

Configuración de kernel y gestión módulos

Laboratorio de Sistemas

Gorka Guardiola Múzquiz, Enrique Soriano

GSYC

13 de mayo de 2021



(cc) 2018 Grupo de Sistemas y Comunicaciones.

Algunos derechos reservados. Este trabajo se entrega bajo la licencia Creative Commons Reconocimiento -

NoComercial - SinObraDerivada (by-nc-nd). Para obtener la licencia completa, véase

<http://creativecommons.org/licenses/by-sa/2.1/es>. También puede solicitarse a Creative Commons, 559 Nathan

Abbott Way, Stanford, California 94305, USA.

Kernel

- Normalmente usamos un kernel estándar instalado a través de un paquete de la distribución.
- En ocasiones, necesitamos configurar y crear nuestro propio kernel para tener una versión modificada, si estamos usando una plataforma especial o desarrollando parte del kernel (p. ej. un driver), depurando, etc.
- Para compilarlo, necesitamos tiempo y espacio en el disco.

Versiones

Una versión tiene cuatro números: A.B.C.D (p. ej. 4.8.17.2):

- A es el número de versión. Se cambia muy poco, solo cuando hay cambios muy importantes.
- B es el número mayor (major) de revisión.
- C es el número menor (minor) de revisión, se incrementa cuando se mete alguna cosa nueva (driver, etc.).
- D es el de corrección, cuando se aplica un parche para arreglar algo.
- En ocasiones se añaden unas letras como *rc1* (release candidate 1), etc.

Versiones

Hay versiones:

- **Stable:** es la que se recomienda para entornos de de producción, no incluye las últimas novedades experimentales. Los cambios que tiene suelen ser correcciones de bugs. Hasta la versión 2.6.x.x, tenían como número menor de revisión un número par. A partir de esa versión, se cambió el convenio y los impares también son estables.
- **Development:** cambia muy rápido e incorpora funcionalidad que todavía no está muy madura. Hasta la versión 2.6.x.x, tenían un número menor de revisión impar y mayor que el de la versión estable. A partir de ese momento, tienen la misma versión pero acabando en letras como *rc1*, etc.

Para compilar

Primero, instalarse las herramientas de compilación y desarrollo

- <https://wiki.ubuntu.com/Kernel/BuildYourOwnKernel>

- Primero, dependencias:

```
sudo apt build-dep linux linux-image-$(uname -r)
sudo apt install libncurses-dev flex bison \
    openssl libssl-dev dkms libelf-dev libudev-dev \
    libpci-dev libiberty-dev autoconf build-essential\
    qt5-default qtscript5-dev libqwt-qt5-dev pkg-config
sudo apt install git
```

Fuente

El fuente del kernel se puede conseguir de distintos sitios, los más comunes:

- *The Linux Kernel Archives* (sitio oficial) tiene la versión *vanilla* (la versión original):

`https://www.kernel.org/pub/`
`https://git.kernel.org/`

- GitHub de Linus Torvalds:

`https://github.com/torvalds`

Fuente

Es más cómodo obtener el fuente de la distribución. Elegimos esta opción para el curso.

- Está preparado para usar sus herramientas.
- <https://wiki.ubuntu.com/Kernel/BuildYourOwnKernel>
- En Ubuntu, para bajar el fuente por omisión:
`sudo apt install linux-source`
- Si queremos una versión concreta:
`sudo apt source linux-image-<kernel version>`
- Ojo: de esta forma, hay que tener las líneas `deb-src` del fichero `/etc/apt/sources-list` descomentadas.
El paquete deja un fichero tar con el fuente en un directorio de `/usr/src/`.
- Ubuntu también dispone de un Git con el kernel para cada versión de Ubuntu (p. ej. bionic):
`git://kernel.ubuntu.com/ubuntu/ubuntu-<release codename>.git`

Kernels de Ubuntu

- **Mainline (upstream) kernel:** es un kernel *vanilla* con algún cambio de configuración y empaquetado en un `.deb`.
- **Ubuntu kernel:** tienen los parches de los desarrolladores de Canonical, que arreglan bugs y añaden drivers (fuente o binario). Si estamos usando Ubuntu, lo normal es usar uno de estos kernels.
- Tienen su propio convenio para las versiones. Se pueden consultar los códigos de versión de los kernels de Ubuntu:

`http://people.canonical.com/~kernel/info/kernel-version-map.html`

- Los paquetes con kernels ya compilados tienen *flavours*, son versiones configuradas de una cierta manera para un tipo de sistema: *generic*, *server*, etc.

`https://wiki.ubuntu.com/Kernel/Dev/Flavours`

Ejemplo

```
root@vmubuntu:/usr/src# apt install linux-source
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  linux-source-4.15.0
Suggested packages:
  libncurses-dev | ncurses-dev kernel-package libqt3-dev
The following NEW packages will be installed:
  linux-source linux-source-4.15.0
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 128 MB of archives.
After this operation, 145 MB of additional disk space will be used.
Do you want to continue? [Y/n]
Get:1 http://archive.ubuntu.com/ubuntu bionic/main amd64 linux-source-4.15.0 all 4.15.0-20.21 [128 MB]
Get:2 http://archive.ubuntu.com/ubuntu bionic/main amd64 linux-source all 4.15.0.20.23 [2,640 B]
Fetched 128 MB in 1min 12s (1,786 kB/s)
Selecting previously unselected package linux-source-4.15.0.
(Reading database ... 255352 files and directories currently installed.)
Preparing to unpack .../linux-source-4.15.0_4.15.0-20.21_all.deb ...
Unpacking linux-source-4.15.0 (4.15.0-20.21) ...
Selecting previously unselected package linux-source.
Preparing to unpack .../linux-source_4.15.0.20.23_all.deb ...
Unpacking linux-source (4.15.0.20.23) ...
Setting up linux-source-4.15.0 (4.15.0-20.21) ...
Setting up linux-source (4.15.0.20.23) ...
root@vmubuntu:/usr/src# ls linux-source-4.15.0
debian debian.master linux-source-4.15.0.tar.bz2
root@vmubuntu:/usr/src# cd linux-source-4.15.0
root@vmubuntu:/usr/src/linux-source-4.15.0# tar xvjf linux-source-4.15.0.tar.bz2
...
```

Parches

- Otra forma de tener una versión del fuente del kernel es *parcheando* un árbol que ya tenemos.
- Aplicamos un parche con las diferencias al fuente del kernel que ya tenemos bajado.
- El parche reemplaza distintas partes de los ficheros del fuente (p. ej. añade un driver, arregla un bug, etc.).
- P. ej.:

```
$ make clean
```

```
$ zcat grsecurity-3.0-3.17.1-201410250027.patch.gz | patch -p1
```

Configuración

- Una vez descomprimido el *tarball* del fuente (o instalado el fuente del paquete), ya tenemos su directorio con todos los ficheros necesarios.
- Tenemos que estar seguros de que están instaladas todas las herramientas necesarias para compilar un kernel. Una forma sencilla es:

```
sudo apt build-dep linux-image-$(uname -r)
```

NOTA: esto requiere tener descomentadas las líneas de `sources.list` citadas anteriormente y además estar en este momento ejecutando un kernel de Ubuntu.

Configuración

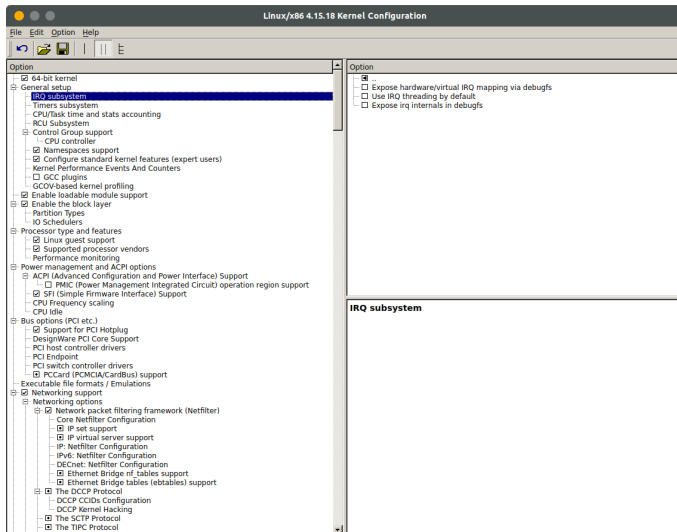
Ahora podemos modificar las opciones para el kernel que queremos compilar:

- Qué queremos incluir en el binario del kernel.
- Qué queremos compilar como módulo del kernel, para poder usarlo opcionalmente cuando se necesite.
- Opciones del kernel: arquitectura, multiprocesador, etc.
- ...

Para ello, podemos ejecutar:

- `make clean`: borra la mayoría de los ficheros generados en otra compilación previa y la configuración
- `make mrproper`: parecido, pero borra más cosas
- `make xconfig`: configuración con interfaz gráfica
- `make menuconfig`: configuración en modo texto con cursores
- `make config`: configuración en modo texto (no recomendada)
- `make oldconfig`: reutiliza un `.config` preguntando
- `make olddefconfig`: reutiliza un `.config` con todo a sí

Configuración: make xconfig



Configuración: make menuconfig

.config - Linux/x86 4.15.18 Kernel Configuration

Linux/x86 4.15.18 Kernel Configuration

Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). 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 --->
[*] 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 --->
    Ubuntu Supplied Third-Party Device Drivers --->
    Firmware Drivers --->
    File systems --->
    Kernel hacking --->
    Security options --->
    *- Cryptographic API --->
[*] Virtualization --->
    Library routines --->
```

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

Configuración: .config

- En `/boot/config-$(uname -r)` está la configuración
- Con la que se compiló ese kernel (hay que poner `CONFIG_SYSTEM_TRUSTED_KEYS=""`) y genera paquetes

Compilación

Una vez que hemos configurado todas las opciones, podemos compilar. Hay distintas formas de compilar.

- La regla `deb-pkg` compila el kernel y genera paquetes `.deb` para instalarlos fácilmente.
- Le podemos dar un nombre de versión para dejar claro que es nuestro kernel.
- Con `-j` especificamos el número de procesadores del sistema que se usarán para compilar (en este ejemplo, todos los que tiene esta máquina):

```
#chown -R paurea.paurea /usr/src/linuxsource-\$(uname -r)
cd /usr/src/linuxsource-\$(uname -r)/
tar -jxvf *.tar.bz
cd /linuxsource-\$(uname -r)
#fakeroot debian/rules clean
make clean
make -j \$(getconf _NPROCESSORS_ONLN) deb-pkg \
```

LOCALVERSION=-laboratorio-sistemas KDEB_PKGVERSION = 1

Compilación

Cada vez hay reglas más estrictas para paquetes debian.

```
apt install kernel-package
#chown -R paurea.paurea /usr/src/linuxsource-\$(uname -r)
cd /usr/src/linuxsource-\$(uname -r)/
tar -jxvf *.tar.bz
fakeroot make-kpkg --initrd \
--revision=1.laboratoriosistemas kernel_image
```

Vete a tomar algo...



- ... porque esto tarda :)
- Depende de tu disco, procesador, etc. de 30 minutos a varias horas.
- Necesitas unos 15 GB de espacio en la partición.

Compilación

En el directorio creado en `/usr/src` dejará varios paquetes `.deb`:

- El kernel (la imagen) y sus módulos.
- Cabeceras del kernel: p. ej. necesarias para poder compilar un módulo para este kernel a posteriori.
- Los símbolos de depuración (dbg).
- Cabeceras relevantes para herramientas de espacio de usuario (p. ej. glibc).
- Firmware que necesitan algunos drivers (puede que no se genere).

```
root@vmubuntu:/usr/src/linux-source-4.15.0# ls *.deb
linux-headers-4.15.18-laboratorio-sistemas_4.15.18-laboratorio-sistemas-1_amd64.deb
linux-image-4.15.18-laboratorio-sistemas_4.15.18-laboratorio-sistemas-1_amd64.deb
linux-image-4.15.18-laboratorio-sistemas-dbg_4.15.18-laboratorio-sistemas-1_amd64.deb
linux-libc-dev_4.15.18-laboratorio-sistemas-1_amd64.deb
```

Instalación

- En la máquina que queramos instalar el kernel, instalamos el paquete `linux-image-*.deb`
- Ese paquete copiará a `/boot`:
 - `vmlinuz-*` es el binario del kernel, la imagen
 - `initrd-*` es el sistema de ficheros de arranque
 - `System.map-*` es la tabla de símbolos del kernel (para depuración)
 - `config-*` es el fichero de configuración con el que se compiló el kernel
- También cambiará la configuración del cargador (`grub`) para que se pueda arrancar el kernel instalado.

Probar el kernel

- Reiniciamos la máquina y pulsamos *Esc* al arrancar para que grub saque el menú.

```
GNU GRUB  version 2.02

Ubuntu
*Advanced options for Ubuntu
Memory test (memtest86+)
Memory test (memtest86+, serial console 115200)

Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the commands
before booting or 'c' for a command-line.
```

Probar el kernel

- Seleccionamos el kernel del que queremos arrancar:

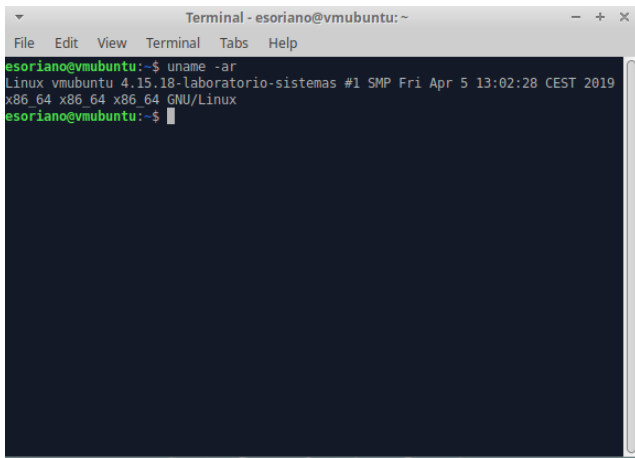
```
GNU GRUB  version 2.02

*Ubuntu, with Linux 4.15.18-laboratorio-sistemas
Ubuntu, with Linux 4.15.18-laboratorio-sistemas (recovery mode)
Ubuntu, with Linux 4.15.0-46-generic
Ubuntu, with Linux 4.15.0-46-generic (recovery mode)
Ubuntu, with Linux 4.15.0-45-generic
Ubuntu, with Linux 4.15.0-45-generic (recovery mode)
Ubuntu, with Linux 4.8.17-custom
Ubuntu, with Linux 4.8.17-custom (recovery mode)

Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the commands
before booting or 'c' for a command-line. ESC to return
previous menu.
```


Probar el kernel

- Arranca el sistema y cuando tengamos un shell, comprobamos la versión:



```
Terminal - esoriano@vmubuntu: ~
File Edit View Terminal Tabs Help
esoriano@vmubuntu:~$ uname -ar
Linux vmubuntu 4.15.18-laboratorio-sistemas #1 SMP Fri Apr 5 13:02:28 CEST 2019
x86_64 x86_64 x86_64 GNU/Linux
esoriano@vmubuntu:~$
```

Grub

Aunque la instalación del paquete que hemos generado ya te configura el cargador, podemos realizar cambios más avanzados:

- El fichero que usará grub en el arranque es:

```
/boot/grub/grub.cfg
```

- Ese fichero es en realidad un script **autogenerado** que **no** debemos editar.
- Tiene todas las entradas para arrancar de los distintos sistemas, en el orden en el que se han instalado. Numera los núcleos empezando desde 0, siendo el 0 el núcleo más reciente.

Grub

Con este comando podemos listar todas las entradas:

```
$ awk -F'"' ' /^ menuentry/{print "entry " n++ ": " $2}' \
/boot/grub/grub.cfg
entry 0: Ubuntu, with Linux 4.15.18-laboratorio-sistemas
entry 1: Ubuntu, with Linux 4.15.18-laboratorio-sistemas (recovery mode)
entry 2: Ubuntu, with Linux 4.15.0-46-generic
entry 3: Ubuntu, with Linux 4.15.0-46-generic (recovery mode)
entry 4: Ubuntu, with Linux 4.15.0-45-generic
entry 5: Ubuntu, with Linux 4.15.0-45-generic (recovery mode)
entry 6: Ubuntu, with Linux 4.8.17-custom
entry 7: Ubuntu, with Linux 4.8.17-custom (recovery mode)
```

Grub

- El directorio `/etc/grub.d` tiene scripts que se encargan de crear las distintas partes del fichero `/boot/grub/grub.cfg` cuando se genera.
- **NO** es recomendable cambiar esos ficheros sin saber bien qué se está haciendo (y realizando antes una copia de seguridad de un fichero `grub.conf` correcto).

Grub

- El fichero `/etc/default/grub` tiene la configuración común de grub.
- Cuando realizamos cambios en ese fichero, debemos ejecutar el comando `update-grub` para que se genere un nuevo `grub.conf`.
- Son pares `clave=valor`

Grub

Algunas claves son:

- GRUB_TIMEOUT: el tiempo que espera el menú (segundos)
- GRUB_DEFAULT: el número de la entrada que se arranca por omisión
- GRUB_CMDLINE_LINUX: los argumentos que se le pasan al kernel (a todas las entradas).

Ejemplo:

```
GRUB_DEFAULT=0
```

```
GRUB_TIMEOUT=0
```

```
GRUB_CMDLINE_LINUX="text" #arranca en modo texto
```

Argumentos del kernel

Hay **muchos**. Algunos son:

- `text` escribe los mensajes en el arranque (modo texto)
- `splash` muestra pantalla con logo (no modo texto)
- `debug` activa la depuración
- `quiet` elimina ciertos mensajes de diagnóstico
- `root=dispositivo` indica el dispositivo para el raíz
- `loglevel=numero` pone el nivel de logging
- `initrd=fichero` indica el fs de arranque a usar
- `module_blacklist=modname1,modname2,modname3` prohíbe la carga de ciertos módulos en el kernel.

Módulos

- Se instalan en

`/lib/modules/<version>/`

- Un módulo sólo funciona en la versión del kernel para la que fue compilado.
- Comandos:
 - `lsmod` muestra los módulos cargados.
 - `modprobe` carga un módulo que tenemos en esos directorios. Mira las dependencias y carga los módulos necesarios.
 - `insmod` carga un módulo a partir de su ruta. No mira dependencias.
 - `rmmmod` descarga un módulo.

Módulos

```
$> cd /lib/modules/$(uname -r)
$> pwd
/lib/modules/4.15.0-46-generic
$> ls kernel/fs/9p/
9p.ko
$> modprobe 9p
$> lsmod | grep 9p
9p                53248  0
9pnet             77824  1 9p
fscache           65536  1 9p
$> rmmod 9p
$> lsmod | grep 9p
9pnet             77824  0
$> rmmod 9pnet
$> lsmod | grep 9pnet
$>
```

Módulos

- modinfo da información sobre un módulo:

```
$> modinfo 9p
filename:      /lib/modules/4.15.0-46-generic/kernel/fs/9p/9p.ko
alias:         fs-9p
license:       GPL
author:        Ron Minnich <rminnich@lanl.gov>
author:        Eric Van Hensbergen <ericvh@gmail.com>
author:        Latchesar Ionkov <lucho@ionkov.net>
srcversion:    6179493B9F8EB6297B76D13
depends:        fscache,9pnet
retpoline:     Y
intree:        Y
name:          9p
vermagic:      4.15.0-46-generic SMP mod_unload
signat:        PKCS#7
signer:
sig_key:
sig_hashalgo:  md4
```

Carga automática

- Systemd se encarga de cargar los módulos en el arranque (systemd-modules-load.service(8)).
- En sistemas sin systemd, `init` carga los módulos que dice `/etc/modules`.
- Estos son los ficheros que mira systemd que contienen las listas de módulos que se deben cargar en el arranque (en este orden de prioridad):
 - `/etc/modules-load.d/*.conf`
 - `/run/modules-load.d/*.conf`
 - `/usr/lib/modules-load.d/*.conf`

Configuración de modprobe

- En `/etc/modprobe.d/` y `/lib/modprobe.d/` tenemos ficheros `.conf` con configuración para modprobe.
- Esos ficheros contienen comandos para poner alias (poner otro nombre a un módulo), prohibir su carga en el arranque (blacklist), opciones para el módulo, etc.
- Los ficheros de blacklist (p. ej. `blacklist.conf`) contienen la lista negra de módulos que modprobe se niega a cargar en arranque (pero si otro módulo lo necesita o si intenta cargar manualmente, se cargarán).