



MANUAL

Sistemas Operativos 1

Los kernel y sus modulos

El documento incluye los pasos para poder realizar la instalacion de kvm para virtualizar maquinas virtuales asi como tambien se incluye la compilacion de un nuevo kernel y su respectiva instalacion en el sistema operativo y por ultimo se da una guia de como crear modulos que puedan ser cargados al kernel compilado.

servio boguerin
200815396

Contenido

Instalacion del hipervisor (KVM)	2
Compilacion e Instalacion de kernel linux	3
Creación de los módulos	10

Instalacion del hipervisor (KVM)

1.-instalar los paquetes qemu

```
root@sopes-PC:/usr/src/linux-4.4.23# apt-get install qemu-kvm qemu-img
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package qemu-kvm
```

2.- ahora que ya se tienen los requerimientos minimos para construir una plataforma virtual se necesitan algunas herramientas administrativas muy útiles para la administración de las plataformas las cuales son:

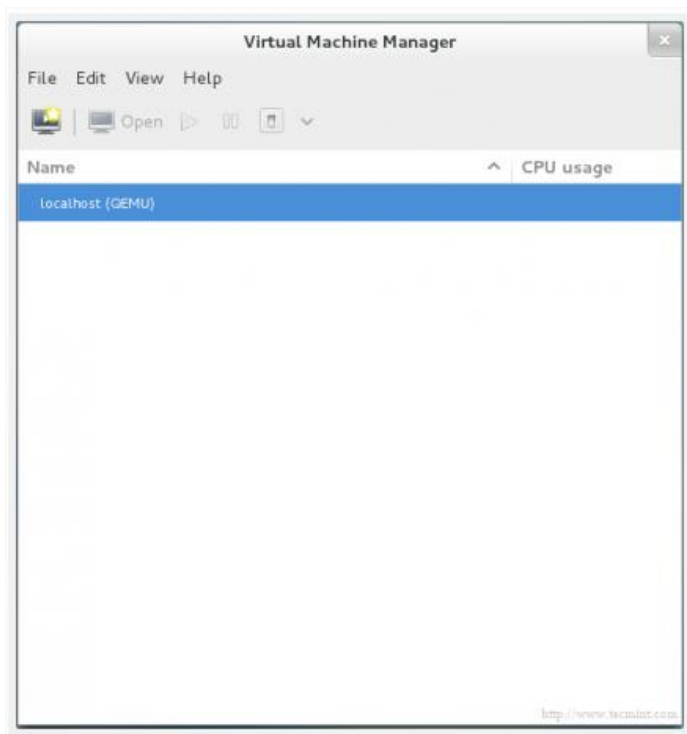
- virt-manager provee una interfaz para administrar las maquinas virtuales.
- libvirt-client provee una Herramienta CL para administrar los ambientes virtuales llamado virsh
- virt-install provee el comando “virt-install” para crear tus propias maquinas virtuales desde un CLI.
- libvirt provee las librerías de los servidores y anfitriones para interactuar con los hipervisores y los sistemas anfitriones.

```
root@sopes-PC:/usr/src/linux-4.4.23# apt-get install virt-manager libvirt libvirt-python libvirt
- client
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

3.- para iniciar a crear arrancamos la interfaz con el comando.

```
root@sopes-PC:/usr/src/linux-4.4.23# virt-manager
The program 'virt-manager' is currently not installed. You
```

4.- nos abrirá una ventana en donde podremos crear nuestras maquinas virtuales



Compilacion e Instalacion de kernel linux

1.- actualizar los repositorios con el comando `apt-get update` con permisos root

```
root@sopes-PC:/usr/src/linux-headers-4.4.0-38# apt-get update
Get:1 http://security.ubuntu.com/ubuntu xenial-security InRelease [94.5 kB]
Get:2 http://gt.archive.ubuntu.com/ubuntu xenial InRelease [247 kB]
Get:3 http://ppa.launchpad.net/elementary-os/stable/ubuntu xenial InRelease [17.5 kB]
```

2.- instalar la libreria ncurses con el comando `apt-get install ncurses-dev` esta libreria es necesaria para compilar el kernel ya que se necesita para mostrar el menuconfig

```
root@sopes-PC:/usr/src/linux-headers-4.4.0-38# apt-get install ncurses-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'libncurses5-dev' instead of 'ncurses-dev'
The following packages were automatically installed and are no longer required:
  gir1.2-javascriptcoregtk-4.0 gir1.2-json-1.0 gir1.2-soup-2.4 gir1.2-webkit2-4.0
  libatkmm-1.6-1v5 libcairomm-1.0-1v5 libglibmm-2.4-1v5 libido3-0.1-0
  libpangomm-1.4-1v5 libsigc++-2.0-0v5 sbsigntool
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
```

3.- instalar la libreria coreutils con el comando `apt-get install coreutils`

```

x Home: apt-get
+ x Home: apt-get
Reading package lists... Done
sopes@sopes-PC:~$ sudo apt-get install coreutils
Reading package lists... Done
Building dependency tree
Reading state information... Done
coreutils is already the newest version (8.25-2ubuntu2).
The following packages were automatically installed and are

```

4.- apt-get install build-essential librerias necesarias para realizar la compilacion del kernel

```

x Home: apt-get
+ x Home: apt-get
sopes@sopes-PC:~$ sudo apt-get install build-essential
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version (12.1ubuntu2).
build-essential set to manually installed.
The following packages were automatically installed and are

```

5.- sudo apt-get install libssl-dev

```

x sudo apt-get install libssl-dev
+ ...pes-PC: /home/sopes x ...t-get install libssl-dev
sopes@sopes-PC:~$ sudo apt-get install libssl-dev
[sudo] password for sopes:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are
no longer required:

```

5.- sudo apt-get install libgtk2.0-dev libglib2.0-dev libglade2-dev librerias necesarias para que funcione el comando make gconfig

```


Setting up libncurses5-dev:amd64 (6.0+20160213-1ubuntu1) ...
sopes@sopes-PC:~$ sudo apt-get install libgtk2.0-dev libglib2.0-dev libgl
ade2-dev

```

6.- descargamos el kernel de la pagina kernel.org el que usaremos nosotros es el 4.4.23


The Linux Kernel Archives

[About](#)
[Contact us](#)
[FAQ](#)
[Releases](#)
[Signatures](#)
[Site news](#)



Protocol	Location
HTTP	https://www.kernel.org/pub/
GIT	https://git.kernel.org/
RSYNC	rsync://rsync.kernel.org/pub/

Latest Stable Kernel:



4.8.1

mainline:	4.8	2016-10-02	[tar.xz]	[pgp]	[patch]	[view diff]	[browse]
stable:	4.8.1	2016-10-07	[tar.xz]	[pgp]	[patch]	[view diff]	[browse]
stable:	4.7.7	2016-10-07	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]
longterm:	4.4.24	2016-10-07	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]
longterm:	4.1.33	2016-09-17	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]
longterm:	3.18.42	2016-09-17	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]
longterm:	3.16.37	2016-08-22	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]
longterm:	3.14.79 [EOL]	2016-09-11	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]
longterm:	3.12.64	2016-10-03	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]
longterm:	3.10.103	2016-08-28	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]
longterm:	3.4.112	2016-04-27	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]
longterm:	3.2.82	2016-08-22	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]
linux-next:	next-20161006	2016-10-06	[tar.xz]	[pgp]	[patch]	[inc. patch]	[view diff]

7.- copiamos el archivo descargado hacia la ruta /usr/src con el siguiente comando `sudo cp -P /home/sopes/Downloads/linux-4.4.23.tar.xz /usr/src`

```

x Home
+ x Home
sopes@sopes-PC:~$ sudo cp -P /home/sopes/Downloads/linux-4.4.23.tar.xz /usr/src

```

8.- descomprimos el archivo con el comando `sudo tar -xvf /usr/linux-4.4.23.tar.xz`

```

x Home
+ x Home
sopes@sopes-PC:~$ sudo tar -xvf /usr/linux-4.6.tar.xz

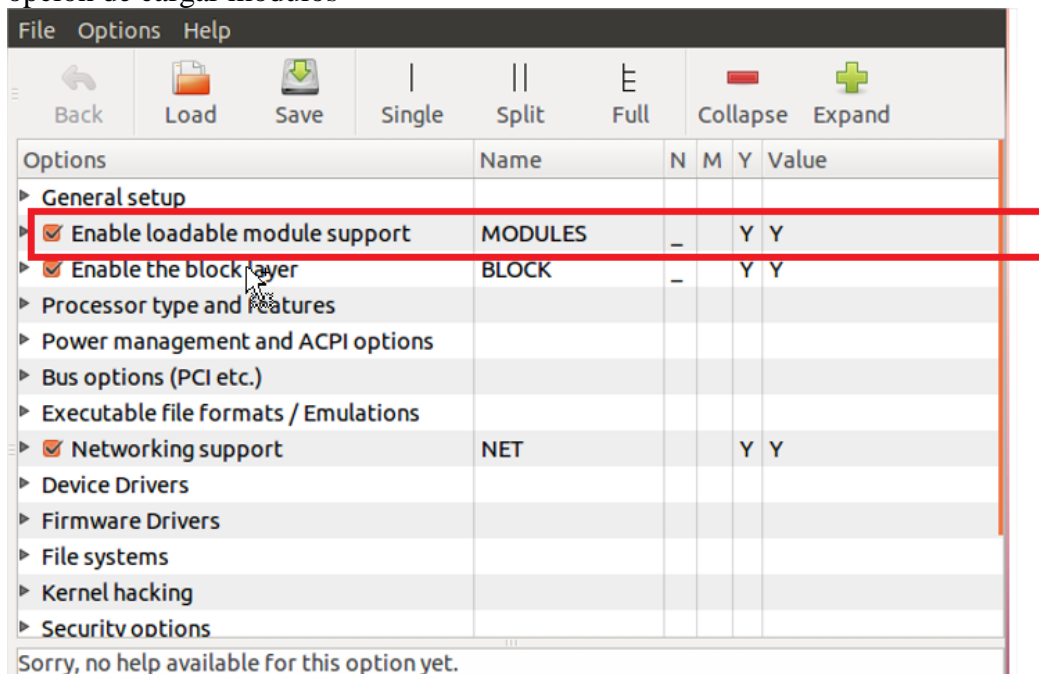
```

9.- cambiamos de directorio hacia la carpeta generada después de descomprimir el archivo con el comando `cd /usr/linux-4.4.23`, seguido de esto ejecutamos los siguientes comandos `make clean`, `make mrproper`, `make localmodconfig`

```
root@sopes-PC: /usr/src/linux-4.4.23
Makefile:959: recipe for target 'arch/x86' failed
make: *** [arch/x86] Interrupt

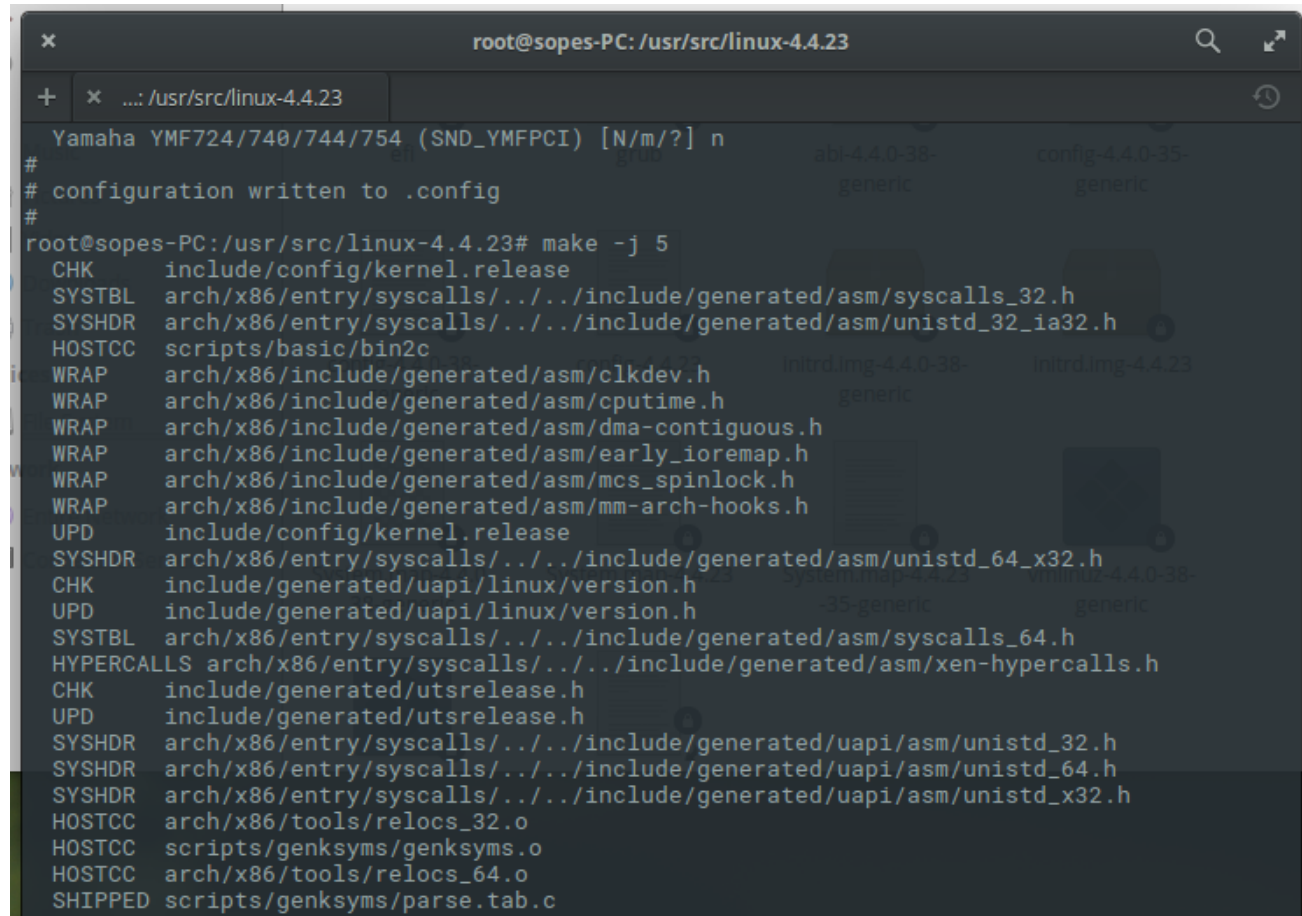
root@sopes-PC:/usr/src/linux-4.4.23# make clean
CLEAN .
CLEAN arch/x86/entry/vdso
CLEAN arch/x86/kernel/cpu
CLEAN arch/x86/kernel
CLEAN arch/x86/purgatory
CLEAN arch/x86/realmode/rm
CLEAN usr
CLEAN arch/x86/tools
CLEAN .tmp_versions
root@sopes-PC:/usr/src/linux-4.4.23# make mrproper
CLEAN scripts/basic
CLEAN scripts/genksyms
CLEAN scripts/kconfig
CLEAN scripts/mod
CLEAN scripts/selinux/genheaders
CLEAN scripts/selinux/mdp
CLEAN scripts
CLEAN include/config include/generated arch/x86/include/generated
CLEAN .config .config.old .version
root@sopes-PC:/usr/src/linux-4.4.23# make localmodconfig
HOSTCC scripts/basic/fixdep
HOSTCC scripts/kconfig/conf.o
SHIPPED scripts/kconfig/zconf.tab.c
SHIPPED scripts/kconfig/zconf.lex.c
SHIPPED scripts/kconfig/zconf.hash.c
HOSTCC scripts/kconfig/zconf.tab.o
```

10.- ejecutamos el comando `make gconfig` nos abrirá una ventana en donde debemos marcar que si a la opción de cargar módulos



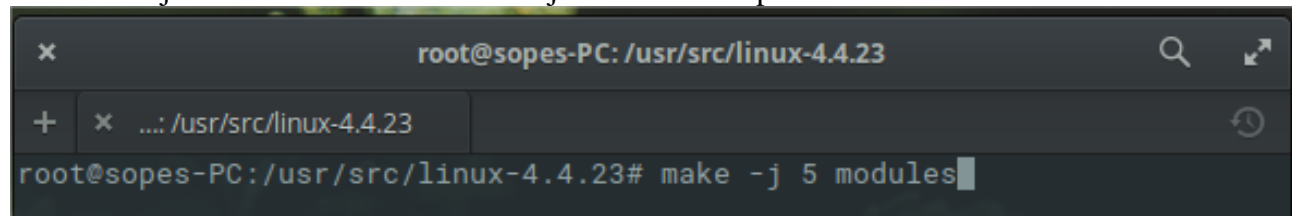
11.- iniciamos la compilación del kernel con el siguiente comando `make -j 5`, esto puede tardar bastante tiempo

Nota: el uso de `-j 5` le indica a make que la compilación la haga en paralelo haciendo uso de los núcleos disponibles en nuestro equipo. Para determinar el número a usar puedes ejecutar `grep processor /proc/cpuinfo | wc -l` y sumar 1 al resultado, para este ejemplo se uso `-j 5` porque la computadora tiene 4 nucleos.



```
root@sopes-PC: /usr/src/linux-4.4.23
# configuration written to .config
root@sopes-PC: /usr/src/linux-4.4.23# make -j 5
CHK      include/config/kernel.release
SYSTBL   arch/x86/entry/syscalls/../../../../include/generated/asm/syscalls_32.h
SYSHDR   arch/x86/entry/syscalls/../../../../include/generated/asm/unistd_32_ia32.h
HOSTCC   scripts/basic/bin2c
WRAP     arch/x86/include/generated/asm/clkdev.h
WRAP     arch/x86/include/generated/asm/cputime.h
WRAP     arch/x86/include/generated/asm/dma-contiguous.h
WRAP     arch/x86/include/generated/asm/early_ioremap.h
WRAP     arch/x86/include/generated/asm/mcs_spinlock.h
WRAP     arch/x86/include/generated/asm/mm-arch-hooks.h
UPD      include/config/kernel.release
SYSHDR   arch/x86/entry/syscalls/../../../../include/generated/asm/unistd_64_x32.h
CHK      include/generated/uapi/linux/version.h
UPD      include/generated/uapi/linux/version.h
SYSTBL   arch/x86/entry/syscalls/../../../../include/generated/asm/syscalls_64.h
HYPERCALLS arch/x86/entry/syscalls/../../../../include/generated/asm/xen-hypercalls.h
CHK      include/generated/utsrelease.h
UPD      include/generated/utsrelease.h
SYSHDR   arch/x86/entry/syscalls/../../../../include/generated/uapi/asm/unistd_32.h
SYSHDR   arch/x86/entry/syscalls/../../../../include/generated/uapi/asm/unistd_64.h
SYSHDR   arch/x86/entry/syscalls/../../../../include/generated/uapi/asm/unistd_x32.h
HOSTCC   arch/x86/tools/relocs_32.o
HOSTCC   scripts/genksyms/genksyms.o
HOSTCC   arch/x86/tools/relocs_64.o
SHIPPED  scripts/genksyms/parse.tab.c
```

12.- ahora ejecutamos el comando `make -j5 modules` este paso no tarda tanto como el anterior



```
root@sopes-PC: /usr/src/linux-4.4.23
root@sopes-PC: /usr/src/linux-4.4.23# make -j 5 modules
```

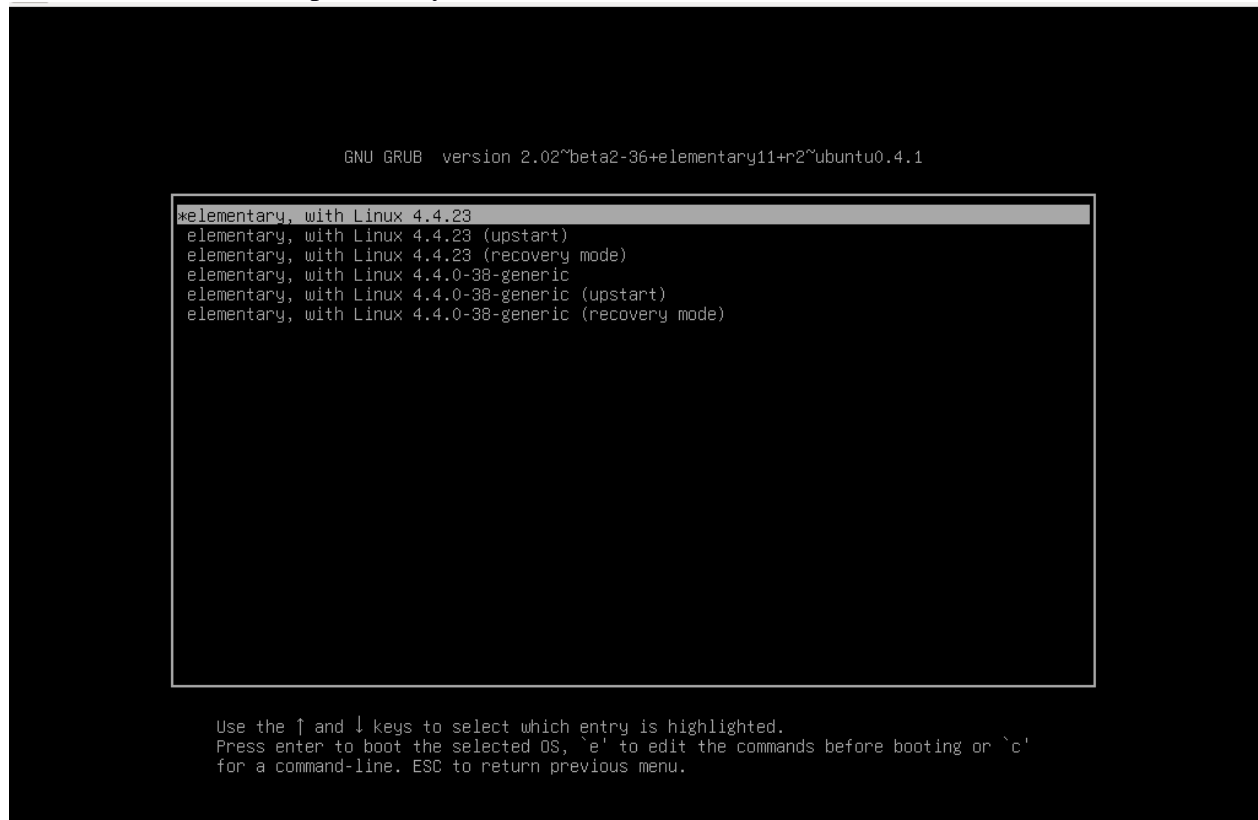

13.- por ultimo ejecutamos el comando `make modules_install install`, este comando lo que hara es automáticamente copiar el resultado de la compilación del kernel en su carpeta destino para que de ultimo solo se actualice el grub para mostrar en el inicio del grub la opción de iniciar con el nuevo kernel

```
root@sopes-PC:/usr/src/linux-4.4.23# make modules_install install
INSTALL arch/x86/kvm/kvm-intel.ko
INSTALL arch/x86/kvm/kvm.ko
INSTALL drivers/ata/pata_acpi.ko
INSTALL drivers/block/floppy.ko
INSTALL drivers/char/lp.ko
INSTALL drivers/char/ppdev.ko
INSTALL drivers/gpu/drm/cirrus/cirrus.ko
INSTALL drivers/gpu/drm/drm.ko
INSTALL drivers/gpu/drm/drm_kms_helper.ko
INSTALL drivers/gpu/drm/ttm/ttm.ko
INSTALL drivers/i2c/algos/i2c-algo-bit.ko
INSTALL drivers/i2c/busses/i2c-piix4.ko
INSTALL drivers/input/input-leds.ko
INSTALL drivers/input/joydev.ko
INSTALL drivers/input/mouse/psmouse.ko
INSTALL drivers/input/serio/serio_raw.ko
INSTALL drivers/macintosh/mac_hid.ko
INSTALL drivers/parport/parport.ko
INSTALL drivers/parport/parport_pc.ko
INSTALL drivers/platform/x86/pvpanic.ko
INSTALL drivers/tty/serial/8250/8250_fintek.ko
INSTALL drivers/video/fbdev/core/fb_sys_fops.ko
INSTALL drivers/video/fbdev/core/syscopyarea.ko
```

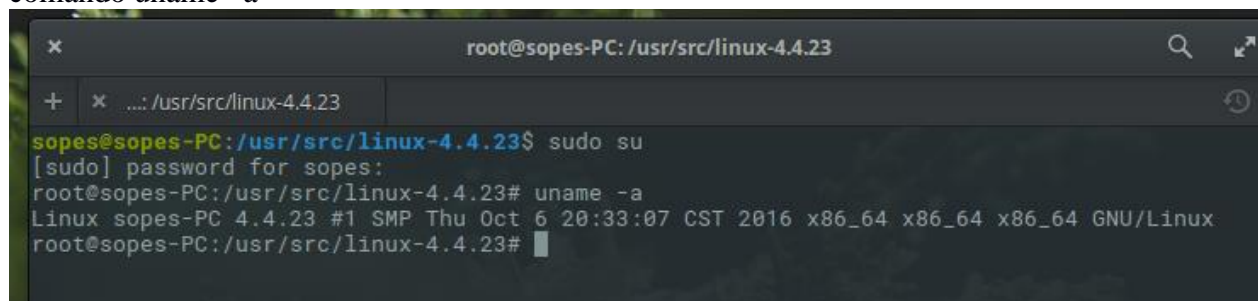
14.- por ultimo actualizamos el grub con el comando `update-grub`

```
root@sopes-PC: /usr/src/linux-4.4.23
+ x ...: /usr/src/linux-4.4.23
root@sopes-PC:/usr/src/linux-4.4.23# update-grub
Generating grub configuration file ...
Warning: Setting GRUB_TIMEOUT to a non-zero value when GRUB_HIDDEN_TIMEOUT is set is no longer supported.
Found linux image: /boot/vmlinuz-4.4.0-38-generic
Found initrd image: /boot/initrd.img-4.4.0-38-generic
```

15.- reiniciamos la computadora y tendremos el nuevo kernel



16.- también una vez iniciado podemos verificar desde la terminal la versión del nuevo kernel con el comando `uname -a`



Creación de los módulos

Primero de todo deberemos incluir algunos **headers** que contienen definiciones que vamos a necesitar:

```
#include <linux/init.h>
#include <linux/module.h>
```

A continuación deberemos definir la **licencia del modulo**:

```
MODULE_LICENSE("GPL");
```

A continuación deberemos definir las funciones de inicialización i destrucción del modulo en las cuales para este sencillo ejemplo haremos un simple **printk**(equivalente de **printf** en el kernel):

```
static int hello_world_init(void)
{
    printk(KERN_ALERT "Hello World!\n");
    return 0;
}
```

En la función de destrucción haremos lo mismo:

```
static void hello_world_exit(void)
{
    printk(KERN_ALERT "Bye World!\n");
}
```

Finalmente deberemos indicar como hemos llamado a las funciones mediante **module_init** y **module_exit**:

```
module_init(hello_world_init);
module_exit(hello_world_exit);
```

Finalmente deberemos crear el fichero de **Makefile**:

```
obj-m += helloworld.o

all:
```

```
make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

Es obligatorio que antes de los comandos “**make**” haya un tabulador y no un conjunto de espacios, sino no va a reconocer el formato del **Makefile**.

Con el comando **make** vamos a compilar el **módulo del kernel**:

```
# make
make -C /lib/modules/2.6.18-53.1.21.el5PAE/build M=/home/jordi/helloworld modules
make[1]: Entering directory `/usr/src/kernels/2.6.18-53.1.21.el5-PAE-i686'
  CC [M]  /home/jordi/helloworld/helloworld.o
  Building modules, stage 2.
  MODPOST
  CC      /home/jordi/helloworld/helloworld.mod.o
  LD [M]  /home/jordi/helloworld/helloworld.ko
make[1]: Leaving directory `/usr/src/kernels/2.6.18-53.1.21.el5-PAE-i686'
```

Mediante un **ls** podemos ver los ficheros que ha generado:

```
# ls
helloworld.c  helloworld.ko  helloworld.mod.c  helloworld.mod.o  helloworld.o
Makefile     Module.symvers
```

Podemos cargar el módulo mediante **insmod**:

```
insmod helloworld.ko
```

A continuación mediante el **dmesg** podremos ver el **Hello World**:

```
audit(1254211500.001:3472101): user pid=1772 uid=0 auid=0 msg='insmod helloworld.ko
'
Hello World!
```

Una forma de ver que el módulo está cargado es mediante **lsmod**:

```
# lsmod | grep hel
helloworld                5504  0
```

Para ver el “**Bye World**” lo descargamos mediante **rmmod**:

```
rmmod helloworld
```

Y de nuevo, con **dmesg**, podremos ver el mensaje de despedida:

```
audit(1254211613.892:3472132): user pid=1772 uid=0 auid=0 msg='rmmod helloworld'
Bye World!
audit(1254211627.421:3472133): user pid=1772 uid=0 auid=0 msg='dmesg '
```

