

# Kernel Linux, Manual del usuario

Ulises Arias

23 de agosto de 2009

# Índice general

0.1. Introducción . . . . .	2
<b>1. Previos a la compilación</b>	<b>3</b>
1.1. Para que compilar el kernel . . . . .	4
1.2. Antes de empezar . . . . .	4
1.2.1. Software Necesario . . . . .	4
1.2.2. Decargar el kernel . . . . .	5
<b>2. Determinar hardware y configuración del kernel</b>	<b>6</b>
2.1. Descubramos nuestro hardware . . . . .	7
2.2. Configurar el kernel . . . . .	9
2.2.1. Device Drivers . . . . .	13
2.2.2. Wireless . . . . .	14
2.2.3. Filesystems . . . . .	15
2.2.4. Fin de la configuración . . . . .	15
<b>3. Compilación e instalación del kernel</b>	<b>16</b>
3.1. Kernel a la debian . . . . .	17
3.2. Modo tradicional . . . . .	17
<b>4. Finalización y desinstalación</b>	<b>18</b>

## 0.1. Introducción

Se ha dicho que GNU/Linux es uno de los mejores sistemas operativos existentes, que se puede hacer de todo sin necesidad de gastar ni un centavo en software, la mayor ventaja que tiene un sistema libre sobre un sistema privativo es **La capacidad de personalización** con la que podemos hacer que el software este configurado y hasta construido para nuestra máquina en particular.

# **Capítulo 1**

## **Previos a la compilación**

## 1.1. Para que compilar el kernel

Como se mencionó anteriormente, la gran ventaja de los sistemas operativos libres es la capacidad de adaptar el software a nuestras necesidades, hacer que pueda gastar menos recursos y además hacer que interactúe mejor con nuestro hardware.

Nosotros ya hemos hecho el primer paso importante usando un sistema operativo libre, y está funcionando en nuestra computadora, entonces ¿Para que necesitamos modificar el corazón del sistema si todo funciona perfectamente? Pues la respuesta es: “Para tener un sistema optimizado y construido para funcionar en nuestra computadora”, no solo para tener un sistema “personalizado” sino también para poder hacer funcionar hardware o alguna aplicación que de otra manera no podría ser habilitada. Además al actualizar el kernel podemos acceder a características que antes no estaban disponibles y claro aquí se mostrará como hacer eso.

## 1.2. Antes de empezar

Suponiendo que ya tengamos instalada la distribución de nuestra predilección podemos empezar pensando ¿Necesito actualizar o compilar mi kernel?. Pues si solo van a hacer una actualización por algun tema de seguridad o por que el nuevo soporta mejor una cosa, pueden probar instalando un paquete de kernel de la distribución en la que se encuentren con las herramientas apt en Debian, rpm-YUM en Fedora, Yast en OpenSUSE, urpmi en mandriva, etc.

Si quieren hacer un cambio mayor o luego de haber hecho lo anterior vieron que en el kernel de la distribución no esta activada la opción que estaban buscando, pueden descargar el código fuente del kernel para empezar la compilación (En algunas distribuciones como Gentoo o Slackware esta sera la unica forma).

### 1.2.1. Software Necesario

Para poder compilar cualquier software necesitamos un compilador de dicho lenguaje, en el mundo linux ya tenemos disponibles varias herramientas entre ellas el compilador gcc, este y muchas herramientas más estan en un paquete debian que se llama *build-essential* que es uno de los necesarios para la compilación:

Paquetes necesarios en Debian, Ubuntu y derivados
build-essential
kernel-package
libncurses5-dev (para make menuconfig)
libqt3-mt-dev (para make xconfig)
libgtk2.0-dev (para make gconfig)

Como pueden ver no son muchas las cosas que se necesitan para hacer un kernel en entornos Debian, el paquete build-essential viene como dependencia de kernel-package, kernel-package es un conjunto de herramientas que en el momento de compilar el kernel lo convierten en un paquete .deb para su fácil instalación y desinstalación por medio de dpkg, los paquetes -dev (dev de development o desarrollo) son bibliotecas compartidas necesarias para configurar lo que queremos incluir en el kernel, si lo haremos en

modo consola y no queremos usar interfaces gráficas para configurar, solo será necesario instalar el paquete libncurses5-dev y omitimos los otros paquetes -dev. Si en cambio queremos utilizar interfaz gráfica usaremos libqt3-mt-dev si es un entorno KDE, si es un entorno GNOME será mejor solo instalar el libgtk2.0-dev.

<b>Lista de paquetes necesarios para Opensuse, Fedora y derivados de RedHat</b>
---

ncurses-devel (para make menuconfig) make gcc yum-utils (Fedora) automake autoconf Yast - Base Development (OpenSuse) Yast - KDE Development (OpenSuse, para make xconfig) Yast - C/C++ Development (OpenSuse) Yast - Linux Kernel Development (OpenSuse) ncurses (Mandriva make menuconfig) (Todo lo que se menciona para una distribucion solo aplica a ella) (Todo lo que no dice para que distribucion es aplica a todas)
---

### 1.2.2. Decargar el kernel

Para compilar un kernel debemos conseguir el código fuente de este, no vamos a escribir nada de él, solo vamos a construir lo que otras personas ya programaron, si tienen todo el software que se mencionó anteriormente entonces ya tienen las herramientas, solo les falta la materia prima, **las fuentes del kernel**, las cuales podemos conseguirlas de 2 formas:

1. De su propia distribución con el gestor de paquetes
2. Descargando el último estable de <http://kernel.org> en la sección F al lado del número de versión 2.6.x.x

Estas vendrán en formato .tar.gz o .tar.bz2, si las descargaron de kernel.org copienlas a /usr/src/ y las descomprimen con:

```
cp linux-2.6.27.4.tar.bz2 /usr/src/  
tar -xvf linux-2.6.27.4.tar.bz2
```

Una vez descomprimado el paquete creamos un link hacia la carpeta de las fuentes del kernel con nombre "linux":

```
ln -s /usr/src/linux-2.6.27.4/ linux
```

## **Capítulo 2**

# **Determinar hardware y configuración del kernel**

## 2.1. Descubramos nuestro hardware

Lo que haremos a continuación será revisar nuestro equipo y determinar nuestro hardware para que el kernel quede configurado de la manera mas cercana posible a nuestro hardware, el mismo sistema nos dara la información que necesitamos para esta tarea.

Primero revisemos que hay en nuestras tarjetas PCI, PCI-express y motherboard:

lspci

Lo cual nos dara este resultado:

```
00:00.0 Host bridge: Intel Corporation Mobile PM965/GM965/GL960 Memory Controller Hub (rev 0c)
00:02.0 VGA compatible controller: Intel Corporation Mobile GM965/GL960 Integrated Graphics Controller (rev 0c)
00:02.1 Display controller: Intel Corporation Mobile GM965/GL960 Integrated Graphics Controller (rev 0c)
00:1a.0 USB Controller: Intel Corporation 82801H (ICH8 Family) USB UHCI Controller #4 (rev 02)
00:1a.1 USB Controller: Intel Corporation 82801H (ICH8 Family) USB UHCI Controller #5 (rev 02)
00:1a.7 USB Controller: Intel Corporation 82801H (ICH8 Family) USB2 EHCI Controller #2 (rev 02)
00:1b.0 Audio device: Intel Corporation 82801H (ICH8 Family) HD Audio Controller (rev 02)
00:1c.0 PCI bridge: Intel Corporation 82801H (ICH8 Family) PCI Express Port 1 (rev 02)
00:1c.1 PCI bridge: Intel Corporation 82801H (ICH8 Family) PCI Express Port 2 (rev 02)
00:1c.3 PCI bridge: Intel Corporation 82801H (ICH8 Family) PCI Express Port 4 (rev 02)
00:1c.5 PCI bridge: Intel Corporation 82801H (ICH8 Family) PCI Express Port 6 (rev 02)
00:1d.0 USB Controller: Intel Corporation 82801H (ICH8 Family) USB UHCI Controller #1 (rev 02)
00:1d.1 USB Controller: Intel Corporation 82801H (ICH8 Family) USB UHCI Controller #2 (rev 02)
00:1d.2 USB Controller: Intel Corporation 82801H (ICH8 Family) USB UHCI Controller #3 (rev 02)
00:1d.7 USB Controller: Intel Corporation 82801H (ICH8 Family) USB2 EHCI Controller #1 (rev 02)
00:1e.0 PCI bridge: Intel Corporation 82801 Mobile PCI Bridge (rev f2)
00:1f.0 ISA bridge: Intel Corporation 82801HEM (ICH8M) LPC Interface Controller (rev 02)
00:1f.1 IDE interface: Intel Corporation 82801HBM/HEM (ICH8M/ICH8M-E) IDE Controller (rev 02)
00:1f.2 SATA controller: Intel Corporation 82801HBM/HEM (ICH8M/ICH8M-E) SATA AHCI Controller (rev 02)
00:1f.3 SMBus: Intel Corporation 82801H (ICH8 Family) SMBus Controller (rev 02)
03:01.0 FireWire (IEEE 1394): Ricoh Co Ltd R5C832 IEEE 1394 Controller (rev 05)
03:01.1 SD Host controller: Ricoh Co Ltd R5C822 SD/SDIO/MMC/MS/MSPro Host Adapter (rev 22)
03:01.2 System peripheral: Ricoh Co Ltd R5C592 Memory Stick Bus Host Adapter (rev 12)
03:01.3 System peripheral: Ricoh Co Ltd xD-Picture Card Controller (rev 12)
09:00.0 Ethernet controller: Broadcom Corporation NetLink BCM5906M Fast Ethernet PCI Express (rev 02)
0c:00.0 Network controller: Broadcom Corporation BCM94311MCG wlan mini-PCI (rev 01)
```

Figura 2.1: resultado de lspci

Luego buscaremos lo que hay en usb con:

```
lsusb
```

que nos dara un resultado como este:

```
Bus 007 Device 004: ID 0204:6025 Chipsbank Microelectronics Co., Ltd CBM2080 Flash drive controller
Bus 007 Device 002: ID 05a9:2640 OmniVision Technologies, Inc.
Bus 007 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 006: ID 0a5c:4503 Broadcom Corp.
Bus 001 Device 005: ID 0a5c:4502 Broadcom Corp.
Bus 001 Device 004: ID 413c:8126 Dell Computer Corp. Wireless 355 Bluetooth
Bus 001 Device 003: ID 0a5c:4500 Broadcom Corp.
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

Ahora solo nos falta ver nuestro procesador:

```
cat /proc/cpuinfo
```

Y con eso obtenemos el siguiente resultado:

```
processor       : 0
vendor_id     : GenuineIntel
cpu family    : 6
model        : 15
model name    : Intel(R) Core(TM)2 Duo CPU   T5250  @ 1.50GHz
stepping     : 13
cpu MHz      : 1000.000
cache size   : 2048 KB
physical id  : 0
siblings    : 2
core id     : 0
cpu cores   : 2
apicid     : 0
initial apicid : 0
fpu        : yes
fpu_exception : yes
cpuid level: 10
wp         : yes
flags      : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat ps
e36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx lm constant_tsc arch_pe
rfmon pebs bts rep_good pni monitor ds_cpl est tm2 ssse3 cx16 xtpr lahf_lm
bogomips    : 2995.48
clflush size : 64
cache_alignment : 64
address sizes : 36 bits physical, 48 bits virtual
power management:
```

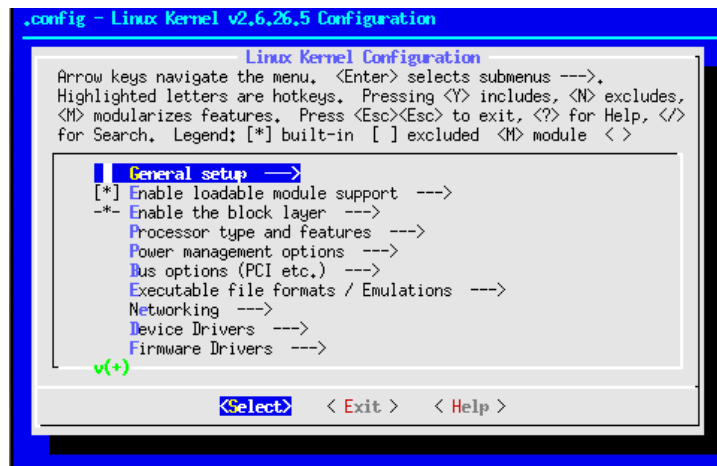
Con esto ya tenemos la información necesaria para empezar a configurar nuestro kernel.

## 2.2. Configurar el kernel

Ahora es tiempo de empezar con la verdadera optimización de nuestro kernel Linux, vamos como root al directorio donde está nuestro kernel que acabamos de descomprimir y ahora podemos elegir entre las siguientes.

1. **make config**: Con este nos preguntará una por una las cosas que queremos cambiar o no (no recomendado, es enorme y no puedes cambiar de opinión si te equivocas)
2. **make menuconfig**: Forma en modo consola pero por menús que permiten cambiar de opinión (recomendada, necesita ncurses5-dev en debian-like ncurses-devel en fedora-like)
3. **make xconfig**: Igual a menuconfig pero con entorno KDE (necesita libqt3-mt-dev debian-like, y qt3-devel en fedora-like)
4. **make gconfig**: Igual que las anteriores pero en GTK (necesita libgtk2.0-dev debian-like, y gtk2-devel en fedora-like)

Nosotros utilizaremos **make menuconfig** ya que nos permite hacer lo mismo que cualquier interfaz gráfica y es independiente del entorno de escritorio que estemos utilizando. Lo primero que veremos será esto:



Esta es la pantalla principal del menú de configuración como vemos trae las opciones principales de configuración:

- General setup – Esta opción configura la forma de manejar los parámetros globales del kernel
- Enable loadable module support – Esta opción habilita o deshabilita el soporte para los módulos cargables (Recomiendo no tocarlo)

- Enable the block layer – Soporte para discos o medios con mas de 2Tb de capacidad (es bueno deshabilitar todo)
- Processor type and features – Todo lo relacionado con el procesador esta aqui
- Power Management Options – Aqui se configura la administración de energia (boton encendido, ventilador, hibernación, escalado de cpu)
- Bus Option (Pci etc.) – Aqui habilitamos los tipos de bus de nuestra motherboard (no confundir con las coaster o bluebird :p)
- Executable file formats/emulations – Los formatos binarios que se ejecutaran en nuestra maquina (a.out, ELF,IA32 caso de amd64 )
- Networking – Todo lo relacionado a redes
- Device drivers – Los drivers estan aqui
- Firmware drivers – Este es el software que usan los drivers
- File systems – Los sistemas de ficheros (formatos de disco duro) que utilizará el sistema
- Kernel hacking – Parametros especiales para probar el kernel (solo desarrolladores)
- Security options – Seguridad incrustada en el kernel (con el default es suficiente)
- Cryptographic API – Soporte de encriptamiento
- Virtualization – Soporte para maquinas virtuales (virtualbox, qemu, vmware)
- Library routines – Configuración de bibliotecas compartidas (también recomendando no tocarla)

<sup>1</sup> Empecemos por *General setup*, ahí veremos una opción llamada *Prompt for development and/or incomplete code/drivers* la cual sirve para instalar en el kernel drivers y características que aun estan incompletas, en fase experimental o que no se han probado adecuadamente a menos que vayamos a desarrollar/programar drivers o código del kernel esta opción no la necesitamos así que la deshabilitamos.

Luego esta la opción *Local version - append to kernel release* en esta podemos escribir un nombre que se añadira al final del nombre del kernel, (como por ejemplo: -testing, -1, -prueba) es buena idea añadir un guión para distinguirlo de el nombre del kernel, para que esto tenga efecto marquemos la opción *Automatically append version information to the version string*.

Lo siguiente que buscaremos sera la opción *Optimize for size* si esta marcada esta hará que el kernel sea lo mas pequeño posible en disco, lo que hará que cargue más rápido en memoria, si no esta marcada hay que marcarla.

---

<sup>1</sup> cuando se marca una opción esta sera: "\*" incrustada en el kernel, "M" módulo o vacío

Ahora ya hemos terminado con la sección de *General Setup* pasaremos a *Enable the block layer* y como en una pc actual no tenemos discos con capacidades mayores a 2Tb entramos a deshabilitar todo lo que encontremos adentro, una vez terminado pasamos a *Processor type and features* donde configuraremos nuestro procesador.

Primero vemos las primeras 3 opciones *Tickless System (Dynamic Ticks)* maneja las interrupciones de sistema para el procesador por defecto aparece marcado, luego esta *High Resolution Timer Support* es el cronometro de alta resolución y luego la que nos interesa: *Symmetric multi-processing support* esta característica es la que nos permite usar mas de 1 cpu así que si nuestra maquina tiene un solo CPU entonces hay que deshabilitarlo, si tiene mas de 1 CPU o es de multiples núcleos entonces hay que habilitarla.

Desde la versión 2.6.27.4 hay una nueva opción *Enable MS table* que es un <sup>2</sup> bugfix para sistemas antiguos de varios CPU que no manejan bien el soporte ACPI (Gestión de energía) y para sistemas ETM64 (core 2 de intel por ejemplo) en los kernel AMD64.

Luego en *Subarchitecture Type* elegimos el tipo de subarquitectura del procesador, lo mejor es dejar el por defecto (*PC-compatible*) ya que los demas tipos son para servidores o routers.

<sup>3</sup> Luego bajamos hasta la opción *Processor family* donde pondremos específicamente el tipo de procesador que tenemos (por ejemplo: Pentium4, AMD Athlon, Celeron, etc.) Ahora las optimizaciones para velocidad, vamos a la opción *Preemption Model* donde nos pregunta que tan "pendiente" tiene que estar el kernel ante una orden, a lo que ponemos *Preemptible Kernel (Low-Latency Desktop)* con lo que el sistema respondera nuestra petición lo mas rápido posible, si queremos también podríamos marcar *Preemptible RCU* con lo que la latencia del kernel baja aun mas, también cambiaremos la opción *Timer frequency* donde pondremos 1000Hz como frecuencia del cronometro, con esto ya tenemos un kernel que trabajara en tiempo real.

<sup>4</sup> En la opción *High Memory Support* elegiremos una opción segun la cantidad de ram que tengamos instalada, las opciones son:

- off – Para máquinas con menos de 1G de ram
- 4G – Para máquinas con entre 1 y 4G de ram
- 64G – Para máquinas con mas de 4G de ram

Aqui hay muchas opciones que son específicamente para Intel o AMD así que si nuestro procesador es intel, deshabilitaremos todo lo que sea para AMD y viceversa.

Salimos de este submenu y vamos a *Power management options* donde podremos poner el escalado de CPU y las opciones de hibernación entre otras cosas.

<sup>5</sup> En la primera parte nos saldrán las opciones principales de energía dejemos todo como esta y vamos hacia el submenu *CPU Frequency scaling* ahí encontraremos la opción *CPU Frequency scaling* que si está habilitada podremos tener el sistema de

<sup>2</sup>Arreglo para un error de ediciones anteriores

<sup>3</sup>Esta información nos la da el comando `cat /proc/cpuinfo`

<sup>4</sup>Esta opción no aparece en AMD 64

<sup>5</sup>Si la maquina para la que se esta armando el kernel es muy vieja (aos 95-97) es mejor deshabilitar toda esta sección

escalado de frecuencia que nos permite usar solo la capacidad necesaria de energía en el CPU ahorrando energía (muy bueno en laptops) o de usar el poder solo cuando lo necesitemos.

<sup>6</sup> Después aparecen otras opciones para debuggin pero el que nos interesa es *Default CPUFreq governor* el cual sera el comportamiento por defecto del escalado de CPU.

- Performance – Mantiene el CPU a poder completo siempre
- Userspace – Da mas poder segun lo requiera el usuario de forma manual (se le dara la frecuencia minima)
- Ondemand – Este aumenta la frecuencia segun la carga del sistema (Recomendado)
- Conservative – Mantendrá el CPU en la frecuencia mínima a menos que el sistema este muy saturado

En este apartado podemos elegir cualquiera que prefiramos, según si queremos que la maquina tenga mas poder o gaste menos energía o poder manejarla manualmente mediante cpufreq, este sera el comportamiento por defecto pero podemos cambiar el governor con <sup>7</sup> cpufreq-set -g \$governor.

Ahora salimos del submenu y marquemos con <M> (modulo) todos los driver de los governor para tenerlos disponibles con cpufreq, luego estan los *\*\*\* CPUFreq processor drivers \*\*\** o los drivers para escalado de procesador, necesarios para usar los governor, marcar como modulo tambien los que necesitaremos, para los procesadores core, pentium d y celeron de nueva generación pondremos el *ACPI Processor P-States driver*, para los AMD de 64 bits usaremos *AMD Opteron/Athlon64 PowerNow!* para los de 32 bits seran *AMD k6/athlon PowerNOW!* para cualquier intel (genérico) podemos usar el *Intel Enhanced SpeedStep (deprecated)* el (deprecated) es por que pronto sera una tecnología obsoleta la cual sera sustituida por *Intel Pentium 4 clock modulation*.

Ahora vamos a configurar los buses, en el menu *Bus options (PCI etc.)* y dejamos todo habilitado, menos lo que tiene que ver con MCA que es un tipo de bus solo disponible en servidores, PCMCIA en caso de que tengamos una desktop no nos sirven estas tarjetas de laptop y tambien ISA que es un sistema muy antiguo de bus,

*Executable file formats / Emulations* lo dejamos como esta ya que por defecto se encontraran marcados todos los ejecutables que el kernel entendera como tales.

Vamos a la sección *Networking* y dedshabilitamos *Amateur Radio support* a menos que le conectemos a la pc un transmisor radioaficionado como el que usan en control de vuelo o para la policia, en caso de que no tengamos ningún dispositivo infrarojo, bluetooth o tarjetas wireless deshabilitemos las opciones referentes a estas.

Ahora entramos en la sección *Networking options* y veremos varias opciones sobre IP y protocolos de red algo que podemos hacer es deshabilitar el soporte a IPv6 que aun no esta vigente (aunque debería) y solo nos añadirá peso en el kernel ademas de

<sup>6</sup>Las opciones "debugging" del kernel son para desarrolladores o para que los errores den mas información

<sup>7</sup>\$governor sustituyase por alguno de la lista

esto buscamos la opción *Network packet filtering framework (Netfilter)* la cual añade características y soporte para hacer nuestra maquina un firewall de red, a menos que la maquina vaya a ser puesta como router o firewall de una red podemos deshabilitar esta opción, también podemos quitar el *Appletalk protocol support* si la maquina no es una mac(apple) y en la opción *Network testing* entramos y deshabilitamos todo, a menos que estemos programando drivers de red para el kernel.

### 2.2.1. Device Drivers

Aquí esta lo mas arduo de la configuración del kernel, ya que aquí se encuentran los drivers de periféricos, tarjetas, discos, etc. En general el hardware funcionara o no dependiendo de lo que hagamos aquí.

Vamos primero a *Parallel port support*, donde añadiremos el soporte para puerto paralelo, si tenemos una laptop nueva o de gama ultraportatil ya no tenemos este puerto (es conocido como el puerto de impresora) así que puedes deshabilitarlo, luego en *Block devices* deshabilitamos *Parallel port IDE device support* que es para conectar discos IDE por medio de puerto paralelo, si en la anterior no quitamos el puerto paralelo o tenemos uno de estos dispositivos es mejor dejarlo como esta, si la maquina no tiene diskettera puedes quitar *Normal floppy disk support*.

Vamos a la opción *IEEE 1394 (FireWire) support* y si tenemos este puerto lo habilitamos todo (siempre que no sea debug) y si no lo tenemos (es un puerto parecido al USB con forma de casa) deshabilitar todo, luego esta la opción *Macintosh device drivers* donde si no tenemos una mac la deshabilitaremos. En la seccion *Network device support* quitaremos lo siguiente:

- ARCnet support – razon: sistema raro de ethernet
- Token Ring driver support – razon: antiguo
- Wireless LAN – En caso de no tener wireless integrado o tarjetas wireless
- USB Network Adapters – En caso de que no tenga USB
- PCMCIA network device support – En caso de que no sea laptop
- Wan interfaces support – razon: Ninguna pc tiene interfaz de fibra optica
- FDDI – razon: La misma que la anterior
- Fibre Channel driver support – razon: ver 6 y 7

Luego al salir de *Network device support* vemos que hay varias opciones de telefonía: *ISDN support*, *Telephony support* estas son para personas que tienen integradas tarjetas de ISDN o telefonía y que ademas tienen el contrato con el ISP para pasar internet de esa manera (como un modem DSL integrado), deshabilitamos ambas.

Ahora entramos en *Input device support* donde pondremos todos los dispositivos de entrada de datos que usaremos en nuestra computadora. Primero deshabilitamos la opción *Support for memoryless force-feedback devices* que es para un tipo especifico de Joystick así que los usuarios normales que no jugamos (:p) deshabilitaremos esta

opción, nos vamos hasta la opción *Keyboard* donde configuraremos que teclado usaremos, usualmente sera un *XT keyboard* o un *AT keyboard* recomendando *XT keyboard* ya que es el mas comun *AT keyboard* es un tipo de teclado antiguo, deshabilitamos los demás.

Luego vamos a *Mice* donde configuramos el ratón habilitamos unicamente *PS/2 mouse* el cual activa los mouse PS/2 y USB, si tenemos un mouse de puerto serie habilitamos también *Serial mouse* y si tenemos una mac con touchpad USB o una laptop ponemos *Apple USB Touchpad support*, deshabilitamos los demás. Luego vemos, si tenemos joystick dejamos activada la opción *Joystick* si no la deshabilitamos, al igual que *Tablets* y *Touchscreens* si no tenemos una tabletpc o una pantalla táctil. Ahora nos salimos de esta sección y seguimos con *Graphics support* donde están los drivers (o el soporte) para tarjeta gráfica, primero, para tener tarjetas con aceleración (ya sea con driver de Xorg o con driver privativo) vamos a la sección */dev/agpgart (AGP Support)* y buscamos nuestro <sup>8</sup> chipset, una vez ubicado quitamos los demás, (por ejemplo si tienes una tarjeta intel no necesitas drivers de ATI, o si tienes ATI no necesitas los de Nvidia). Hacemos lo mismo en *Direct Rendering Manager (XFree86 4.1.0 and higher DRI support)* y en la sección *Support for frame buffer devices* debemos asegurarnos de que además de nuestros drivers este marcado *VGA 16-color graphics support* que es el soporte básico de video. Si queremos que un tux nos de la bienvenida al sistema podemos activar: *Bootup logo* (si tienes doble o cúadruple nucleo sera un tux por CPU).

Luego vamos a *Sound* donde estan: *Advanced Linux Sound Architecture* (aka. ALSA) y *Open Sound System* (aka. OSS), podemos usar el que queramos o compilar ambos, recomendando habilitar solo ALSA ya que OSS está deprecated y esta siendo reemplazado por ALSA, en caso de que después de compilar el kernel no tengamos sonido o tengamos problemas para usar este entonces podremos poner las dos alternativas. luego entramos en ALSA y deshabilitamos los drivers que no queremos en la sección *PCI devices* también podemos ver en las otras secciones sobre PCMCIA y USB.

Una vez terminado el sonido no tenemos necesidad de tocar nada mas en Drivers y podemos pasar a las siguientes secciones.

### 2.2.2. Wireless

Mientras estamos en la parte de networking vamos a ver algunas opciones referentes a wireless, las personas con tarjetas de este tipo deben dejar habilitado el soporte, y además asegurarse de que los protocolos y drivers de su tarjeta esten habilitados.

En la seccion *Wireless* del menu *Networking* verifiquemos que todo este marcado como modulo (podemos dejar deshabilitados los debug o debugging).

Luego en *Device Drivers* en la seccion *Network device support - Wireless LAN* dejemos habilitado el protocolo de nuestra tarjeta pre-802.11 o 802.11, normalmente sera esta última, y luego habilitamos solo el fabricante de nuestra tarjeta, (Intel, Atheros, Broadcomm, etc.).

---

<sup>8</sup>Esta información se encuentra en lspci

### 2.2.3. Filesystems

Aquí es hora de quitar un montón de cosas que nunca usaremos (a menos que tengamos el sistema operativo dueño de ese sistema de ficheros), empezamos marcando con <\*> las siguientes:

- Second extended fs support
- Ext3 journalling file system support
- Ext4 journalling file system support: Este es estable desde la versión 2.6.28
- Reiserfs support (si nuestra partición /boot o / usa este fs)

Y podemos quitar estos otros:

- Reiserfs – Si no lo tenemos en ninguna partición
- JFS
- XFS
- OCFS2

Luego dejamos todo lo demás como viene por defecto y vamos a la sección: *Miscellaneous filesystems* y deshabilitamos todo lo que no usamos (en algunos casos serán todos) con seguridad pueden deshabilitar todo aquí. Después vamos a *Partition Types* y deshabilitamos todo lo que no tenga que ver con particiones windows (o mac)

### 2.2.4. Fin de la configuración

Si no somos desarrolladores del kernel difícilmente necesitaremos la característica *Kernel hacking* así que podemos entrar a deshabilitar todo, y luego salimos del menú con el comando "Exit" de la parte de abajo, guardamos los cambios y hemos terminado con la configuración de nuestro kernel, al final no costo tanto ¿verdad?.

## **Capítulo 3**

# **Compilación e instalación del kernel**

Ya después de meditar en lo que pusimos y no pusimos en el kernel, si nos damos cuenta de que nos faltó algo o descubrimos que va algo que no debíamos poner esta es la hora de arreglarlo entrando de nuevo al `make menuconfig`, `xconfig`, `gconfig` o `config` y arreglar esto.

<sup>1</sup>Ahora ya seguros de lo que hicimos nos disponemos a compilar, yo por el momento pongo 2 formas de hacer esto:

1. Debian form: Convertir el kernel en un paquete `.deb` que podemos instalar y desinstalar con `dpkg`
2. Forma tradicional: Compilar el kernel de la forma normal, posible en todas las distribuciones.

Veremos ambas a continuación:

### 3.1. Kernel a la debian

Esta forma es la mas fácil para usuarios de Debian, Ubuntu y derivados, primero debimos haber instalado `kernel-package` en el capítulo **Previos a la compilación** la forma debian es un solo comando:

```
make-kpkg --initrd kernel_headers kernel_image
```

Que nos crea los paquetes imagen y headers del kernel, ahora solo salimos a `/usr/src` e instalamos asi: `dpkg -i *.deb`

Si queremos acelerar el proceso bastara con hacer varios trabajos simultneos, asi: antes de compilar agreguen la siguiente linea a `/etc/kernel-pkg.com`:

```
echo "CONCURRENCY_LEVEL := 3" >> /etc/kernel-pkg.conf
```

Con esto nos hara 3 trabajos de compilación simultáneos, la regla es numero de CPUS/cores +1 si tenemos una maquina con un cpu de un núcleo sera 2 en lugar de <sup>2</sup>3

### 3.2. Modo tradicional

Esta es la forma en la que el kernel se puede construir en cualquier distribución linux moderna, la forma tradicional viene a ser:

```
make && make modules_install && make install
```

Y ahora creamos el disco ram para este kernel (en este caso el 2.6.27.6)

```
mkinitramfs -o /boot/initrd-2.6.27.6.img 2.6.27.6 2.6.27.6
```

Ahora solo llamamos a nuestro gestor de arranque: `lilo` o `update-grub`.

Si quieren que su kernel se **compile más rápido** pueden hacer lo siguiente: Modifiquemos el comando de make y hagamoslo de esta forma:

```
make -j3 && make modules_install && make install
```

donde el `j3` es el número de trabajos simultáneos que hara el compilador, el 3 generalmente es numero de CPUS/cores +1, si tenemos un CPU de un nucleo sería `-j2`.

<sup>1</sup>Cualquiera de las dos formas tomara su tiempo en terminar

<sup>2</sup>el ejemplo es para doble núcleo

## Capítulo 4

# Finalización y desinstalación

Ahora solo nos queda reiniciar con el nuevo kernel instalado, es recomendado mantener siempre el kernel anterior para evitar problemas en caso que el kernel que nosotros hicimos no funcione.

En caso de haber problemas podemos desinstalar el kernel recién hecho así: Si lo hiciste a la manera de Debian primero inicia con el kernel que tenía la máquina y desinstala el que hiciste así:

```
apt-get purge linux-image-2.6.27.6
```

Si el kernel que hicimos es el 2.6.27.6

Si lo hiciste a la manera tradicional inicia con el kernel anterior y haces lo siguiente:

```
rm -rf /boot/*2.6.27.6*  
rm -rf /lib/modules/2.6.27.6/
```

Y luego corremos el cargador de arranque como lo hicimos después de compilar el kernel.

Con esto ya puedes crear tus propios kernels personalizados para cualquier distribución.