco duro de XT): interesante si quiere usar una controladora XT de 8 bits.

`PCI bios support`(soporte de Bios PCI): si tiene PCI, puede interesarle. Vaya con cuidado: algunas placas PCI antiguas pueden colgar al sistema con esta opción. Más información al respecto se encontrará en el documento *PCI-HOWTO*.

`Kernel support for ELF binaries` (soporte para binarios *ELF*): *ELF* es un intento de uniformizar arquitecturas y sistemas operativos. Linux también pretende conseguirlo[19].

`Set version information on all symbols for modules` (mantener información de versión en los símbolos de los módulos): en el pasado, los módulos se recompilaban con cada nuevo núcleo. Si responde afirmativamente, podrá usar módulos de otra versión. Para más detalle, consulte el fichero `README.modules`.

`Enable loadable module support` Responda que sí a esto si pretende hacer uso de los módulos cargables en tiempo de ejecución.

`Kernel daemon support (e.g. autoload for modules)`Esto es absolutamente necesario si pretende hacer uso de `kerneld`. En breve se dispondrá de una versión en castellano del *Kerneld Mini-HOWTO*. Ver sección 14.

[17]Esto es obsoleto en el caso de los núcleos `2.0.x`, recuérdelo; con los `2.0.x` puede compilarlo todo seguido, y arrancar posteriormente, ejecutar `depmod -a` y estará todo listo.

[18]Mejor `modprobe`, que como hemos comentado, tendrá en cuenta las dependencias entre módulos que ha generado `depmod -a`, cargándolos todos.

[19]De hecho, es el estándar actual para todo el software en Linux. Podrá seguir ejecutando sus (posiblemente obsoletos) antiguos binarios `a.out`, dotando al kernel, –modularmente, por ejemplo– de soporte para los mismos.

10.2 Opciones de red

Se describen en el documento *NET-2-HOWTO* (o *NET-algo-HOWTO*).

11 Consejos y trucos

11.1 Redirección de la salida de compilación o parcheado.

Si le interesa tener "*logs*" de lo que hagan los comandos '`make`' o '`patch`', puede hacerlo redirigiendo su salida a un fichero. Primero, debe saber qué shell está utilizando (con el comando '`grep root /etc/passwd`' puede saberlo, buscando en el resultado algo así como '`/bin/csh`' o '`/bin/sh`'). Una vez localizado, si utiliza `sh` o `bash`, puede redirigir la salida de un comando como '`make`' con:

```
(comando) 2>&1 | tee (fich)
```

La línea anterior pondrá la salida de '`(comando)`' en el fichero '`(fich)`'.

Si se trata de `csh` o `tcsh`, teclee,

```
(comando) |& tee (fich)
```

Para `rc` (shell que raramente usará) se teclearía,

```
(comando) >[2=1] | tee (fich)
```

11.2 Instalación condicional del núcleo

Hay más métodos para probar un nuevo núcleo que no sea usar un disquete. A diferencia de otros Unix, `LILO` puede arrancar cualquier fichero con el núcleo que se encuentre en el disco[20] (siempre y cuando éste esté por debajo de los 1024 cilindros, lea la documentación de `LILO` para más detalle). Puede tener al final de `/etc/lilo.conf` una línea parecida a ésta:

```
image = /usr/src/linux/arch/i386/zImage
label = nuevo_nucleo
```

lo que le permitirá seleccionar en el arranque (habiendo ejecutado antes lilo para reconfigurar) el *nuevo_nucleo* en lugar del habitual que será probablemente `/vmlinuz`. Para poder hacer esta selección hay que mantener pulsada la tecla SHIFT al arrancar, para que aparezca el *prompt* de `LILO` que le permita teclear la etiqueta (nuevo_nucleo u otra) del núcleo elegido.

Si desea tener varios fuentes de núcleos distintos en el disco (¡cuidado!, le puede suponer muchos *megas*), lo normal es tener cada versión `x.y.z` en el directorio `/usr/src/linux-x.y.z` y "seleccionar" o activar cada núcleo sin más que redefinir el enlace simbólico `/usr/src/linux` apuntando al directorio que desee (por ejemplo, yo hice '`ln -sf /usr/src/linux-1.2.3 /usr/src/linux`' en su momento). Asegúrese que `/usr/src/linux` no es ya un directorio, sino como mucho antiguos enlaces simbólicos. De otro modo, el resultado del anterior comando no será el esperado.

[20] Así como montar particiones de disco distintas según se le indique, con lo cual podemos tener varias instalaciones distintas y poder arrancar con una y otra, incluso compartiendo alguna de ellas.

11.3 Actualizaciones del núcleo

Russell Nelson, `nelson@crynwr.com` lleva una lista que resume los cambios en las nuevas versiones del núcleo. Son breves pero interesantes antes de decidirse por instalar una nueva versión. Se puede encontrar por FTP anónimo en `ftp://ftp.emlist.com/pub/kchanges` o a través del URL:

`http://www.crynwr.com/kchanges`

12 Otros documentos *COMO* que pueden serle útiles

- *Sonido-Como*[21]: tarjetas de sonido y utilidades
- *SCSI-HOWTO*: todo sobre controladores y dispositivos SCSI
- *NET-2-HOWTO*: todo sobre redes
- *PPP-Como*: redes con PPP concretamente
- *PCMCIA-HOWTO*: acerca de manejadores para portátiles
- *ELF-HOWTO*: *ELF*: qué es, cómo migrar...
- *Hardware-HOWTO*: revisión del hardware soportado por Linux

13 Miscelánea

13.1 El autor

El autor y encargado de mantener el original de este documento en Inglés es Brian Ward `bri@blah.math.tu-graz.ac.at`. Envíenme cualquier comentario, añadidos o correcciones que considere oportunos. En particular, lo que más me interesa son las correcciones. Si quiere agradecerme de alguna forma especial que haya escrito este documento, siempre podrá enviarme ordenadores o CD-ROMs que no use, o simplemente una bonita postal de su zona...

Puede ver mi página principal de WWW en los URLs:

`http://www.math.psu.edu/ward/`

`http://blah.tu-graz.ac.at/~ward/`

Aunque intento atender en lo posible el correo, recuerde que sin embargo recibo *muchos* mensajes diarios, lo que hará que pueda pasar tiempo antes de contestar su mensaje. Especialmente, si me envía alguna consulta, trate de ser lo más claro y detallado posible en su mensaje. Por ejemplo, si se trata de hardware que no funcione, trate de explicarme con el máximo detalle la configuración de que se trata. No se preocupe si hace preguntas simples, puesto que si no se la pregunta a nadie ¡nunca tendría respuesta!. Agradeceré también las sugerencias.

Si me escribe y no obtiene respuesta en mucho tiempo (digamos, más de tres semanas), puede que accidentalmente haya borrado su mensaje (lo siento, inténtelo de nuevo).

Recibo muchos mensajes sobre cosas que realmente son problemas de hardware. Está bien, pero tenga en cuenta que yo no conozco todo el hardware existente en el mundo y no siempre puedo ayudarle. Personalmente, uso máquinas con discos IDE y SCSI, CD-ROMs SCSI, tarjetas de red 3Com y WD, ratones serie, placas base con PCI, controladoras SCSI NCR 810, CPUs AMD 386DX40 con coprocesador Cyrix, procesadores AMD 5x86, 486DX4 y también de Intel (todo esto es una lista de lo que conozco, pero ciertamente no es una recomendación para su equipo, aunque si quiere, puede preguntarme `:-)`).

[21] Aquellos cuyo nombre incluya la palabra "*Como*" están disponibles traducidos.

La versión -0.1 se escribió el 3 de Octubre de 1994. Este documento está disponible en SGML, PostScript, TeX, roff y ASCII.

Nota del traductor: Al igual que el autor original, me pongo a su disposición para resolver problemas que estén en mi mano, y sobre todo, aceptar sugerencias sobre esta traducción, perteneciente al proyecto *LUCAS* (LinUx en CAStellano) coordinado por Ramón Gutiérrez. Actualmente, se han preparado versiones SGML, HTML y ASCII.

Nota de la revisión 2.1: Ver la sección 14. Me sumo a las propuestas anteriores. A partir de esta revisión, este *COMO* está disponible en SGML, HTML, ASCII, PostScript, DVI, LyX, y cualquier otro formato soportado por `linuxdoc-sgml`. Le recomiendo encarecidamente que si piensa imprimirlo, lo haga con las versiones `.dvi` o `.ps`, la calidad no tiene color en comparación con las otras versiones para imprimir. Y al fin y al cabo, me he tomado el trabajo de revisarlo precisamente por esto `:-)`, asi como actualizarlo un poco.

El traductor, Juan José Amor, `jjamor@ls.fi.upm.es` se encuentra disponible por correo electrónico y también en el WWW:

`http://lml.ls.fi.upm.es/~jjamor`.

13.2 Pendiente de hacer

La sección 11 es pequeña. Espero poderla agrandar gracias a sus sugerencias.

Lo mismo para la sección 6.

Hay que añadir más información acerca de recuperación de cuelgues y errores.

13.3 Colaboraciones

Se incluyó una pequeña parte del `README` de Linus (acerca de las opciones de "`kernel hacking`"). (¡Gracias, Linus!).

`uc@brian.lunetix.de` (Ulrich Callmeier): `patch -s` y `xargs`.

`quinlan@yggdrasil.com` (Daniel Quinlan): muchas correcciones y añadidos.

`nat@nataa.frmug.fr.net` (Nat Makarevitch): `mrproper`, `tar -p`.

`boldt@math.ucsb.edu` (Axel Boldt): recopiló descripciones de configuraciones del núcleo en la red, luego me envió la lista.

`lembark@wrkhors.psyber.com` (Steve Lembark): sobre múltiples formas de arrancar.

`kbriggs@earwax.pd.uwa.edu.au` (Keith Briggs): algunas correcciones y sugerencias.

`mailto:rmcguire@freenet.columbus.oh.us` (Ryan McGuire): adiciones realizables.

`dumas@excalibur.ibp.fr` (Eric Dumas): traducción al Francés.

`simazaki@yu-gate.yamanashi.ac.jp` (Yasutada Shimazaki): traducción al Japonés.

`jjamor@lml.ls.fi.upm.es` (Juan José Amor Iglesias): traducción al Castellano. `;-)`

`donahue@tiber.nist.gov` (Michael J Donahue): errores tipográficos, ganador de la "sliced bread[22] competition".

`rms@gnu.ai.mit.edu` (Richard Stallman): concepto sobre la documentación de libre distribución.

`dak@Pool.Informatik.RWTH-Aachen.DE` (David Kastrup): El tema del NFS.

[22]Proviene de la popular expresión anglosajona "*... it's the best thing since sliced bread*", algo así como (literalmente) "*... es lo mejor que se ha inventado desde las tostadas*", suele utilizarse para expresar "*... es lo mejor que haya parido mente humana*".

La gente que me ha enviado correo con preguntas y problemas también me ha sido de extrema ayuda.

13.4 Notas sobre el Copyright, Licencia y Todo Eso

Nota del traductor: Aunque se traducen las notas siguientes, se copian también intactas para respetar los términos de la Licencia.

14 Anexo: El INSFLUG

El *INSFLUG* forma parte del grupo internacional *Linux Documentation Project*, encargándose de las traducciones al castellano de los Howtos (Comos), así como la producción de documentos originales en aquellos casos en los que no existe análogo en inglés.

En el **INSFLUG** se orienta preferentemente a la traducción de documentos breves, como los *COMOs* y *PUFs* (**P**reguntas de **U**so **F**recuente, las *FAQs*. `:)`), etc.

Diríjase a la sede del INSFLUG para más información al respecto.

En la sede del INSFLUG encontrará siempre las **últimas** versiones de las traducciones: `www.insflug.org`. Asegúrese de comprobar cuál es la última versión disponible en el Insflug antes de bajar un documento de un servidor réplica.

Se proporciona también una lista de los servidores réplica (*mirror*) del Insflug más cercanos a Vd., e información relativa a otros recursos en castellano.

Francisco José Montilla, `pacopepe@insflug.org`.