Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores | Bergantells Usuaris de GNU/Linux de Mallorca i Afegitons

Debian: kernel-image desde adentro. (18948 lectures)

Per **juanca**, ***juanca*** (http://)

Creado el 01/08/2003 23:50 modificado el 01/08/2003 23:59

> *Como usario Debian desde un tiempo he trabajado mucho con el montando servidores y dejando de lado el sombrero rojo. Durante todo este tiempo he recompilado kernels cada dia asi que me pase tiempo buscando informacion hacerca de como hacerlo y varios conceptos relacionados al mismo al estilo Debian. Por sobre me di cuenta que necesitaba saber muchas cosas, como por que utilizar ese estilo y que funcion desempeñaba cada aplicacion al ser corrida, Me he topado con dinosaurios para que operen como servidores(66 MHZ) placas de red tambien del periodo cretaceo, asi que se imaginaran mi vida, realmente miserable. No les parece?*

Bueno ya saben un poco sobre compilacion al estilo Debian publicado aqui con todos los colores, yo les hablare un poco sobre todo lo que conlleva realizarlo paso por paso, concepto por concepto de todo lo que iremos corriendo. Todo el material lo he recompilado para ustedes y para mi por supuesto usando muchos howtos.

Sale el Tutorial

Preparativos
1. Prerando un boot diskette
Antes del compilado lo primero que se deberia hacer un preparar uno, pero desde mi punto de vista no seria tan necesario, el por què?. Bueno yo en particular cuando tengo un mensaje de kernel Panic, no entro en panico, solucion: si lo boteo de nuevo y mientras lo hace presiono varias veces SHIFT y el lilo me muestra dentro pantalla el kernel nuevo y el viejo, asi que lo booteo con el kernel viejo y comienzo de nuevo, pero primero desistalo el kernel fallido. De todos modos muestro como hacer uno.

juanca:~# mkboot path_del_kernel
Reemplacen el path_del_kernel por la informacion apropiada, ejemplo, /boot/vmlinuz-2.4.21
2 Què instalar?

Yo en particular trabajo con Debian Woody 3.0r1, pero tengo algunos sid`s conmigo tambien.
1- gcc
2. kernel-package
3. kernel-source-2.4.21
4. libc6-dev
5. tk8.0 or tk8.1 or tk8.3
6. libncurses5-dev
7. fakeroot
8. bin86 (para la construccion de los 2.2.x kernels en PCs)

Puedes instalar estos paquetes haciendo

juanca:~# apt-get install gcc kernel-package kernel-source-2.4.18 libc6-dev tk8.3 libncurses5-dev fakeroot

Instalando estos paquetes cuasara muchos otros paquetes que necesitan ser instalados para satisfacer las dependencias. Si estas compilando un 2.2 serie del kernel para arquitecturas de PC (maquinas con procesadores AMD son PCs, pero no las MAC o Alphas) para ellos necesitaras el paquete bin86
3. Estableciendo el arbol fuente
"Source tree" o arbol fuente es solo un termino para el directorio que contiene el codigos fuentes que seran compilado.

3.1. Agregandote al grupo de src.
Cuando el paquete de kernel-source esta instalado, el source tarball es hubicado en el diretorio /usr/src/ . Para trabajar en este directorio debes estar como root o pertenecer al grupo src. Para evitar hacer cosas potencialmente peligrosas.
Agregate al grupo de src.
juanca:~# adduser username src
tambien de esta forma si ya tienes el user.
juanca:~# usermod -g src username
Debes de desloguearte para pasarte a este usuario. haciendo:
1- Si estas en KDE presiona CRTL+Alt+Pagedown, o si estas en una consola como root has "su - username" que se encuentra dentro del grupo src.
2- O simplemente desde una consola "logout"
Para asegurarte de que el usuario username se encuentra dentro del grupo src ejecuta esto.
username:~$ groups
username src
Deberas ver lo que esta arriba.
3 Expandiendo el kernel-source
Desbzipear el kernel compactado.
bash:~$ cd /usr/src
bash:/usr/src$ ls
kernel-source-2.4.21.tar.bz2
Si ves un link simbolico llamado linux remuevelo.
bash:/usr/src$ rm linux.
Los fuentes del kernel-source de Debian se expanderan en un directorio llamado kernel-souce-2.4.21/ ""
Para expandir el bzip2 has.
bash:/usr/src$ tar jxf kernel-source-2.4.21.tar.bz2
bash:/usr/src$ cd /usr/src/kernel-source-2.4.21
3.2. Estableciendo un link simbolico
bash:/usr/src$ ln -s kernel-source-2.4.21 linux
Usa este comando que usa un amigo Javier Rugel siempre.
username:/usr/src/$ ls -las
total 66908
lrwxrwxrwx 1 root src 30 Jul 30 16:36 linux -> kernel-source-2.4.21

3.3. Checkeo de Requerimientos Minimos.
Todos los ejemplos los encontraras en /usr/src/kernel-source-2.4.21/Documentation/Changes

o Gnu C 2.95.3 # gcc --version
o Gnu make 3.77 # make --version
o binutils 2.9.1.0.25 # ld -v
o util-linux 2.10o # fdformat --version
o modutils 2.4.2 # insmod -V
o e2fsprogs 1.25 # tune2fs
o jfsutils 1.0.12 # fsck.jfs -V
o reiserfsprogs 3.6.3 # reiserfsck -V 2>&1|grep reiserfsprogs
o pcmcia-cs 3.1.21 # cardmgr -V
o PPP 2.4.0 # pppd --version
o isdn4k-utils 3.1pre1 # isdnctrl 2>&1|grep version

En vez de checkear por cada programa requerido individualmente, puedes correr el script ver_linux, que te buscara en el directorio de scripts del arbol del kernel. Pero primero necesitas hacerlo ejecutable.

Ejemplo 3. Usando ver_linux para el checkeo de requeriminetos minimos

bash:/usr/src/linux/Documentation$ cd ../scripts
bash:/usr/src/linux/scripts$ chmod +x ver_linux
bash:/usr/src/linux/scripts$ ./ver_linux
If some fields are empty or look unusual you may have an old version.
Compare to the current minimal requirements in Documentation/Changes.

Linux juanca 2.4.21 #1 Sat Jul 7 22:34:55 EST 2003 i686 unknown

Gnu C 2.95.4
Gnu make 3.79.1
util-linux 2.11z
mount 2.11z
modutils 2.4.21
e2fsprogs 1.27
pcmcia-cs 3.2.1
PPP 2.4.1
Linux C Library 2.2.5
Dynamic linker (ldd) 2.2.5
Procps 2.0.7
Net-tools 1.60
Console-tools 0.2.3
Sh-utils 2.0.11
Modules Loaded ide-cd cdrom nls_cp437 vfat fat input ipt_TOS ipt_state ipt_REJECT ipt_LOG ipt_limit iptable_mangle iptable_filter parport_pc lp parport apm sb sb_lib uart401 ipt_conntrack ip_conntrack_irc ip_nat_irc ip_nat_ftp ipip ip_gre v_midi awe_wave sound soundcore isa-pnp ipt_MIRROR ipt_REDIRECT ipt_MASQUERADE iptable_nat ip_tables ip_conntrack_ftp ip_conntrack 8139too mii af_packet rtc ext2 ext3 jbd ide-disk ide-probe-mod ide-mod unix

Si no puedes determinar la version del programa en tu Debian usando el metodo de arriba, intenta usando dpkg.

Ejemplo 4. Usando dpkg para checkear los Requerimientos minimos

bash:/usr/src/linux/scripts$ dpkg -l make
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name Version Description
+++-==============-==============-==
ii make 3.79.1-14 The GNU version of the "make" utility.

Si estas usando los fuentes de kernel.org o de Debian testing o de la rama unstable, puedes necesitar hacer un update de algunas de tus herramientas para conocer los requeriminientos minimos.
4. Configurando el kernel
4.1. Notas Generales
Un importante principio de la configuracion del kernel es TANSTAAFL (There Ain`t No Such Thing As A Free Lunch). algo asi como "No existe nada como un almuerzo gratis"
Algunas caracteristicas que le agregues al kernel aumentara su tamaño(a mi me paso que tuve un disco de 700 MGb y tuve que sacarlo de todo para que me quepe el kernel con todo lo que queria) y el tiempo para construirlo , incluso si escojes agregarle como modulos
Para hacer el kernel utilizable para mas personas en el hardaware, Las distribuciones Linux generalmente incluyen soporte para muchas arquitecturas de hardware y sus funciones, Debian no es un excepcion; el kernel pre-compilado incluye supporte de todo tipo de hardware que nunca has imginado, e idiomas que jamas has leido. Si testeas tu nuevo kernel y encuentras que algo de tu hardware no funciona, es facil de cambiar la configuracion y construccion de otro kernel.
El tamaño del kernel depende de la configuracion, El kernel 2.4.21 tiene alrrededor de 700 KB; el kernel que esta construido usa alrrededor de 1350 KB o mas.
Aparte del gran tamaño del mismo, otra caracteristrica del reciente kernel image is su uso de initrd (que viene incorporado desde el kernel 2.4.18) como algo mas de que preocuparce?

Puedes evitar la necesidad de preocuparte de initrd asegurandote de que compile directamente dentro del soporte del kernel (no como modulos) para el booteo del hardware y le root filesystem

Si el dispositivo duro que boteas desde el IDE, compila el soporte IDE dentro del kernel.y so tu root filesystem is Reiserfs asegurate de que el soporte Reiserfs no este construido como modulo directamente dentro del kernel.

Asegurate de construir un soporte de floppy disk dentro del kernel. Si lo compilas como modulos no podras bootear con tu nuevo kernel desde un floppy disk
Para optener mas informacion sobre la configuracion del kernel, vea The Linux Documentation Project`s "kernel-Howto" www.tldp.org y los archivos en /usr/src/linux/Documentation, y la documentacion de tu hardware.

Busca los archivos de debian-user-mailing list y has preguntas ahi y si no encuentras respuestas ahi lo tienes al amigo www.google.com/linux
4.2. make xconfig

Existen muchas formas de configurar el kernel. La primera de es "xconfig" que tira las optiones en tu "x"
bash:/usr/src/linux$ make xconfig
Nota que el IEEE 1394 y el sopote Bluetooth estan con un gris claro sin proderce seleccionar. No se encuentra disponible por que ellos dependen de otras opciones las cuales no han sido habilitadas. Has click en "Code maturity level options", Ahora clicka en el boton del menu "Main menu" y podras notar que ambos botones pueden ser selecionados como los otros. Donde veas opciones que no se pueden seleccionar, es que dependen de otras opciones que han sido deshabilitadas en alguna parte.
4.3. make menuconfig
Tal vez no estas usando el X, y eres alguien que cuando se le pasa el X se pierde has esto.

bash:/usr/src/linux$ make menuconfig
Las opciones son las mismas que xconfig, pero ya con algo de colores dentro de tu consola.
Para los que se inician con Linux o compilando el kernel deben de probar primero con xconfig y luego cuando estan familiarizados con la configuracion del kernel probar con menuconfig.
4.4. make config
Que seria consola pura y la mas larrrrrrrrrrrrrrrrrrrrga que puedes encontrar alrrededor de 1300 direntes preguntas un por una, asi que es mucho mejor hacer un xcofig o menuconfig
5. Construccion de kernel o Building kernel
Si tienes un maquiena vieja (como yo las tengo) con procesadores lentos puede tomar varias horas y cuando digo horas me refiero a 4 o 5 para la construccion de kernel-image de Debian
Es imposible construir el kernel en una maquina rapida e instalarlo en una maquina lenta, se los digo, no lo hagan.

5.1. make-kpkg
Para construir el kernel invocaras "make-kpkg", un script el cual automaticamente reemplaza la secuencia de "make dep; make clean; make bzImage;make modules". Tomate un tiempo y lee sobre el manual de make-kpkg esta muy interesane.
make-kpkg forma parte del kernel-package
La documentacion la podras encontrar en /usr/share/doc/kernel-package/.
El comandon make-kpkg puede ser complejo y a primera vista intimidatorio.
La sintaxis basica es la siguiente.
make-kpkg <options> <target>
Tu target seria tu kernel-image
Examinemos dos de las optiones mas importantes y comunes."--append-to-version" and "--revision".

5.2. --append-to-version
La primera opcion te permite especificar una al version del kernel, que luego forma parte del nombre del kernel

Deberas usar carateres alphabeticos , "+" y "." (period or full stop), no utilices esta raya "_".
Aqui esta el kernel, y lo estoy corriedo ahora.

bash:/usr/src/linux$ uname -a |cut -f3
bash:/usr/src/linxu$ Linux sirius 2.4.18.030709 #1 Sun Jul 9 22:15:39 EST 2003 i586 unknown unknown GNU/Linux.
Si utilizas "--append-to-version=.030709" cuando construyas tu imagen para este kernel
Es una taquigrafia de 2003 Julio 09 y es una forma de distinguir la fecha en que compilaste el kernel.
Deberas evitar usar --appen-to-version con las optiones de "-686", -k7 and -sparc, son muy utilizados para los Debian precompilados.
Los modulos del kernel se encuentran en subdirctorios de /lib/modules: cada kernel tiene su propio subdirectorio de /lib/modules para mantener sus modules.
Esto significa que cada que le des un valor de --apped-to-version tendra un nuevo nombre, y no tendra conflictos con otros kernels.

Si instalas un kernel con el mismo nombre (la misma version --append-to-version) de un kernel que ya esta instalado, el kernel nuevo hara un overwrite del viejo kernel , y los modulos .

Te lo advertira y podras abortarlo.
Asi toma esa opcion, y usa otro valor --append-to-version y reconstruyelo.
5.3. --revision
Otro opcion de make-dpkg is "--revision", el cual ofrece el nombre del paquete de Debian, pero no el nombre del kernel. Con --append-to-version, puedes usar solo caracteres alfanumericos "+" y ".", no utilices las rayas o guion bajo, si no soporta un valor para --revision, make-kpkg , usara "10.00.Custom".
Usando diferentes valores de --revision no prevendra conflictos entre los kernel con el mismo nombre.
No usaras --revision en este tutorial.
5.4. Los nombers de los paquetes del Kernel.
Los archivos de los nombres del kernel de Debian tienen esta forma.
kernel-image-(kernel-version)(--append-to-version)_(--revision)_(arquitectura).deb
El nombre del paquete es un todo antes del primer guion bajo o raya. Aqui hay una lista parcial de este kernel.

bash:/usr/src$ ls
kernel-image-2.4.21.030720_1.0_i386.deb
kernel-image-2.4.21.030720_10.00.Custom_i386.deb
Ahora puedes ver porque las rayas no son permitidas en las optciones de make-kpkg --ellos separan los elementos de los nombres de los paquetes
Existen dos kernels con los nombres de arriba, y puedes instalar ambos sin ninguna precupacion
Por favor dale una mirada a /usr/share/doc/kernel-package/README.gz, si --append-to-version o --revision si algo no esta claro.
5.5. fakeroot
Cuando te agregas al grupo de src sera posible que muchos funciones del kernel-package puedan ser ejecutadas con privilegios normales. Sin embargo los archivos que son parte de kernel-package (como el kernel y los modulos del kernel), seran adueñados por el root y corridos con privilegios de root.
Usando fakeroot puedes iniciar make-kpkg como un usuario normal, y muchas de las operaciones seran corridas con permisos normales. Cerca del final de la operacion, fakeroot simulara un entorno de root para crear el paquete de kernel-image
El la pagina del manual para make-kpkg describe un forma de usar fakeroot, nosotros utilizaremos el mas simple metodo de poner fakeroot al principio del comando make-kpkg, como esto.
fakeroot make-kpkg <options> <target>
5.6. Construyendo la imagen del kernel
Finalmente la hora ha llegado --estas listo para hacer un make a la imagen del kernel
bash:/usr/src/kernel-source-2.4.21$ fakeroot make-kpkg clean
Luego has

bash:/usr/src/kenel-source-2.4.21$ fakeroot make-kpkg --append-to-version=.030720 kernel_image
Ahora que todos pueden hacer esto esperen que la computadora finalice, Toma unos minutos para finalizar.
6. Instalando el paquete kernel-image
Una vez que la imagen del kernel esta consrtruida deberras instalar el kernel-image, que incluye el kernel y sus modulos. Primero revisa para asegurarte que se construyo exitosamente cambiando todo el directorio /usr/src y listando el contenido.

bash:/usr/src/kernel-source-2.4.21$ cd ..
bash:/usr/src$ ls
kernel-source-2.4.21
kernel-source-2.4.21.tar.bz2
kernel-image-2.4.21.030720_10.00.Custom_i386.deb
linux
Se encuentra ahi? , como puedes notar la --append-to-version llego a ser parte del nombre del kernel.
Para instalar el kernel y todos los modulos cambiate a root y has:
bash:/usr/src# dpkg -i kernel-image-2.4.21.030720_10.00.Custom_i386.deb

Para instalar el paquete del kernel image desde el disco usa el nombre completo
El script de post-instalcion deberia pedir si quiere hacer un boot disk. Si hiciste uno antes puedes decir que no aqui.
Si no has hecho un boot disk todavia, ahora necesitas hacerlo. Si el proceso de post-instalacion de no se lo dices a lilo,

el simplemente no podra reiniciar sin un boot disk.
Asegurate de decir "no" cuando el script pregunte si deseas instalar un block de booteo usando un lilo existente /etc/lilo.cong. Eso estaria intentando bootear con un nuevo kernel con la vieja configuracion, y no funcionara.
Luego te preguntaras si quieres sacar tu vieja configuracion de lilo y hacer una nueva, si estas usando lilo tendras que decir "si" luego te preguntara sobre la configuracion de booteo desde el disco duro.
Si estas usando un bootloader distinto que lilo deberas mirar esta documentacion en /usr/share/doc/kernel-package antes de responder tus preguntas. Probablemente no quieras el post-install script para mantener el bootloader.
Si respondes que "si", estaras incitado para instalar una particion de boot record, instala en el master boot record y has tu particion de root y activala.
Si usas grub pegate una miradita por kernel-package y grub
Si todo tuvo exito, todo lo que necesitaras es un hacer reboot a tu maquina. Si tu kernel no bootea, inserta un boot disk y bootealo de vuelta, o si no quieres hacer eso cuando bootea comienza a presionar SHIFT y podras seleccinar el kernel viejo. luego regresa a la configuracion anterior y intentalo de nuevo.
Felicitaciones en la compilacion de tu primer kernel con kernel-package al estilo Debian, pero hay todavia cosas por hacer antes de cantar victoria.
7. Ahora què?
7.1. Mantener el!
Si usaste --append-to-versin no necesitaras preocuparte sobre apt-get tratando de hacer un upgrade a tu kernel.
Si eres paranoico has esto:
bash:~# echo "kernel-image-2.4.21.030720 hold" | dpkg --set-selections
Sustiyendo el nombre por supuesto por el nombre del kernel que usas.
Para referirnos a un paquete que dpkg conoce (un paquete instalado o uno en una lista de archivos) necesitas usar el nombre del paquete en vez del nombre completo del archivo.
Mira que el paquete realmente esta en hold; si es cierto deberas ver esto.
juanca:~# dpkg --get-selections | grep kernel-image
kernel-image-2.4.21.030720 hold
7.2. Removiendo el symlink
Luego, deberas liberarte del sysmlink que has creado en el directorio /usr/src. Existen muchas rezones para esto
1. La siguiente vez que bajes el kernel no podria ser un archivo de Debian. Cuando expandas el source tarball puede ser reescrito tu viejo source tree atravez del viejo sysmlink.
2. La siguiete vez que bajes los fuentes de Debian , deberias poder expandir el source dentro su propio arbol sin problemas. Desde que estas listo ya tienes un symlink llamado "linux" en el directorio src, podrias seguir adelante y compilarlo.

3. Cuando descargas los patches u otro source code dentro de un source tree especifico.
Removiendo el symlink podrias prevenirlo.
1 Desde que ocurre.
para remover el link has esto:
bash:~# cd /usr/src
bash:/usr/src# rm linux
7.3. Haciendo un Back up de tu kernel
Mientras no lo requiera, es una buena idea hacerlo para personalizar tu kernel.deb, copialo a un lugar distinto.
Una vez que tu kernel ha sido empaquetado con estos modulos, puedes instalarlo en cualquier maquina que tiene el hardware que has especificado en la configuracion de tu kernel.

Nota: Una vez use un kernel-image de una Pentium III solo para probar en un Pentium IV y funciono de pelos, pero no les recomiendo que hagan lo mismo, solo lo hice para probar.

Incluso podrias incluso reinstarlo en tu actual maquina despues de reinstalarlo en tu sistema.
7.4. Creando un nuevo disco de booteo
bash:/usr/src# cd /boot
bash:/boot# mkboot /boot/vmlinuz-2.4.21.030720

7.5. Removiedo los kernels viejos existentes
Ahora que puedes ver que es facil de construir e instalar kernel, no estas lejos antes de que tu directorio de /boot esta lleno de kernels. configs y System.maps. Usa dpkg para purgar lo que no necesitas.
bash:/boot# dpkg -P kernel-image-2.4.21.030709
Si optienes este mensajej que "/lib/modules/2.4.21.030709 no esta vacio por lo tanto no removido, probablemente necesitaras instalar la tercera parte de la particion ahi, ellos tambien podran ser removidos, purgando los paquetes (Mira

la tercera parte de los modulos para averiguar como hacerlo con make-kpkg)

bash:/boot# dpkg -P pcmcia-modules-2.4.21.030709
Si los subdirectorios de modulos todavia no estan vacios, probablemente tienes los modulos del kernel que no han sido construidos con make-kpkg. Puedes remover los subdirectorios de modulos, una vez que no necestias nada mas.
7.6. Construyendo tu siguiente kernel
Si deseas reconstruir tu kernel, por que te compraste una nueva placa de sonido, o quieres habilitar nuevas caracteristics que olividaste la primera vez, todo lo que tienes que hacer es reconfigurarlo haciendo "fakeroot make-kpkg clean" y reconstrui tu kernel con un valor diferente a la version --append-to-version. Tu sesion debera lucir de esta manera:

bash:~$ cd /usr/src
bash:/usr/src$ ln -s kernel-source-2.4.21 linux
bash:/usr/src$ cd linux
bash:/usr/src/linux$ make xconfig

(reconfigurar tu kernel)
bash:/usr/src/linux$ fakeroot make-kpkg clean
bash:/usr/src/linux$ fakeroot make-kpkg --append-to-version=.030401 kernel_image

8. Topicos Avanzados
Parecera un poco extraño tener una pagina de Topicos Avanzados en un documento para newbies.
Se sabe que algunos lectores no son nuevos en linux o incluso en la construccion de kernels; Ello s estan tratando de aprender el modo Debian para administrar sus sistemas

Una vez que has exitosamente usado el kernel-package para construir e instalar tu propio kernel, querras explorar algunas formas para hacer tu vida mejor.
8.1. Usando una configuracion existente
Cuando instalas un kernel personalizado, su configuracion fue copiada a /boot/config-x.y.z.

Descarcarga y descompacta el nuevo source tarball y has un nuevo symlink, luego cambia los directorios a /usr/src/linux, copia la configuracion existente ahi y has "make oldconfig"
bash:/usr/src/linux$ cp /boot/config-2.4.21.030720 .config
bash:/usr/src/linux$ make oldconfig
Te preguntaras sobre las nuevas opciones. Es generalmente seguro responder "no" a todas las preguntas, has notas de las nuevas opciones que te interresen.
Despue de terminar la vieja configuracion puedes correr xconfig o menuconfig para reveer tus selecciones, cambiar tu respuestas, o investigar sobre las opcione que anotaste
Luego has "fakeroot make-kpkg clean" y ya estas listo para la construccion de la ultima version estable del kernel con tu propia configuracion.
8.2. Kernel-package y el grub
Todavia Lilo sigue siendo el cargador por default de Debian, funciona bien con make-kpkg.
Despues de instalar grub, necesitas correr "grub-install /dev/hda" (sustituyendo tu dispositivo de booteo "/dev/hda").
Luego corre "update-grub". Edita /boot/grub/menu.lst y sustitui tu default en las lineas "#groot" y "#kopt" (y alguna otra linea que necesitas para tu situacion)
Corre "update-grub" de nuevo.
Crea el archivo /etc/kernel-img.conf e inserta las siguientes lineas:
postinst_hook=/sbin/update-grub
postrm_hook=/sbin/update-grub
Ahora cuando sea que instales el nuevo kernel-image, el update-grub scanneara tu directorio de /boot, el inventario del kernel ahi, y escribira un nuevo boot menu. Si usas --append-to-version=.yymmdd, tu kernel sera clasificado por la version del kernel y la fecha.
Por default el Grub es comenzar con el kernel en la lista, a menos que escojas uno diferente antes de que el menu haga un time out. Cuando remueves el kernel, update-grup removera su entrada del menu de boot.
Cuando estes comodo con grub, has "dpkg -P lilo" y no te volvera a preguntar sobre lilo cuando instales el nuevo kernel.
8.3. Modulos fuera del kernel sourcer tree.
Un ejemplo es "pcmcia-cs", para muchos usuarios de laptops, lo necesitan, es decir el utimo soporte PCMCIA.
Para usar pcmcia-cs, apt-get el paquete "pcmcia-source"; el source tarball deberia estar en /usr/src

Com un usuario normal, descomprime el source tarball a /usr/src/modules/pcmcia-cs. Si todavia no tienes el directorio de modulos, sera creado.
Cuando configuras tu kernel, con el soporte PCMCIA, (Si tienes soporte PCMCIA habilitado, pcmacia-cs no construira los drivers.) Si deseas usar wireless LAN, habilitalo bajo
"Network device support" , pero no lo selecciones ningun otro dispositivo.
Cuando estes listo para la construccion del kernel, agrega "modules_image" en la linea de comando:
bash:/usr/src/linux$ fakeroot make-kpkg --append-to-version=.030701 kernel_image modules_image
Despues de que tu kernel-image esta hecho, make-kpkg ira a /usr/src/modules y construira los modulos .deb por cada subdirectorio, con la misma version y --append-to-version como tu kernel.
Instala los modules-image.debs despues de instalar el kernel-image.deb

8.4. Agregando los modulos a un kernel ya existente.
Como has notado muchos estan interesados en ALSA (Advanced Linux Sound Architecture)
Habras notado que tu nueva tarjeta de sonido no suena tan bien como deberia, y deseas probrar alsa por ti mismo.
"make menuconfig", ya que tienes experiencia en esto, pero en la seccion de Sound no ves nada nombrado como alsa.
Que hacer?
Instalar los fuentes de alsa, los cuales se situaran en:
bash:/usr/src$ ls
alsa-driver.tar.bz2
modules
pcmcia-modules-2.4.21.030701_3.1.33-6+10.00.Custom_i386.deb
Descompacta el alsa-driver tarball y vuelve a crear el symlink. Como podras notar el valor --append-to-version de tu kernel-image; lo necesitaras como opcion para make-kpkg.
bash:/usr/src$ cd linux
bash:/usr/src/linux$ fakeroot make-kpkg --append-to-version=.030701
--added-modules=alsa-driver modules_image
Como podras notar no necesitas el kernel-image aqui. A menos que reconfigures el kernel, no hay necesidad de reconstruirlo. Usando la opcion "--added-modules" permite la construccion solo de los modulos que necesitas, en vez de todos los modulos en /usr/src/modules.
Si deseas construir un modulo .deb con un --revision diferente al del kernel, podrias necesitar apasiguar tu make-kpkg haciendo "fakeroot make-kpkg clean".

8.5. Los patches de kernel de Debian
Debian ofrece un amplia varieda de kernel patches, entre ellos kernel-patch-debianlogo
El patch debianlogo reemplaza a Tux en la parte superior izquierda del framebuffer boot console con otra imagen mas buena como en el 2.4.18. Si no tienes un directorio de kernel-patches, se creara cuando lo instalas tu primer kernel-patch package.
El directorio de kernel-patches es solo para los paquete de Debian kernel-patch. Si tienes otro kernel-patch, ellos deberian en otra parte.

Instalando un paquete de kernel-patch no aplica el patch. Eso recien se logra invocando make-kpkg con la opcion de "--added-patches":
bash:/usr/src/linux$ fakeroot make-kpkg --append-to-version=.030701
--added-patches=debianlogo kernel_image modules_image
Si tienes varios patches para aplicarlos a la lista despues "--added-patches" separados por comas.
Por default make-kpkg corre "make oldconfig" despues de aplicar lo patches del kernel. Si tu parche incluye una nueva opcion del kernel, te lo preguntara, si quieres hacer un review en vez de toda la configuracion, puede usar otra opcion de comando, "--config=menuconfig".

Cuando guardas tu configuracion y sales de menuconfig, make-kpkg continuara.
Despues de que la imagen del kernel este construida, make-kpkg hace un reves de los patches que has aplicado, asi que comenzaras con el ciclo de configure/build con un todavia no patched arbol de fuentes.
9. Checklist
1. Realizar un disco de booteo
2. Instalar los paquetes requeridos
1. gcc, kernel-package, kernel-source-x.y.z, libc6-dev, tk8.3, libncurses5-dev, fakeroot
3. Setup source tree.
1.Agregarte al grupo de src . Logout, login. Usa groups para confirmar.
2.Borrar el viejo symlink.

3.Expandir el source tarball.
4.Hacer un checkeo del nuevo symlink.
5.Hacer un checkeo de los requerimientos minimos
4. Configurar el kernel.
1.Utiliza el viejo archivo de configuracion si esta disponible . cp /boot/config-x.y.z .config
2. make oldconfig o make xconfig o make menuconfig.
3.Guardar el archivo .config en alguna parte para su posterior uso.
5. Construccion de los paquetes de la imagen del kernel
1. fakeroot make-kpkg clean
2. fakeroot make-kpkg --append-to-version=alpha+numeric.but.no.underscore --added-modules=pcmcia-cs,alsa-driver --added-patches=debianlogo,otros ,patches --config=menuconfig kernel_image modules_image
6.Instalar el paquete del kernel-image.
1. dpkg -i kernel-image-x.y.z.yy.mm.dd_10.00.Custom_i386.deb
2. Reboot.
7. Que sigue?
1. Manten tu paquete , echo "kernel-image-x.y.z.yymmdd hold" | dpkg --set-selections
2. Remueve symlink.
3. Has un Back up del kernel image.
4. Remueve la imagen vieja del los paquetes del kernel image
5. Has un nuevo boot diskette.
Y eso eso es todo por ahora es primera vez que escribo algo asi reconmpilando informacion.

Dedicatoria:
En principio a uno de mis teachers el Profesor Carlo Iriondo Sys Admin de la facultad a la cual acudo. Profesor gracias por aguantarme durante tanto tiempo.

Saludos a todos.
Desde San Lorenzo Paraguay
Facultad Politecnica
Universidad Nacional de Asuncion

```
_sudZUZ#Z#XZo=_ DDDD EEEEEE BBBB IIIIII AAAA NN NN
_jmZZ2!!~----~!!X##wa DD DD EE BB BB II AA AA NNN NN
.<wdP~~ -!YZL, DD DD EEEEE BBBBB II AAAAAA NNNN NN
.mX2' _%aaa__ XZ[. DD DD EE BB BB II AA AA NN NNNN
oZ[ _jdXY!~?S#wa ]Xb; DDDD EEEEEE BBBBB IIIIII AA AA NN NN
_#e' .]X2( ~Xw| )XXc
.2Z` ]X[. xY| ]oZ( Linux Version 2.4.21
.2#; )3k; _s!~ jXf` Compiled #1 SMP Tue Jul 22 11:50:10 PYT 2003
1Z> -]Xb/ ~ __#2( One 700MHz Intel Pentium III Processor
-Zo; +!4ZwaaaauZZXY' 128M RAM
*#[, ~-?!!!!!!-~ 1392.64 Bogomips Total
XUb;.
)YXL,,
+3#bc,
-)SSL,,
~~~~~
```

E-mail del autor: juanca _ARROBA_ esdebian.org