

# VI Jornadas sobre el sistema operativo Linux: Recompilación del kernel Linux (2.6 series)

Francisco J.  (Tsao) Santín

e-mail: [tsao@enelparaiso.org](mailto:tsao@enelparaiso.org)

Grupo de Programadores y Usuarios de Linux- Coruña Linux Users  
Group

GPUL-CLUG

25 de Abril de 2006

## **Para despistados...**

¿Que es el kernel (o núcleo) del sistema operativo?

- Es el programa que gestiona los recursos de la máquina a demanda de las aplicaciones
- Linux en sí es un kernel
- En conjunción con las aplicaciones tenemos el Sistema Operativo GNU/Linux
- Con un sistema de instalación a mayores tenemos las distribuciones: Debian, Mandriva, Slackware...

## Algunos tipos de kernels

- Microkernels: Mach, L4
- Kernels monolíticos: Linux
- Kernels megalíticos: Windows

## Un poco de historia...

- Previa:Nacimiento del proyecto GNU en 1985
- Versión 0.01 de Linux Septiembre 1991
- Primera versión oficial:0.02 5 Octubre 1991
- Versión 1.2 primera modular
- ...

## ¿Por qué recompilar el kernel?

- Porque podemos: es software libre
- Porque debemos: optimización/actualización el kernel, integración de nuevas funcionalidades
- Porque queremos: la metafísica de la computación

## **Estructura (básica) del kernel**

El kernel está formado por una base y por una extensión. Esta extensión la forman los módulos:

- Controladores de dispositivos
- Controladores de sistemas de fichero
- System calls
- Controladores de red
- ...

## Requisitos para recompilar el kernel

- Una máquina mínimo 486,4 mb de RAM... ¡Conocimiento del hardware! (lspci y lsusb ayudan)
- El código fuente del kernel, lógicamente. Software adecuado:make,gcc, binutils,module-init-tools(2.6)...
- Documentación adecuada sobre la versión del kernel
- Te/café, galletitas...importante: música adecuada (Miles Davis, John Coltrane, Charlie Parker, GHU!)

## Recompilación del kernel: método clásico (I)

Pasos a seguir:

- Descomprimir el código (un buen sitio: /usr/src)
- Crear el fichero .config con las opciones del kernel: make config/menuconfig/xconfig: escogemos que partes van a formar parte de la base y cuales de la extensión del kernel
- make clean make dep make bzImage/zImage
- make modules

## Recompilación del kernel: método clásico (y II)

- `make modules_install` (copia los modulos a `/lib/modules/x.y.z`, crea ficheros de configuracion de módulos)
- copiar la `bzImage-x.y.z` (en `arch/i386/boot`) a `/boot/vmlinuz-x.y.z`, y el `System.map-x.y.z`
- ... y reconfigurar de nuevo el arranque (lilo/grub)

## Recompilación del kernel: kpkg (Debian)

Necesitamos kernel-package de Debian

- De nuevo, make config/...
- make-kpkg clean
- make-kpkg --revision=custom.x.y kernel\_image
- dpkg -i kernel-image....deb
- ... y de nuevo reconfigurar el bootloader

## **Manejo de módulos: metodo clásico (I)**

Pregunta: ¿Por qué módulos cargables (LKM, Loadable Kernel Modules)? Hay módulos que se utilizan mucho menos que otros, luego si no los cargamos hasta que los usamos, conseguimos:

- mayor rapidez en el sistema (kernel que ocupa menos en memoria)
- más memoria disponible ;-)
- facilidad de paso de parámetros
- facilidad de depuración

## **Manejo de módulos: metodo clásico (II)**

Desventajas del uso de módulos:

- Problemas de seguridad
- Fragmentación de memoria

### **Manejo de módulos: metodo clásico (III)**

Si el kernel está configurado para poder cargarle módulos, podremos evitar en muchos casos recompilarlo, aunque en otras situaciones no nos quedará más remedio que integrar dentro de la base el código de algún módulo

## Manejo de módulos: metodo clásico (IV)

Comandos de manejo de módulos:

- insmod: instalar módulo. Admite argumentos (io,irq...)
- rmmod: borrar módulo
- modprobe: instalar módulo junto con los de que depende

## Manejo de módulos: metodo clásico (V)

- depmod: establecer la dependencia entre módulos
- lsmod: listado de los módulos instalados

## Manejo de módulos: metodo clásico (y VI)

Ficheros de configuración:

- `/etc/modprobe.conf` establece opciones para los módulos
- `/lib/modules/2.x.x/modules.dep` generado por `depmod`, establece dependencias entre módulos
- `/etc/modules` listado de modulos que se cargan al arrancar

El paquete `module-init-tools` debe corresponderse con la version del kernel

## Manejo de módulos. Otras situaciones

- Para compilar aparte módulos de las mismas fuentes, podemos utilizar de igual manera `kpkg`
- Módulos-drivers de fabricantes: suelen tener sus propias instrucciones

## Actualización

- Actualización bruta: repetir el proceso con kernel nuevo
- Actualización fina: “parcheados”

## Actualización.Parches

Aplicación de parches:

- copiamos el parche al mismo directorio que el del código fuente (linux.x.y.z)
- entramos dentro del directorio del fuente
- `bzcat ../patch-x.y.z.bz2 | patch -p1`
- ... y recompilamos  
Cuidado con los parches: es posible que requieran un orden

## Bibliografia

- “The Linux Kernel HOWTO” Linux Documentation Project
- “Linux Loadable Kernel HOWTO”, de Bryan Henderson. Linux Documentation Project

## Direcciones de interés

- <http://www.kernel.org>
- <http://www.kernel-labs.org>
- <http://debian.org>
- <http://tldp.org>