Creación de kernels personalizados con el sistema de empaquetado de kernels de Debian

Jesse Goerz

jwgoerz@users.sourceforge.net

Traducido al castellano por: Baltasar Pérez Díaz <`baltipd@yahoo.com`>

Este documento está dirigido a ayudar a los iniciados en Debian que quieran utilizar el sistema de empaquetamiento de kernels para crear kernels personalizados.

Gracias a Erdmut Pfeifer y a Oliver Elphick por las respuestas a mis preguntas sobre `make xconfig` en la lista de correo debian-user. No podría haber completado este documento sin su ayuda.

Gracias a Sebastian por el apunte que hizo de que casi cualquier cosa puede hacerse como un usuario normal cuando éste es miembro del grupo src. Gracias también a Robert Voigt por seguir las instrucciones de este documento y enviarme sus impresiones sobre los puntos que necesitaban mejorarse.

1. Introducción

Yo acostumbraba a compilar mis kernels de forma manual, lo que implicaba seguir una serie de pasos que se debían ejecutar en orden; el paquete kernel-package se escribió para hacer todos estos pasos de forma automática (además de otras cosas mas). Esto es especialmente importante para los iniciados: el comando make-kpkg realiza todos los pasos necesarios para compilar el kernel y la posterior instalación de este. --Manoj Srivastava, extraído del manual del kernel-package.

Si se ha compilado un kernel anteriormente, se sorprenderá de lo sencillo que resulta usando el paquete kernel-package. Este tutorial se creó con el fin de dar a conocer las ventajas del sistema de empaquetamiento de kernels que utiliza Debian. Lo que hace en resumidas cuentas es simplificar la compilación de un kernel y sus módulos creando un paquete con formato .deb de la imagen. Luego es solo cuestión de instalarlo con dpkg.

Compilar un kernel no es un asunto trivial. Lo que se va a hacer era anteriormente un trabajo reservado a los administradores y a los usuarios avanzados de Debian.

Si no se tiene cuidado se puede dejar el sistema incapaz de iniciarse. Se recomienda fuertemente que se lea todo el documento, desde principio a fin, por lo menos una vez, y luego intentar construir el kernel (e incluso entonces, a lo mejor se necesitaría mas información).

Como ejemplo en este documento se utilizará el código fuente del kernel 2.2.19. No se tiene que utilizar la misma versión; solamente se debería si es su primera vez. Si nunca antes había compilado un kernel, intente elegir una versión distinta a la que tiene funcionando en su sistema. Aunque en el documento se comentarán los pasos a seguir en el caso de que se esté compilando la misma versión del kernel, se ahorrarán unos cuantos pasos si se elige una versión distinta. En la siguiente sección se comentará la forma de averiguar la versión del kernel que se tiene instalada.

Por favor, recuerde que inevitablemente alguien usará este documento y dejará su sistema inoperativo. No permita que sea usted. Si no entiende alguna cosa intente buscar mas información antes de proceder. Si lee este documento y encuentra algun error, por favor hágalo saber en `<newbiedoc-discuss@lists.sourceforge.net>`.

2. ¿Qué instalar?

En primer lugar necesita instalar los siguientes paquetes para poder seguir este documento:

1. kernel-package
2. dpkg-dev
3. debianutils
4. binutils
5. libc6-dev
6. gcc
7. make

8. kernel-source-2.2.19
9. task-tcltk-dev (o bien, libncurses5-dev)
10. bin86 (solamente en PCs)

Instale estos paquetes simplemente escribiendo lo siguiente como usuario root:

```
bash# apt-get install kernel-package dpkg-dev debianutils binutils libc6-dev gcc make kernel-source-2.2.19 task-tcltk-dev bin86
```

> Observe que el paquete bin86 solamente es necesario en PCs (ni en Macs ni en Alphas). Además, puede descargar cualquier versión de las fuentes del kernel. Se ha escogido aquí la 2.2.19 porque es la última aparecida para Potato cuando se escribió este documento. También se recomienda elegir una versión del kernel que no sea igual a la que tiene instalada. Mas adelante se comentarán los pasos a seguir en el caso de que esté usando las mismas versiones.

> Nota sobre los paquetes con el código fuente del kernel: Los responsables en Debian de esos paquetes han hecho un excelente trabajo empaquetando el código fuente del kernel en ficheros .deb . Se recomienda que se elijan esos paquetes antes que otros por otros medios. La razón principal es que esos paquetes se descomprimen dentro de un directorio tal como `kernel-source-Version`. Por alguna razón el codigo fuente "oficial" de los kernels se descomprimen dentro de un directorio llamado `linux`. Éste puede causar problemas si no se tiene cuidado cuando se descompriman. La única razón que veo para no utilizar los paquetes con código fuente del kernel de Debian es si no tienen la versión que se necesita para un determinado hardware. Si ese es el caso, la siguiente sección explicará como evitar el solape de código fuente.

Para averiguar la versión del kernel que se tiene funcionando en el sistema, teclear lo siguiente:

```
bash$ uname -r
```

3. Comprobando que se tiene todo lo que se necesita

3.1. Añadiendo el usuario al grupo src

Ahora se debe echar un vistazo a las fuentes del kernel para asegurarse que se tienen instalados todos los programas que se necesitan. El código fuente del kernel se encuentra por defecto instalado en el directorioi `/usr/src` Normalmente se necesita ser el usuario root para realizar los cambios que se harán a partir de aquí. Puesto que siempre se debe evitar ser root, se hará un truco que permitirá a un usuario normal realizar los cambios que se necesiten. Abra un terminal y teclee la siguiente orden (en este momento si se debe ser root para ejecutar este comando):

```
bash# addgroup nombre_usuario src
Adding user nombre_usuario to group src...
Done.
```

Asegúrese de sustituir su nombre de usuario por `nombre_usuario`. Esta orden añade `nombre_usuario` al grupo src. Para que esta orden tenga efecto se debe salir y entrar de nuevo en la shell. Hay tres maneras de hacer esto:

1. Si está dentro de una ventana de terminal en las X, debe salir de las X, luego salir (logout) de la consola y finalmente volver a entrar.
2. Si está usando un gestor de login gráfico como xdm, gdm o kdm, simplemente debería salir de las X y volver a entrar.
3. Si está en la consola, simplemente salga y vuelva a entrar.

Para asegurarse de que la orden funcionó, ejecute el siguiente comando (después de haber hecho uno de los pasos anteriores):

```
bash$ groups
nombre_usuario src
```

Debería ver el grupo src como se muestra en el ejemplo. A continuación se creará el directorio de los ficheros fuente.

3.2. Creación del directorio de los ficheros fuente

El directorio del código fuente del núcleo contiene todos los ficheros necesarios para compilar el núcleo. Vaya al directorio `/usr/src` para ver si se tiene todo lo que se necesita. Escriba lo siguiente:

```
bash$ cd /usr/src
```

En ese directorio verá el fichero (comprimido con tar y bzip2) que instaló usando el comando apt-get (o comprimido con tar y gzip dependiendo de si usó apt-get o no):

```
bash:/usr/src$  ls
kernel-source-2.2.19.tar.bz2
```

Si no descargó el código fuente del kernel desde un sitio Debian necesita comprobar que *no* tiene un enlace simbólico llamado `linux` en su directorio `/usr/src`. Si lo tiene, debería eliminarlo antes de descomprimir el código fuente. Si no lo elimina sobreescribirá el anterior código

> fuente con el nuevo, y a lo mejor el nuevo kernel no se compilará correctamente. Vaya a la sección Creación del enlace simbólico. Puede ignorar este aviso si descargó el código usando el comando apt-get que se comentó en la sección "¿Qué instalar?".

Para descomprimir el fichero .tar.bz2 haga como usuario normal:

```
bash:/usr/src$  tar -xIf kernel-source-2.2.19.tar.bz2
```

Para descomprimir el fichero .tar.gz haga como usuario normal:

```
bash:/usr/src$  tar -xzf kernel-source-2.2.19.tar.gz
```

Al descomprimirse creará un directorio con el nombre `kernel-source-2.2.19`. Dentro de ese directorio se encuentra otro llamado `Documentation`. Vaya a ese directorio y busque un fichero llamado `Changes`. Edite ese fichero con su editor de textos favorito y busque una sección llamada Current Minimal Requirements. Este es el aspecto que tiene el mio:

Ejemplo 1. Contenido parcial de /usr/src/kernel-source-2.2.19/Documentation/Changes

```
Current Minimal Requirements
****************************

   Upgrade to at *least* these software revisions before thinking you've
encountered a bug!  If you're unsure what version you're currently
running, the suggested command should tell you.

- Kernel modutils         2.1.121                 ; insmod -V
- Gnu C                   2.7.2.3                 ; gcc --version
- Binutils                2.8.1.0.23              ; ld -v
- Linux libc5 C Library   5.4.46                  ; ls -l /lib/libc*
- Linux libc6 C Library   2.0.7pre6               ; ls -l /lib/libc*
- Dynamic Linker (ld.so)  1.9.9                   ; ldd --version or ldd -v
- Linux C++ Library       2.7.2.8                 ; ls -l /usr/lib/libg++.so.*
- Procps                  2.0.3                   ; ps --version
- Procinfo                16                      ; procinfo -v
- Psmisc                  17                      ; pstree -V
- Net-tools               1.52                    ; hostname -V
- Loadlin                 1.6a
- Sh-utils                1.16                    ; basename --v
- Autofs                  3.1.3                   ; automount --version
- NFS                     2.2beta40               ; showmount --version
- Bash                    1.14.7                  ; bash -version
- Ncpfs                   2.2.0                   ; ncpmount -v
- Pcmcia-cs               3.0.14                  ; cardmgr -V
- PPP                     2.3.10                  ; pppd --version
- Util-linux              2.9z                    ; chsh -v
- isdn4k-utils            v3.1beta7               ; isdnctrl 2>&1|grep version
```

En el listado se puede ver las versiones mínimas de los programas que tiene instalados para que el nuevo kernel funcione sin problemas. En la primera columna aparece el nombre, en la segunda la versión mínima que deben poseer, y en la última columna la forma de comprobar su versión. Tenga en cuenta que no necesita todos los paquetes listados si no se están utilizando en su sistema (p.e. no necesitará el paquete pcmcia-cs si no tiene ninguna tarjeta pcmcia).

Tenga que en cuenta que el que escribió ese archivo fue un desarrollador/a del kernel basándose en la distribución de GNU/Linux que tenía, por lo que algunos de los comandos para comprobar la versión puede que no funcione en su Debian. Si le ocurre esto, intente averiguar la versión del comando siguiendo el siguiente ejemplo.

```
bash:/usr/src$ man nombre_programa
```

Sustituyendo `nombre_programa` por el nombre del programa que desea.

3.3. Creación del enlace simbólico

Una vez que se ha comprobado que se tiene instalado todo lo que se necesita, ahora se debe pasar a crear un enlace simbólico en el directorio del código fuente del kernel.

```
bash:/usr/src$  cd /usr/src
```

Antes de establecer el enlace simbólico, compruebe que no tiene ya uno en ese directorio llamado `linux`. En el **Ejemplo 2** se le indica como hacerlo. Si ya tiene uno puede ser debido a que ya se ha instalado o compilado un kernel antes. Asegúrese de borrar el enlace antiguo y crear uno nuevo que apunte al directorio del código fuente que se quiere compilar.

Para crear el enlace simbólico debe colocarse en el directorio `/usr/src` y teclear lo siguiente:

```
bash:/usr/src$  ln -s kernel-source-2.2.19 linux
```

El formato es **ln** -s directorio_con_el_código_del_kernel linux.

Debe sustituir directorio_con_el_código_del_kernel por el directorio que se creó al descomprimir.

Para cerciorarse de que se hizo el enlace correctamente haga un **ls -l**. Debería aparecer lo siguiente.

Ejemplo 2. Enlace simbólico

```
bash:/usr/src# ls -l
total 66908
-rw-r--r--    1 root     root     15433132 Apr  7 21:38 kernel-source-2.2.19.tar.bz2
drwxr-sr-x   15 jesse  ❶src          4096 May 23 16:30 kernel-source-2.2.19
lrwxrwxrwx    1 jesse    src            20 May 23 19:16 linux -> kernel-source-2.2.19
                                                                ❷
```

❶ Observar que el grupo al que pertenecen el enlace y el directorio con el código fuente es en ambos src. Esto permitirá compilar el código fuente sin necesidad de ser el usuario root. Su nombre de usuario debería estar incluido en el grupo src como se comentó en la sección "Añadiendo el usuario al grupo src".

❷ Observe la pequeña flecha que indica a donde apunta el enlace.

Ahora que se ha creado el enlace simbólico se podrá pasar a configurar el kernel.

4. Configuración del kernel

4.1. make xconfig

Existen varios programas para configurar un kernel. El primero que se verá es el xconfig. Se puede acceder a él con el comando **make xconfig**. Debe asegurarse que está en el directorio `/usr/src/linux` antes de invocarlo.

```
bash:/usr/src$ cd linux
```

```
Y luego:
```

```
bash:/usr/src/linux$ make xconfig
```

Si le surge algún problema consulte la sección "Problemas comunes". Si todo va bien saldrá una ventana como la siguiente:

El cómo configurar el kernel se escapa al objetivo de este documento. Para encontrar información de como hacerlo puede leer el Kernel-COMO en El Proyecto de Documentación en Linux (http://www.linuxdoc.org/HOWTO/Kernel-HOWTO.html), la documentación que se encuentra en `/usr/src/linux/Documentation`, y la documentación de su hardware. En el futuro se piensa añadir a esta sección un apartado de configuración de las opciones mas comunes del kernel.

Una vez terminada la configuración del kernel presione el botón "Save and Exit" y pase a la siguiente sección, "Creación de la imagen del kernel."

4.2. make menuconfig

Necesita libncurses5-dev. Utilizado normalmente cuando no se tiene las X.

4.3. make config

Casi nunca se utiliza. Use "make oldconfig" en su lugar.

5. Creación de la imagen del kernel

Esta es la parte que mas tiempo tarda. Si posee un ordenador antiguo con un procesador lento puede llevarle incluso horas, por ello se provee la facilidad de crear el kernel en un ordenador rápido y luego copiarlo al ordenador lento. Esta parte no se cubrirá en este documento. Yo construí el kernel 2.2.19 en un pentium 120 con 128 MB de RAM y tardó unos 20-25 minutos.

Para crear el kernel se utilizará el programa `make-kpkg`. Antes de empezar se debe primero ver que son las revisiones y los flavours. Para obtener una información mas amplia se puede consultar la documentación que se instala con este paquete.

make-kpkg es parte del paquete "kernel-package". Los documentos mas importantes se pueden localizar en `/usr/share/doc/kernel-package/`. Una buena idea es consultar su manual con **man make-kpkg**.

5.1. Revisiones

La revisión es una opción que se le pasa a **make-kpkg**, con la que se añade información extra al nombre del kernel que se va a crear. Esto tiene su importancia porque ayudará a mantener kernels distintos y a saber distinguirlos, sobre todo si se utiliza dselect o apt-get. Una cosa importante a recordar es que solamente se pueden utilizar caracteres alfanuméricos y el signo "+" para definir la revisión. Los guiones bajos "_" *no* se pueden utilizar. Además se puede preceder al nombre de la revisión con un dígito seguido de dos puntos ":".

A lo largo de este tutorial se utilizará como ejemplo la siguiente revisión `--revision=custom.1.0 kernel_image`. También se podría preceder al nombre con dos puntos ":".

Si desea utilizar la forma `--revision=1:custom.1.0 kernel_image` por favor lea antes `/usr/share/doc/kernel-package/README`. Esta acción provoca una marca de tiempo. Ahora mismo la discusión de ello se escapa al ámbito de este documento.

Por favor lea `/usr/share/doc/kernel-package/README` si no entiende algo de esto o quiere agregar otras opciones.

5.2. Flavours

Los flavours tienen en común con las revisiones que añaden información extra al nombre del paquete. Tienen el mismo convenio para nombrarlos que las revisiones (caracteres alfanuméricos, "-" "+" "."). *No* use mayúsculas o guiones bajos. La razón principal para usar la opción de flavour es cuando va a compilar un kernel con el mismo número de versión que el que está actualmente corriendo.

PRIMERO se necesitará hacer un "make-kpkg clean" si desea recompilar el kernel usando la opción de flavour. Al utilizar flavour pueden aparecer conflictos de módulos que le creará problemas

Si siguió el consejo que se le dió antes y consiguió una versión diferente del kernel que la que esta corriendo, no tiene que preocuparse de la opción de flavour. Si no tiene elección consulte la documentación sobre los flavours. Se espera que en futuras versiones de este documento trate mejor este tema. Si posee conocimientos sobre esto y le gustaría contribuir por favor contacte conmigo en `<newbiedoc-discuss@lists.sourceforge.net>`.

Lea `/usr/share/doc/kernel-package/Flavours` si esta compilando un kernel con la misma versión que la que esta corriendo. Puede encontrar la versión del kernel que tiene corriendo mirando al final de esta página.

Puede conseguir un kernel sin utilizar la opción flavour si está dispuesto a deshacerse de sus módulos antiguos y sobreescribir su actual kernel. No se recomienda esto porque si comete un error su única solución es un disquete de arranque (si es que tiene uno), y reparar algunas cosas utilizando unas herramientas con las que probablemente no esté familiarizado. Si es aventurero y quiere intentarlo pues adelante, pero tenga en cuenta que debe hacer backup de sus datos mas importantes y tener a mano un disquete de arranque.

5.3. Creando la imagen del kernel

Lo primero antes de compilar el kernel es hacer un disquete de arranque. De esta manera si algo va mal todavía se tiene la oportunidad de arrancar el ordenador y repararlo, sin tener que reinstalar el sistema. Para crear un disquete de arranque simplemente busque un disquete que no piensa borrar y escriba esto como usuario root:

```
bash# mkboot ruta_al_kernel
```

Cambie `ruta_al_kernel` con la ruta absoluta hacia el kernel que quiere utilizar, por ejemplo `/boot/vmlinuz-2.2.17`. El directorio `/boot` contiene su kernel actual (a no ser que lo haya colocado en otro sitio). `vmlinuz` es el nombre de su kernel y la versión se le añade a este. Si ha compilado un kernel anteriormente puede tener el nombre de `bzImage` o `zImage`.

Por fin ha llegado el momento. Ya está todo listo para crear la imagen del kernel y sus módulos. Ejecute los siguientes comandos como usuario normal:

```
bash$ cd /usr/src/linux
```

```
bash:/usr/src/linux$ make-kpkg --revision=custom.1.0 kernel_image
```

No cambie `kernel_image` con la versión de su kernel. El comando se debe escribir exactamente como se indica, a no ser que decida ponerle otro número de revisión.

Ahora todo lo que tiene que hacer es esperar a que su ordenador termine de compilar.

Si ha utilizado este documento antes para crear un kernel empaquetado deberá cerciorarse de que incrementa el número de revisión. Por ejemplo, podría cambiar el número de revisión a 1.1 o 2.0. Se necesita hacer esto porque dpkg no actualizará su versión previa del kernel al observar que tienen el mismo número de revisión y pensará que es la misma. También deberá renombrar el directorio que contiene los módulos del actual kernel, por ejemplo a 2.2.19-old. Los módulos están en `/lib/modules/`.

6. ¿Y ahora qué?

6.1. Instalación del kernel creado

Una vez que ha terminado de compilar el kernel necesitará instalarlo junto con sus módulos. Lo primero es comprobar si el kernel se compiló sin errores cambiando al directorio `/usr/src` haciendo lo siguiente:

```
bash:/usr/src/linux# cd ..
bash:/usr/src# ls
custom-kernel.config                     kernel-source-2.2.19    kernel-source-2.2.19.tar.bz2
❶kernel-image-2.2.19_custom.1.0_i386.deb  linux
```

❶ Ahí está. Como se puede comprobar ha añadido la opción de revisión que indicó junto con la arquitectura del sistema.

Ahora para instalar el kernel y todos sus módulos escriba lo siguiente:

```
bash:/usr/src# dpkg -i kernel-image-2.2.19_custom.1.0_i386.deb
```

Como parte del proceso de post-instalación se le hace una pregunta de si quiere instalar los módulos también. Responda que si.

Si ha usado la opción flavour o está instalando una versión del kernel que es igual a la que tiene corriendo actualmente en su sistema, abra INMEDIATAMENTE un terminal y cambie el nombre de su directorio de módulos actual. Si no hace esto, se provocarán conflictos entre módulos dejando al sistema inusable. Lea `/usr/share/doc/kernel-package/Flavour` para obtener mas información.

El comando para mover los módulos sería el siguiente (como usuario root):

```
bash:/lib/modules# mv 2.2.19 2.2.19-old
```

Sustituyendo 2.2.19 por su número de versión.

El proceso de post-instalación debería preguntar si quiere hacer un disquete de arranque. Si no lo hizo antes, hágalo ahora.

También pregunta si quiere correr lilo. La documentación dice que esto es un fallo pero no se han observado problemas. Conteste afirmativamente.

Si no ha hecho un disquete de arranque *hágalo*, ya que si por cualquier razón falla algo no será capaz de reiniciar el sistema. Si utiliza loadlin o cualquier otro gestor de arranque, debería consultar la documentación en `/usr/share/doc/kernel-package` antes de contestar a la anterior pregunta. Lo mas probable es que conteste que no.

Si todo ha ido bien solamente necesitará reiniciar el ordenador y comprobar si funciona. Si falla al reiniciar haga uso del disquete de arranque que hizo. En futuras versiones de este documento se quiere cubrir mas fallos que puedan surgir.

6.2. Terminando

Enhorabuena, ha compilado su primer kernel con el paquete kernel-package. Hay un par de cosas mas que se debe hacer antes de utilizar el nuevo kernel.

En primer lugar, puesto que no se le colocó una marca de tiempo antes al kernel, se necesita poner a nuestro kernel recién instalado en estado "hold" para evitar que las herramientas dselect o apt-get intenten actualizarlo. Esto se hace como se muestra a continuación (como usuario root):

```
bash# echo "kernel-image-2.2.19 hold" | dpkg --set-selections
```

Se ha de sustituir el nombre del kernel que aparece arriba por el de su kernel. El carácter "|" se consigue con **AltGr-1**. Para comprobar que el kernel está en estado "hold" escriba como root:

```
bash# dpkg --get-selections | grep kernel-image
```

Si el paquete está en "hold" ha de aparecer lo siguiente:

```
bash# dpkg --get-selections | grep kernel-image
kernel-image-2.2.19     hold
```

A continuación se necesita eliminar el enlace simbólico que se creó en el directorio `/usr/src`. No es absolutamente necesario pero es una buena idea. Existen varias razones porque la mayoría de la gente borra este enlace:

1. La próxima vez que se quiera descargar el código fuente de un nuevo kernel y no sea de un servidor Debian, al descomprimirlo sobreescribirá su antiguo directorio de código fuente del kernel por el nuevo debido a el enlace.
2. La próxima vez que se descargue el código fuente del kernel desde un servidor de Debian no pasará nada al descomprimirlo, pero si se olvida de borrar el enlace y crear uno nuevo al nuevo código estará sin querer recompilando el kernel antiguo.

3. Cuando se descarguen parches o código fuente del kernel y se descompriman en su directorio específico es mejor que no exista otro código que interfiera.

No es necesario eliminar el enlace, pero se recomienda. Es fácil volverlo a crear y no afecta a nada el que se borre. Para eliminarlo se ha de ejecutar lo siguiente como usuario root:

```
bash# cd /usr/src
bash:/usr/src# rm linux
```

Por último se recomienda que se guarde el paquete que contiene al nuevo kernel en otro lugar. Se puede crear otro disquete de arranque que contenga al nuevo kernel. No es una buena idea utilizar el disquete con el kernel antiguo ya que este no se ha comprobado mucho. Si se instaló el kernel-2.2.19 escriba lo siguiente como root (si es otro nombre cambielo):

```
bash# mkboot /boot/vmlinuz-2.2.19
```

Un apunte final, si necesita recompilar su kernel porque compró un nuevo hardware o faltó algo por habilitar en las opciones del kernel, lo único que se ha de hacer es un "make-kpkg clean" e incrementar en 1 la opción de revisión. A continuación se resumen los pasos:

```
bash$ cd /usr/src/linux

luego:

bash:/usr/src/linux$ make-kpkg clean

Reconfigurar el kernel:

bash:/usr/src/linux$ make xconfig

Y luego:

bash:/usr/src/linux$ make-kpkg --revision=custom.2.0 kernel_image
```

7. Resumen de los pasos a seguir

1. Instalar paquetes necesarios y sus dependencias.
   a. kernel-package, dpkg-dev, debianutils, libc6-dev, gcc, make, kernel-source-x.x.xx, binutils, task-tcltk-dev o libncurses5-dev, bin86 (solamente en PCs)

2. Creación del directorio del código fuente.
   a. Si no se ha hecho ya, añadir el usuario al grupo src: `addgroup user group`. Salga del terminal/shell (logout) y vuelva a entrar (login).

   b. Utilizar comando `groups` para confirmar.
   c. Comprobar que el enlace simbólico se ha hecho correctamente.

3. Configurar el kernel.
   a. Cargar fichero de opciones (.config) anterior si esta disponible.
   b. make xconfig, menuconfig, oldconfig, config.
   c. Guardar fichero de configuración en algún otro lugar para utilizarlo como base en nuevas compilaciones.

4. Crear la imagen del kernel.
   a. Si esta es una recompilación del mismo kernel compruebe el número de revisión y haga `make-kpkg clean`.
   b. Crear un disquete de arranque.
   c. `make-kpkg --revision=custom.X.X kernel_image`

5. Instalar el paquete del kernel creado.
   a. Si esta es una recompilación del kernel hacer `mv /lib/modules/X.X.XX /lib/modules/X.X.XX-old` si es necesario.
   b. Poner el kernel instalado en estado "hold". `echo "kernel-image-2.2.19 hold" | dpkg --set-selections`
   c. Eliminar el enlace simbólico.
   d. Guardar la imagen del kernel y hacer otro disquete de arranque.

8. Problemas comunes

Si se obtiene el error siguiente al hacer un "make xconfig"

```
bash:/usr/src/linux# make xconfig
rm -f include/asm
( cd include ; ln -sf asm-i386 asm)
make -C scripts kconfig.tk
make[1]: Entering directory '/usr/src/kernel-source-2.2.19/scripts'
cat header.tk >> ./kconfig.tk
./tkparse < ../arch/i386/config.in >> kconfig.tk
echo "set defaults \"arch/i386/defconfig\"" >> kconfig.tk
echo "set ARCH \"i386\"" >> kconfig.tk
cat tail.tk >> kconfig.tk
chmod 755 kconfig.tk
make[1]: Leaving directory '/usr/src/kernel-source-2.2.19/scripts'
wish -f scripts/kconfig.tk
Xlib: connection to ":0.0" refused by server
Xlib: Client is not authorized to connect to Server
Application initialization failed: couldn't connect to display ":0"
Error in startup script: invalid command name "button"
    while executing
"button .ref"
    (file "scripts/kconfig.tk" line 51)
make: *** [xconfig] Error 1
```

No se alerte, es una medida de seguridad de Debian. Para arreglarlo hacer lo siguiente:

```
bash:/usr/src/linux# xauth merge ~user/.Xauthority

Y luego hacer otra vez:

bash:/usr/src/linux# make xconfig
```

Cambie *user* por el nombre de usuario con el que ha entrado en el sistema. *No* ponga root. Ponga su nombre de usuario. Usted podría estar actualmente como usuario root (en el terminal) pero ha entrado (o debería) en X-Windows como un usuario normal.

El hacer esto tiene algún riesgo de seguridad. Consulte los archivos (http://lists.debian.org/) de la lista de correo debian-user si siente curiosidad.