



Compilación del kernel Linux

Arquitectura Interna de Linux - 2016



Compilación del Kernel Linux



Antes de empezar ...

- 1** Arrancar PC del laboratorio en Ubuntu iniciar sesión con usuario "postgrado"
- 2** Descargar este documento del Campus Virtual
- 3** Modificar las propiedades de la máquina virtual de Debian
 - Aumentar el número de cores a 4
 - Opcionalmente aumentar la RAM hasta 4GB
- 4** Arrancar la máquina virtual alterada
- 5** Abrir una ventana de terminal dentro de la máquina virtual



Modificar configuración de la MV de Debian

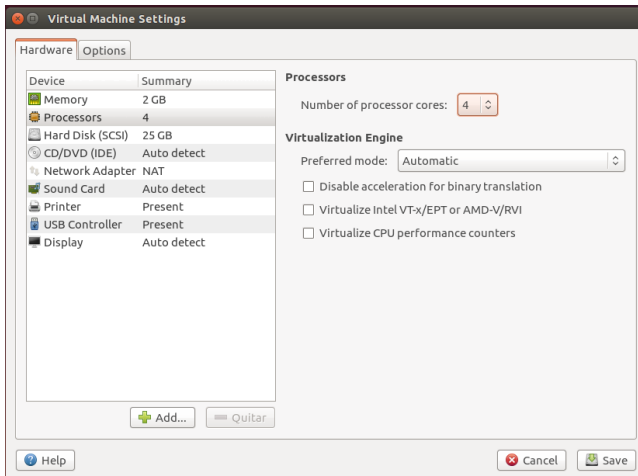


Aumentar el número de cores de la MV a 4

- 1 Virtual Machine -> Virtual Machine Settings
- 2 Hardware [Tab] -> Processors
- 3 Number of processor cores: 4



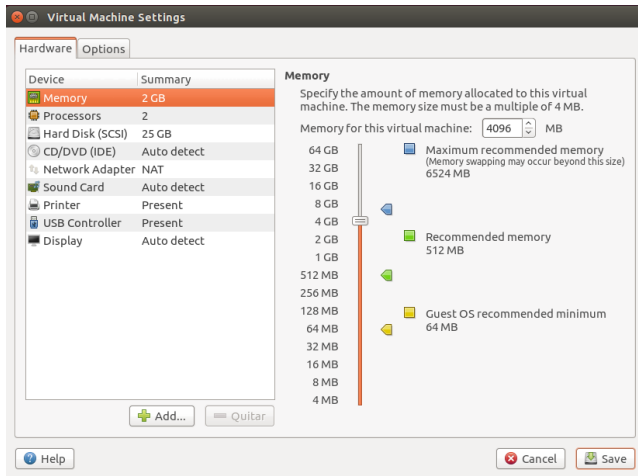
Modificar configuración de la MV de Debian



Modificar # de cores y hacer clic en "Save"



Modificar configuración de la MV de Debian



Opcionalmente aumentar la RAM hasta 4GB

Contenido



1 Compilación tradicional del kernel Linux

2 Compilación a la Debian

- Compilación desde la máquina virtual
- Gestión de paquetes en Debian

3 Otros aspectos



Contenido



1 Compilación tradicional del kernel Linux

2 Compilación a la Debian

- Compilación desde la máquina virtual
- Gestión de paquetes en Debian

3 Otros aspectos



Compilación del Kernel Linux Tradicional (I)



Paso 1: Obtener las fuentes

- Vanilla: (The Linux Kernel Archives) www.kernel.org (repositorio git o tarball)
 - <https://www.kernel.org/pub/linux/kernel/v3.x/linux-3.14.1.tar.gz>
- Las Distros facilitan las fuentes que utilizan (contienen parches).
- Por ejemplo **Debian**:
 - Instalación de paquetes con `apt-get` o `apt source`
 - Paquete de fuentes de `linux-image-*`
 - Paquete `.deb` `linux-source` (instala tarball en `/usr/src/`)
 - Más información en [Debian Linux Kernel Handbook](#)



Compilación del Kernel Linux Tradicional (II)



Paso 2: Configurar el kernel

- ¿Qué componentes o módulos van a formar parte del kernel?
 - La compilación se extiende a todos aquellos módulos escogidos pero no todos quedarán enlazados en el binario final
- Sistema de configuración sofisticado
 - En Linux, un único árbol de fuentes para:
 - 1 Múltiples arquitecturas
 - 2 Sistemas monoprocesador/multiprocesador
 - 3 Características para servidores, equipos de escritorio, portátiles, sistemas de tiempo real,...
- Objetivos
 - 1 Ajustar Linux a nuestras necesidades y a las de la máquina donde se ejecutará
 - 2 Reducir el *footprint* del kernel
 - 3 Reducir el tiempo de compilación



Compilación del Kernel Linux Tradicional (III)



Paso 2: Configurar el kernel (Cont.)

- Gestionado por la herramienta make + scripts (scripts/kconfig)
- Tres alternativas:

1 Modo interactivo:

- `make config / menuconfig / xconfig / gconfig`

2 Modo no interactivo (A) (`make oldconfig`)

- Si disponemos de un fichero de configuración

```
$ cd ${LINUX_SOURCE}  
$ cp /boot/config-3.14.1.lin .config  
$ make oldconfig
```

3 Modo no interactivo (B) - Opciones por defecto (`make defconfig`)

- Busca fichero `.config`
- Si no está presente se escoge la configuración del kernel que esté en ejecución `/boot/config-3.x.y-...`



Compilación del Kernel Linux Tradicional (IV)



Paso 3: Compilar el kernel

■ `make`

- Opcionalmente se puede usar con `-j<N>`
- Resultado: kernel (`vmlinux`) + módulos cargables

Paso 4: Instalación (como root)

1 Instalar módulos en `/lib`

- `make modules_install`

2 Instalar imagen del kernel en `/boot`

- `make install`

3 Incluir entrada en el gestor de arranque (ej.: GRUB, uboot, UEFI)

- Dependiente de la versión del gestor de arranque y de la distribución de GNU/Linux



Contenido



1 Compilación tradicional del kernel Linux

2 Compilación a la Debian

- Compilación desde la máquina virtual
- Gestión de paquetes en Debian

3 Otros aspectos



Compilación a la Debian (I)



Debian style

- Utilidad para construcción de paquetes del kernel: **make-kpkg**
- No sólo se genera la imagen del kernel y los módulos. También se generan los paquetes correspondientes
 - “linux-image*”
 - La instalación de uno de estos paquetes supone la instalación efectiva del kernel (copia /boot, actualizar entradas gestor arranque, ...)
 - “linux-headers*”
 - Instala en /usr/src/linux-headers* los ficheros de cabecera utilizados en la compilación del kernel
 - Con la macros/defines utilizadas en la compilación
 - Útil para compilar drivers (módulos) adicionales sin necesidad de utilizar todas las fuentes



Compilación a la Debian (II)



make-kpkg

- Después de configurar el kernel (*aconsejable*)
- Al igual que make funciona con *targets*:
 - **kernel_image** (imagen)

```
$ make-kpkg --initrd --rootcmd fakeroot \  
--append-to-version ".mikernel" \  
--revision=1.0 kernel_image
```
 - **kernel_headers** (cabeceras específicas)
 - **kernel_source** (código fuente excepto doc)
 - **binary**: image + headers + source + docs
 - **kernel_debug** (símbolos de depuración)
 - **clean**
- Puede explotarse paralelismo utilizando variables de entorno
 - **export CONCURRENCY_LEVEL=5**
 - Se suele escoger número de hilos = `#cores + 1`





Compilar el kernel desde la MV (I)

Obtener fuentes y fichero de configuración

- Las fuentes están ya disponibles en la MV
~/linux-3.14.1.tar.gz
 - Alternativamente, descargar usando [este enlace](#)
- Usaremos el fichero de configuración del kernel en ejecución de la máquina virtual (/boot/config-3.14.1.lin)
- En el tutorial, compilaremos el kernel en el directorio /var/tmp/linux-3.14.1





Compilar el kernel desde la MV (II)

Preparación de las fuentes para la compilación (en /var/tmp)

- Descomprimir las fuentes en /var/tmp:

```
$ cd /var/tmp  
$ tar xzvf ~/linux-3.14.1.tar.gz
```

- Situarse dentro del directorio donde están las fuentes del kernel:

```
$ cd /var/tmp/linux-3.14.1
```

- Copiar el archivo de configuración:

```
$ cp /boot/config-3.14.1.lin .config
```

- Aplicar los cambios del .config. Dos opciones:

- 1 make menuconfig + Cargar .config + Hacer cambios en la configuración + Save
- 2 make oldconfig

■ Más apropiado si no tenemos que modificar .config





make menuconfig (I)

```
.config - Linux/x86 3.14.1 Kernel Configuration

Linux/x86 3.14.1 Kernel Configuration
Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenu ----).
Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes
features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in [ ]
excluded <M> module <-> module capable

[*] 64-bit kernel
  General setup --->
  [ ] Provide system-wide ring of trusted keys
  [*] Enable loadable module support --->
  -* Enable the block layer --->
    Processor type and features --->
    Power management and ACPI options --->
    Bus options (PCI etc.) --->
    Executable file formats / Emulations --->
  -* Networking support --->
    Device Drivers --->
    Firmware Drivers --->
    File systems --->
    Kernel hacking --->
    Security options --->
  -* Cryptographic API --->
  [*] Virtualization --->
    Library routines --->

<Select>  < Exit >  < Help >  < Save >  < Load >
```





make menuconfig (II)

```
.config - Linux/x86 3.14.1 Kernel Configuration

Linux/x86 3.14.1 Kernel Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenu ----).
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes
features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]
excluded <M> module <-> module capable

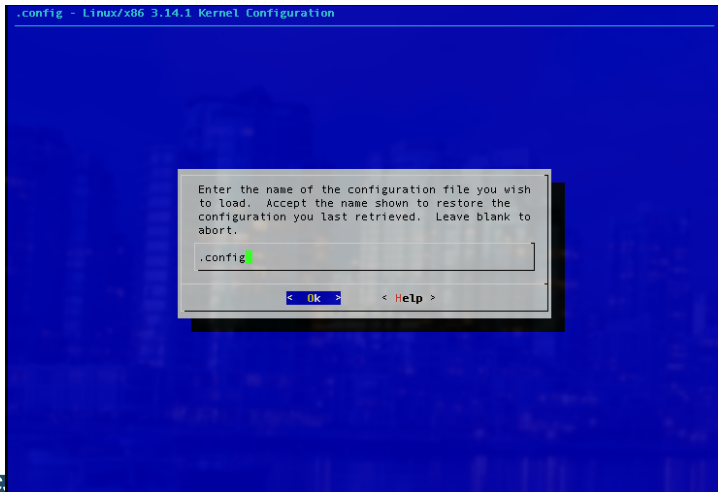
[*] 64-bit kernel
  General setup --->
  [ ] Provide system-wide ring of trusted keys
  [*] Enable loadable module support --->
  -* Enable the block layer --->
    Processor type and features --->
    Power management and ACPI options --->
    Bus options (PCI etc.) --->
    Executable file formats / Emulations --->
  -* Networking support --->
    Device Drivers --->
    Firmware Drivers --->
    File systems --->
    Kernel hacking --->
    Security options --->
  -* Cryptographic API --->
  [*] Virtualization --->
    Library routines --->

<Select>  < Exit >  < Help >  < Save >  < Load >
```





make menuconfig (III)





make menuconfig (IV)

- Modificar la configuración si se desea

```
.config - Linux/x86 3.14.1 Kernel Configuration

Linux/x86 3.14.1 Kernel Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenu ----).
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes
features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]
excluded <M> module < > module capable

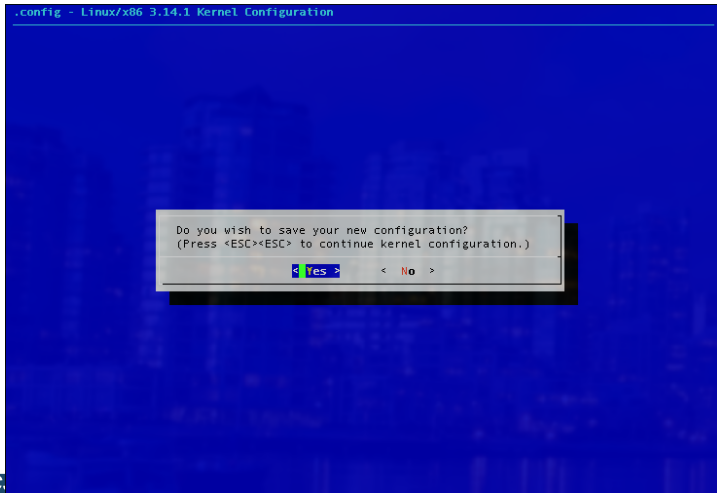
[*] 64-bit kernel
  General setup --->
  [ ] Provide system-wide ring of trusted keys
  [*] Enable loadable module support --->
  -*. Enable the block layer --->
    Processor type and features --->
    Power management and ACPI options --->
    Bus options (PCI etc.) --->
    Executable file formats / Emulations --->
  -*. Networking support --->
    Device Drivers --->
    Firmware Drivers --->
    File systems --->
    Kernel hacking --->
    Security options --->
  -*. Cryptographic API --->
  [*] Virtualization --->
    Library routines --->

<Select> < Exit > < Help > < Save > < Load >
```





make menuconfig (V)





Compilar el kernel desde la MV (III)

Compilación con make-kpkg

- Habilitar compilación con múltiples threads^a

```
$ export CONCURRENCY_LEVEL=5
```

- Compilación

```
$ make-kpkg --rootcmd fakeroot --initrd \  
    --revision=1.0 --append-to-version=".mikernel" \  
    kernel_image kernel_headers
```

^aPara una máquina virtual sin modificar, usar `export CONCURRENCY_LEVEL=3`



Compilar el kernel desde la MV (IV)



Instalación

- ... Y por último se instalan con dpkg

```
$ cd ..
```

```
$ sudo dpkg -i \  
    linux-image-3.14.1.mikernel_1.0_amd64.deb \  
    linux-headers-3.14.1.mikernel_1.0_amd64.deb
```

- Una vez instalado reiniciar el equipo:

```
$ sudo reboot
```

- ... y seleccionar el kernel en el gestor de arranque





Compilación desde la MV original

- La compilación del kernel puede ser **bastante lenta** en la MV original



- 2 Alternativas para acelerar:
 - 1 Añadir más “hierro” a la MV
 - Incrementar el número de cores de la MV y la memoria
 - 2 Compilación desde otro sistema Debian 7.x con más recursos
 - Copiar los paquetes generados a la MV e instalarlos
 - Si sistema no es Debian 7.x, usar *debootstrap*





Paquetes en Debian

■ Dependencias

- Filosofía Unix: proporcionar herramientas específicas
 - Muchas de las aplicaciones se construyen de forma modular combinando/ampliando herramientas específicas (dependen de)
- Las herramientas de gestión de paquetes son una parte fundamental de los distros Linux

■ Paquetes en Debian

- Binarios .deb
 - Aplicaciones, comandos, librerías, documentación,... pueden estar vacíos (virtuales)
- Fuentes src
 - Código fuente + scripts (compilación) para generar .deb





Anatomía de un Paquete Binario (I)

- Nombre de un paquete binario
 - Tres campos separados por “_” (no pueden contener “_”)
 - Nombre del paquete
 - Versión + revisión Debian/Ubuntu
 - Arquitectura: i386, amd64, ...
- Ejemplos:
 - `sudo_1.7.4p4-2.squeeze.4_amd64.deb`
 - `gcc-4.4-multilib_4.4.5-8_amd64.deb`
- Restaurar nombre: `dpkg-name`





Anatomía de un Paquete Binario (II)

Formato .deb:

- archivo BSD ar
- `ar t foo.deb`
 - **Debian-binary**
 - Identifica archivo ar como paquete .deb
 - **control.tar.gz**
 - Tarball con la información de control necesaria por el gestor de paquetes
 - **data.tar.gz**
 - Tarball con el contenido del paquete
- Ejemplo: extraer linux-image*

```
$ sudo -i
$ cd /tmp; mkdir decomp; cd decomp
$ ar x /root/Kernel/linux-image-3.14.1.lin_120814_amd64.deb
$ tar xzvf control.tar.gz
```





Anatomía de un Paquete Binario (III)

Ficheros de control (control.tar.gz)

- **control**: metainformación (único fichero obligatorio)
- **conffiles**: listado de ficheros de “configuración”. Gestión especial para preservar ficheros de configuraciones previas.
- Scripts para la instalación / desinstalación
 - **preinst**
 - **postinst**
 - **prerm**
 - **postrm**
- **md5sums**
 - md5 de los ficheros instalados por el paquete
- Interacción con administrador
 - **config**: obtiene información del admin (debconf db)
 - **templates**: cuestiones que se presentarán al admin





Gestión de Paquetes dpkg

- dpkg, dpkg-deb, dpkg-query
 - dpkg: instalación y eliminación de paquetes
 - Instalar: `-i` ó `--install`: `--unpack + --configure` (postinst)
\$ `dpkg -i <ruta_fichero_deb>`
 - Desinstalar: `-r` ó `--remove` (prerm)
\$ `dpkg -r <nombre_paquete_deb>`
 - `--purge` (postrm)
 - dpkg-deb: manipulación ficheros .deb
 - `--info`, `--contents`, `--field`, `--control`, `--extract`, `--build`
 - dpkg-query: acceso (lectura) bases de datos utilizadas por dpkg (`/var/lib/dpkg`)
 - `--show`, `--status`, `--list`, `--search`





Gestión de Paquetes: apt-get

apt-get: complementa dpkg

- Resolución automática de conflictos
- Interacción con repositorios para adquisición de paquetes
 - Los repositorios se especifican en `/etc/apt/sources.list`
 - deb URI distribución componentes
 - deb-src URI distribución componentes
- Comandos útiles
 - `apt-get install (--reinstall / --download-only)`, `apt-get remove (--purge)`
 - `apt-get update`, `apt-get upgrade`, `apt-get dist-upgrade`
 - `apt-get build-dep`
 - `apt-cache search`, `apt-cache show`
 - `apt-file search`, `apt-file list`
 - `apt-get autoclean (/var/cache/apt/archives)`





Gestión de Paquetes: aptitude

aptitude: complementa dpkg

- Características similares a las de apt-get pero:
 - Gestión más robusta de las dependencias
 - Modo de actualización seguro (safe-upgrade)
 - Interfaz de configuración ncurses (opciones para el administrador)
- Comandos básicos
 - aptitude install, aptitude remove
 - aptitude update, aptitude safe-upgrade, aptitude full-upgrade

■ Ejemplo

```
$ sudo -i
$ aptitude update
$ aptitude install cowsay
$ [Ctrl+D]
$ cowsay "Hola"
```

Contenido



1 Compilación tradicional del kernel Linux

2 Compilación a la Debian

- Compilación desde la máquina virtual
- Gestión de paquetes en Debian

3 Otros aspectos



Gestión de parches



Creación parche

```
## Borrar restos de compilaciones anteriores
$ cd ${DIR_KERNEL_MODIFICADO}
$ make-kpkg clean
$ make mrproper

## Crear parche
$ cd ..
$ diff -urpN ${DIR_KERNEL_ORIGINAL} ${DIR_KERNEL_MODIFICADO} > mi_parche
```

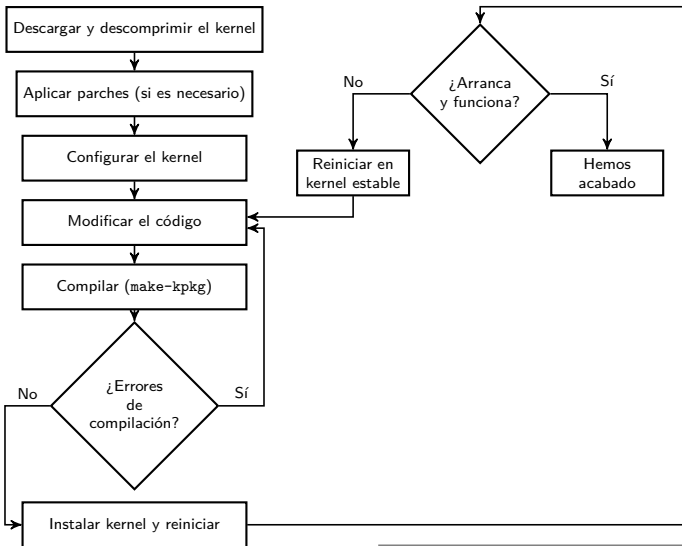
Aplicar el parche

```
## Descomprimir las fuentes originales
$ tar xzvf linux-3.14.1.tar.gz
$ cd linux-3.14.1

## Aplicar parche
$ patch -p1 < ${RUTA_MI_PARCHE}
```



Flujo de trabajo con el kernel



Licencia



Arquitectura Interna de Linux - Compilación del kernel Linux Versión 0.3

©J.C. Sáez

*This work is licensed under the Creative Commons **Attribution-Share Alike 3.0 Spain License**. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/3.0/es/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.*

*Esta obra está bajo una licencia **Reconocimiento-Compartir Bajo La Misma Licencia 3.0 España de Creative Commons**. Para ver una copia de esta licencia, visite <http://creativecommons.org/licenses/by-sa/3.0/es/> o envíe una carta a Creative Commons, 171 Second Street, Suite 300, San Francisco, California 94105, USA.*

Este documento (o uno muy similar) está disponible en
<https://cv4.ucm.es/moodle/course/view.php?id=70009>

