

## Laboratorio #4

**Fecha de Entrega:** 11 de abril, 2025.

Derek Arreaga - 22537

**Repositorio:** <https://github.com/FabianKel/LAB4-SISTOS>

**Descripción:** este laboratorio reforzará sus conocimientos de diseño e implementación de sistemas operativos con tres ejercicios: creación y carga de un módulo propio al *kernel*; uso de la herramienta SystemTap; e instalación de un *bootstrap program* llamado LILO.

**Entregables:** Debe entregar en Canvas un archivo pdf con sus respuestas a las preguntas planteadas y con las capturas de pantalla solicitadas.

**Materiales:** una máquina virtual Linux.

### Contenido

#### Ejercicio 1 (30 puntos)

- a. Descargue la herramienta SystemTap con el siguiente comando:

```
sudo apt-get install systemtap
```

- b. Cree un archivo llamado `profiler.stp`, con el siguiente código:

```
probe timer.profile{  
  printf("Proceso: %s\n", execname())  
  printf("ID del proceso: %d\n",  
    pid())  
}
```

- c. Ejecute su archivo usando el siguiente comando:

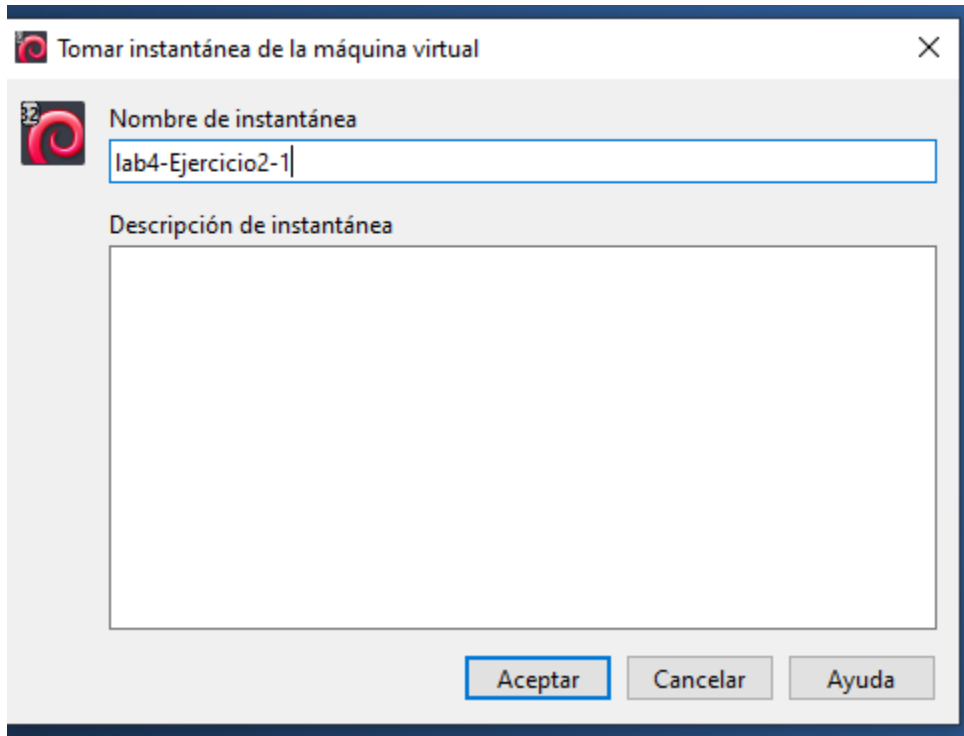
```
sudo stap profiler.stp
```

Durante la ejecución verá mucho *output*. Realice algunas acciones en su sistema operativo sin perder de vista el *output* que la terminal le muestra (*e.g.*, minimice una ventana, abra un archivo de texto, etc.).



**Ejercicio 2 (30 puntos)**

- a. Abra su máquina virtual y tómela una *snapshot*.



- b. Cree un programa en C llamado `simple.c`. Este programa deberá #incluir los siguientes encabezados:
- `<linux/init.h>`
  - `<linux/kernel.h>`
  - `<linux/module.h>`
  - `<linux/list.h>`
- c. Escriba dos métodos en su programa llamados `simple_init` y `simple_exit`. Ambos métodos deben declarar como parámetro únicamente `void`, y el primero debe retornar tipo `int` mientras que el segundo tipo `void`. El primer método debe devolver cero.
- ¿Cuál es la diferencia en C entre un método que no recibe parámetros y uno que recibe `void`?
- Declarar una función sin parámetros no garantiza que la función no reciba parámetros. Significa que la función puede aceptar cualquier número de parámetros. Esto es un comportamiento heredado de C para mantener compatibilidad con código antiguo.
- Por otro lado, al declarar una función con parámetro `void`, se está indicando explícitamente que la función no acepta ningún parámetro. Es una forma más estricta y clara de definir la interfaz de la función, evitando errores.

- d. En el primer método incluya la siguiente instrucción:

```
printk(KERN_INFO "Loading Module\nSistops");
```

Reemplace el texto `Sistops` por un mensaje personalizado. En el segundo incluya la siguiente instrucción:

```
printk(KERN_INFO "Removing Module\nSistops");
```

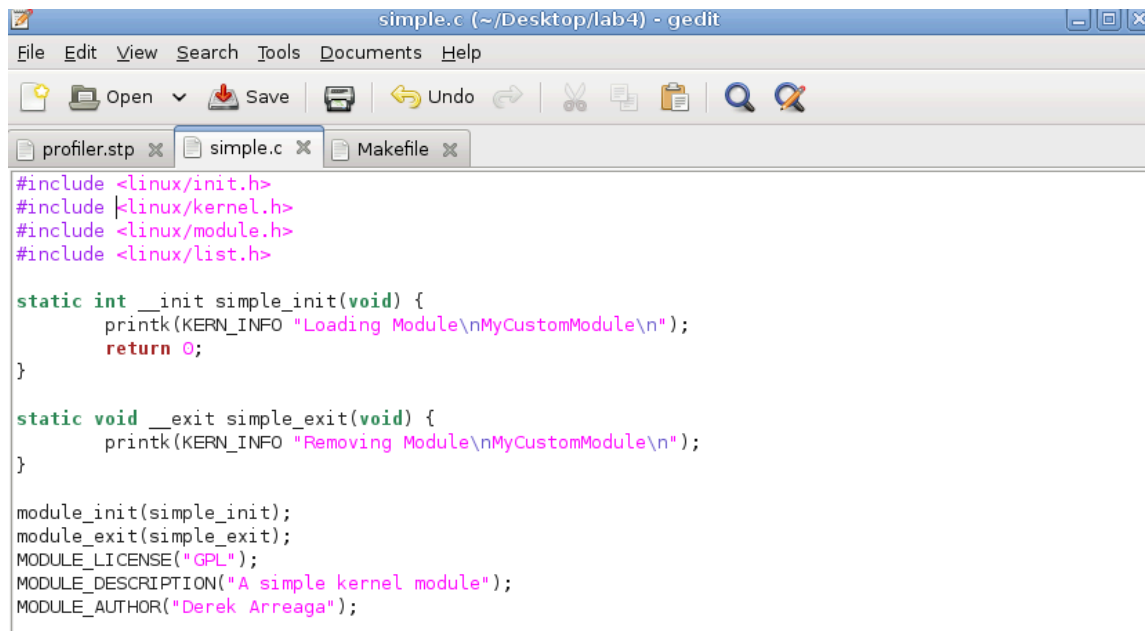
Nuevamente reemplace el texto `Sistops` por un mensaje personalizado.

- ¿Qué diferencia hay entre `printk` y `printf`?
  - El `printk` sirve para imprimir mensajes en el buffer de log del kernel, lo cual solo se puede ver con las herramientas como `dmesg`, es seguro para usar en el contexto del kernel, donde no hay acceso a `stdout`. Por otro lado el `printf` es utilizada en espacio de usuario, funciona para imprimir mensajes en la salida estándar.
- ¿Qué es y para qué sirve `KERN_INFO`?
  - Es una constante de nivel de prioridad, se usa con `printk` en el kernel de linux.
  - `KERN_INFO` tiene una prioridad de nivel 6 y funciona para categorizar mensajes en el log del kernel para poder analizar y filtrar los procesos.

- e. Abajo de sus dos métodos incluya las siguientes instrucciones (reemplazando `<Su nombre>` con su nombre y `<Descripcion>` con una descripción personalizada):

```
module_init(simple_
init);
module_exit(simple_
exit);
MODULE_LICENSE("GPL
");
MODULE_DESCRIPTION("<Descri
pcion>");
MODULE_AUTHOR("<Su
nombre>");
```

Grabe su programa.



```
simple.c (~/Desktop/lab4) - gedit
File Edit View Search Tools Documents Help
Open Save Undo Cut Copy Paste Find
profiler.stp x simple.c x Makefile x
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/list.h>

static int __init simple_init(void) {
    printk(KERN_INFO "Loading Module\nMyCustomModule\n");
    return 0;
}

static void __exit simple_exit(void) {
    printk(KERN_INFO "Removing Module\nMyCustomModule\n");
}

module_init(simple_init);
module_exit(simple_exit);
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("A simple kernel module");
MODULE_AUTHOR("Derek Arreaga");
```

f. Cree un archivo *Makefile* para su programa, que contenga el siguiente código:

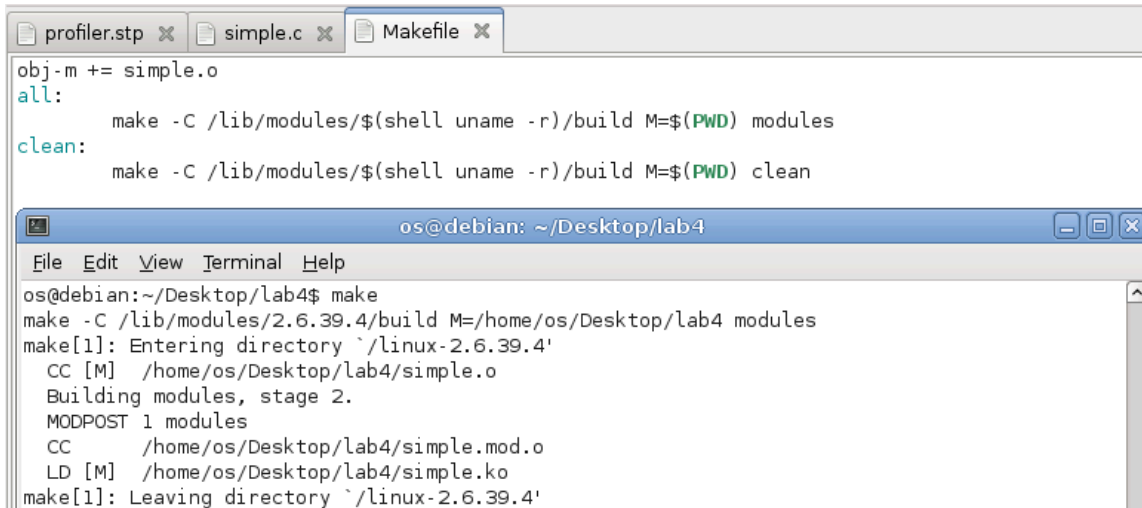
```
obj-m += simple.o

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

- ¿Qué es una **goal definition** o definición de meta en un *Makefile*, y qué se está haciendo con la definición de meta `obj-m`?  
Una goal definition en un Makefile es una regla que especifica un target, son como las dependencias para construirlo y comandos para generarlo.  
En el caso la meta `obj-m`, se está definiendo una lista de módulos del kernel que se construirán como módulos `.ko`
- ¿Qué función tienen las líneas `all:` y `clean:`?  
`all` define la sección de las acciones que se realizarán al ejecutar con `make`  
`clean` define la sección de las acciones que se realizarán al ejecutar con `make clean`, que usualmente es para remover los archivos generados cuando hayan cambios en el código fuente para posteriormente ejecutar `make` otra vez
- ¿Qué hace la opción `-C` en este *Makefile*?  
la opción `-C` cambia el directorio de trabajo antes de ejecutar lo del Makefile para usar el sistema de compilación del kernel para generar el módulo
- ¿Qué hace la opción `M` en este *Makefile*?  
La opción `M` es para especificar el directorio donde se encuentran los archivos fuentes del módulo.

- g. Ejecute el comando `make` en el directorio donde haya creado `simple.c` y su correspondiente *Makefile*.

```
sudo insmod  
simple.ko dmesg
```



The screenshot shows a terminal window titled 'os@debian: ~/Desktop/lab4'. The terminal output for the 'make' command is as follows:

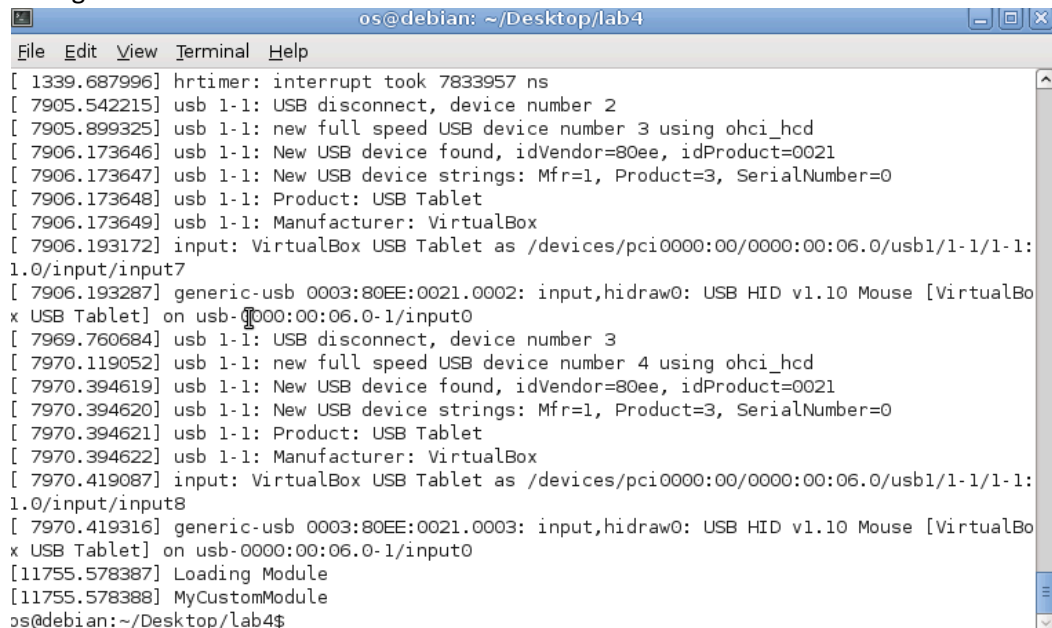
```
obj-m += simple.o  
all:  
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules  
clean:  
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean  
  
os@debian:~/Desktop/lab4$ make  
make -C /lib/modules/2.6.39.4/build M=/home/os/Desktop/lab4 modules  
make[1]: Entering directory `/linux-2.6.39.4'  
  CC [M] /home/os/Desktop/lab4/simple.o  
Building modules, stage 2.  
MODPOST 1 modules  
  CC      /home/os/Desktop/lab4/simple.mod.o  
  LD [M] /home/os/Desktop/lab4/simple.ko  
make[1]: Leaving directory `/linux-2.6.39.4'
```

- h. Ejecute los siguientes comandos:

- `sudo insmod simple.ko`  

```
os@debian:~/Desktop/lab4$ sudo insmod simple.ko  
[sudo] password for os:  
os@debian:~/Desktop/lab4$
```

- `dmesg`



The screenshot shows a terminal window titled 'os@debian: ~/Desktop/lab4'. The terminal output for the 'dmesg' command is as follows:

```
[ 1339.687996] hrtimer: interrupt took 7833957 ns  
[ 7905.542215] usb 1-1: USB disconnect, device number 2  
[ 7905.899325] usb 1-1: new full speed USB device number 3 using ohci_hcd  
[ 7906.173646] usb 1-1: New USB device found, idVendor=80ee, idProduct=0021  
[ 7906.173647] usb 1-1: New USB device strings: Mfr=1, Product=3, SerialNumber=0  
[ 7906.173648] usb 1-1: Product: USB Tablet  
[ 7906.173649] usb 1-1: Manufacturer: VirtualBox  
[ 7906.193172] input: VirtualBox USB Tablet as /devices/pci0000:00/0000:00:06.0/usb1/1-1/1-1:1.0/input/input7  
[ 7906.193287] generic-usb 0003:80EE:0021.0002: input,hidraw0: USB HID v1.10 Mouse [VirtualBox USB Tablet] on usb-0000:00:06.0-l/input0  
[ 7969.760684] usb 1-1: USB disconnect, device number 3  
[ 7970.119052] usb 1-1: new full speed USB device number 4 using ohci_hcd  
[ 7970.394619] usb 1-1: New USB device found, idVendor=80ee, idProduct=0021  
[ 7970.394620] usb 1-1: New USB device strings: Mfr=1, Product=3, SerialNumber=0  
[ 7970.394621] usb 1-1: Product: USB Tablet  
[ 7970.394622] usb 1-1: Manufacturer: VirtualBox  
[ 7970.419087] input: VirtualBox USB Tablet as /devices/pci0000:00/0000:00:06.0/usb1/1-1/1-1:1.0/input/input8  
[ 7970.419316] generic-usb 0003:80EE:0021.0003: input,hidraw0: USB HID v1.10 Mouse [VirtualBox USB Tablet] on usb-0000:00:06.0-l/input0  
[11755.578387] Loading Module  
[11755.578388] MyCustomModule  
os@debian:~/Desktop/lab4$
```

- ¿Para qué sirve `dmesg`?

Significa Diagnostic Message, funciona para mostrar los mensajes almacenados en el buffer de log del kernel, en nuestro caso al final se observan los mensajes del módulo que creamos.

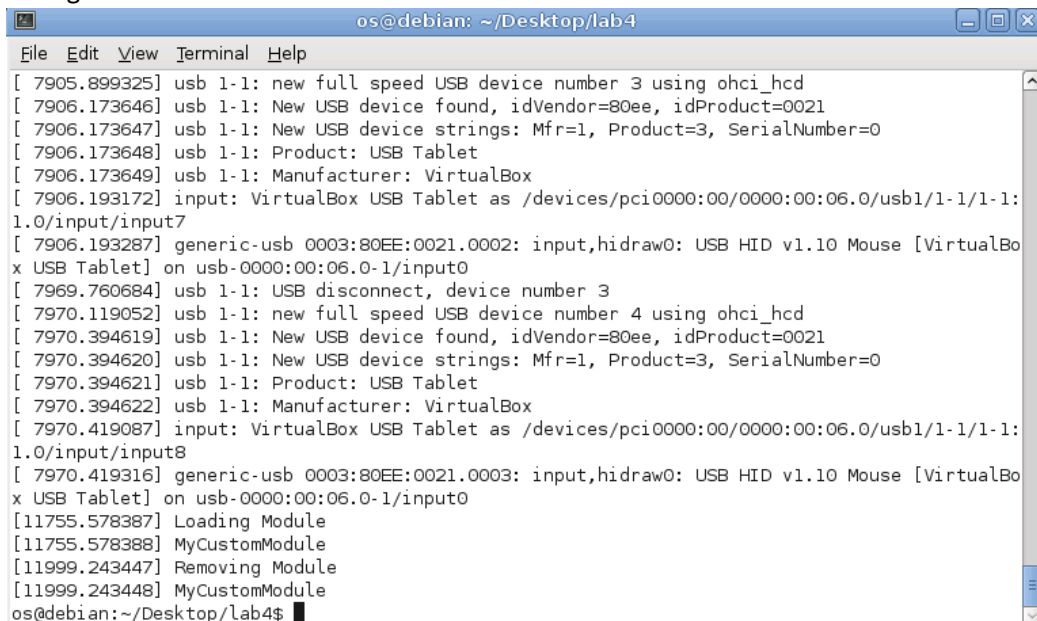
- ¿Qué hace la función `simple_init` en su programa `simple.c`?  
`simple_init` es un punto de entrada del módulo cuando se carga con `insmod`, es para realizar tareas de inicialización del módulo.

i. Ahora ejecute los siguientes comandos:

- `sudo rmmod simple`

```
os@debian:~/Desktop/lab4$ sudo rmmod simple
os@debian:~/Desktop/lab4$
```

- `dmesg`

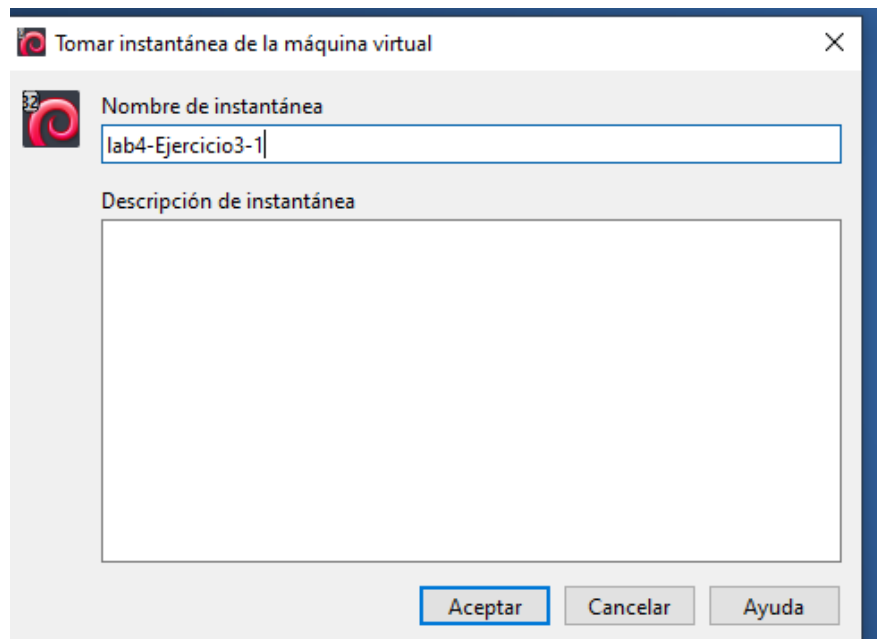


```
os@debian: ~/Desktop/lab4
File Edit View Terminal Help
[ 7905.899325] usb 1-1: new full speed USB device number 3 using ohci_hcd
[ 7906.173646] usb 1-1: New USB device found, idVendor=80ee, idProduct=0021
[ 7906.173647] usb 1-1: New USB device strings: Mfr=1, Product=3, SerialNumber=0
[ 7906.173648] usb 1-1: Product: USB Tablet
[ 7906.173649] usb 1-1: Manufacturer: VirtualBox
[ 7906.193172] input: VirtualBox USB Tablet as /devices/pci0000:00/0000:00:06.0/usb1/1-1/1-1:1.0/input/input7
[ 7906.193287] generic-usb 0003:80EE:0021.0002: input,hidraw0: USB HID v1.10 Mouse [VirtualBox USB Tablet] on usb-0000:00:06.0-l/input0
[ 7969.760684] usb 1-1: USB disconnect, device number 3
[ 7970.119052] usb 1-1: new full speed USB device number 4 using ohci_hcd
[ 7970.394619] usb 1-1: New USB device found, idVendor=80ee, idProduct=0021
[ 7970.394620] usb 1-1: New USB device strings: Mfr=1, Product=3, SerialNumber=0
[ 7970.394621] usb 1-1: Product: USB Tablet
[ 7970.394622] usb 1-1: Manufacturer: VirtualBox
[ 7970.419087] input: VirtualBox USB Tablet as /devices/pci0000:00/0000:00:06.0/usb1/1-1/1-1:1.0/input/input8
[ 7970.419316] generic-usb 0003:80EE:0021.0003: input,hidraw0: USB HID v1.10 Mouse [VirtualBox USB Tablet] on usb-0000:00:06.0-l/input0
[11755.578387] Loading Module
[11755.578388] MyCustomModule
[11999.243447] Removing Module
[11999.243448] MyCustomModule
os@debian:~/Desktop/lab4$
```

- ¿Qué hace la función `simple_exit` en su programa `simple.c`?  
Es para marcar el punto de salida del módulo-
- Usted ha logrado crear, cargar y descargar un módulo de Linux. ¿Qué poder otorga el ejecutar código de esta forma?  
Otorga un gran poder para el desarrollador, nos permite tener un acceso directo al kernel, se puede acceder a estructuras internas como procesos, memorias o dispositivos. También nos permite personalizar el sistema añadiendo controladores para hardware o funciones personalizadas.

### **Ejercicio 3 (40 puntos)**

- a. Si todo ha salido bien con los demás ejercicios, tómese una *snapshot* a su máquina virtual. De lo contrario no continúe con este ejercicio y complete los demás, asegurándose de que su sistema queda estable. Repito: **no continúe este ejercicio sin sacar una *snapshot* estable de su máquina primero**. Deje constancia de los pasos seguidos y comandos ejecutados apoyándose de screenshots.



- b. Ejecute el siguiente comando en una terminal (note el guion al final):  
`sudo apt-get --purge install lilo grub-legacy-`

Durante la instalación aparecerá una pantalla que le indicará ejecutar `liloconfig` y `/sbin/lilo` más adelante. Presione *Enter* e ignórela. Estos comandos harían automáticamente lo que los siguientes incisos le ayudarán a hacer “a pie”.

```
os@debian: ~  
File Edit View Terminal Help  
Need to get 420 kB of archives.  
After this operation, 1,249 kB of additional disk space will be used.  
Do you want to continue [Y/n]? y  
WARNING: The following packages cannot be authenticated!  
mbr lilo  
Install these packages without verification [y/N]? y  
Err http://ftp.us.debian.org/debian/ squeeze/main mbr i386 1.1.10-2  
404 Not Found [IP: 208.80.154.139 80]  
Get:1 http://archive.debian.org/debian/ squeeze/main mbr i386 1.1.10-2 [22.9 kB]  
Get:2 http://archive.debian.org/debian/ squeeze/main lilo i386 1:22.8-10 [397 kB]  
]  
Fetched 420 kB in 0s (501 kB/s)  
Preconfiguring packages ...  
Selecting previously deselected package mbr.  
(Reading database ... 134426 files and directories currently installed.)  
Unpacking mbr (from .../archives/mbr_1.1.10-2_i386.deb) ...  
Selecting previously deselected package lilo.  
Unpacking lilo (from .../lilo_1%3a22.8-10_i386.deb) ...  
Processing triggers for man-db ...  
Processing triggers for menu ...  
Setting up mbr (1.1.10-2) ...  
Setting up lilo (1:22.8-10) ...  
Processing triggers for menu ...  
os@debian:~$
```

- c. Vaya al directorio `/dev/disk/by-id` y ejecute el comando `ls -Al`. El resultado le mostrará varios *links* simbólicos (si está utilizando la máquina virtual de Linux Mint los links están marcados en un celeste brillante), algunos de los cuales se dirigen a algo igual o parecido a



../../sda. Anote el nombre del *link* que no incluye algo como “partN” y que apunta exactamente a ../../sda. Además, anote la ruta completa de la ubicación del *link*.

```
os@debian: /dev
File Edit View Terminal Help
root@debian:/dev/disk/by-id# ls -Al
total 0
lrwxrwxrwx 1 root root 9 Apr 11 16:40 ata-VBOX_CD-ROM_VB2-01700376 -> ../../hdc
lrwxrwxrwx 1 root root 9 Apr 11 16:40 ata-VBOX_HARDDISK_VBd07caeba-e4659d70 -> ../../sda
lrwxrwxrwx 1 root root 10 Apr 11 16:40 ata-VBOX_HARDDISK_VBd07caeba-e4659d70-part1 -> ../../sda1
lrwxrwxrwx 1 root root 10 Apr 11 16:40 ata-VBOX_HARDDISK_VBd07caeba-e4659d70-part2 -> ../../sda2
lrwxrwxrwx 1 root root 10 Apr 11 16:40 ata-VBOX_HARDDISK_VBd07caeba-e4659d70-part5 -> ../../sda5
lrwxrwxrwx 1 root root 9 Apr 11 16:40 scsi-SATA_VBOX_HARDDISK_VBd07caeba-e4659d70 -> ../../sda
lrwxrwxrwx 1 root root 10 Apr 11 16:40 scsi-SATA_VBOX_HARDDISK_VBd07caeba-e4659d70-part1 -> ../../sda1
lrwxrwxrwx 1 root root 10 Apr 11 16:40 scsi-SATA_VBOX_HARDDISK_VBd07caeba-e4659d70-part2 -> ../../sda2
lrwxrwxrwx 1 root root 10 Apr 11 16:40 scsi-SATA_VBOX_HARDDISK_VBd07caeba-e4659d70-part5 -> ../../sda5
```

- d. Vaya al directorio `/etc` y lea el contenido del archivo `fstab`. Verá una tabla (probablemente desalineada) y deberá buscar la fila cuya columna llamada `<mount point>` contenga `/`. De esa fila anote el contenido de la columna `<file system>`.

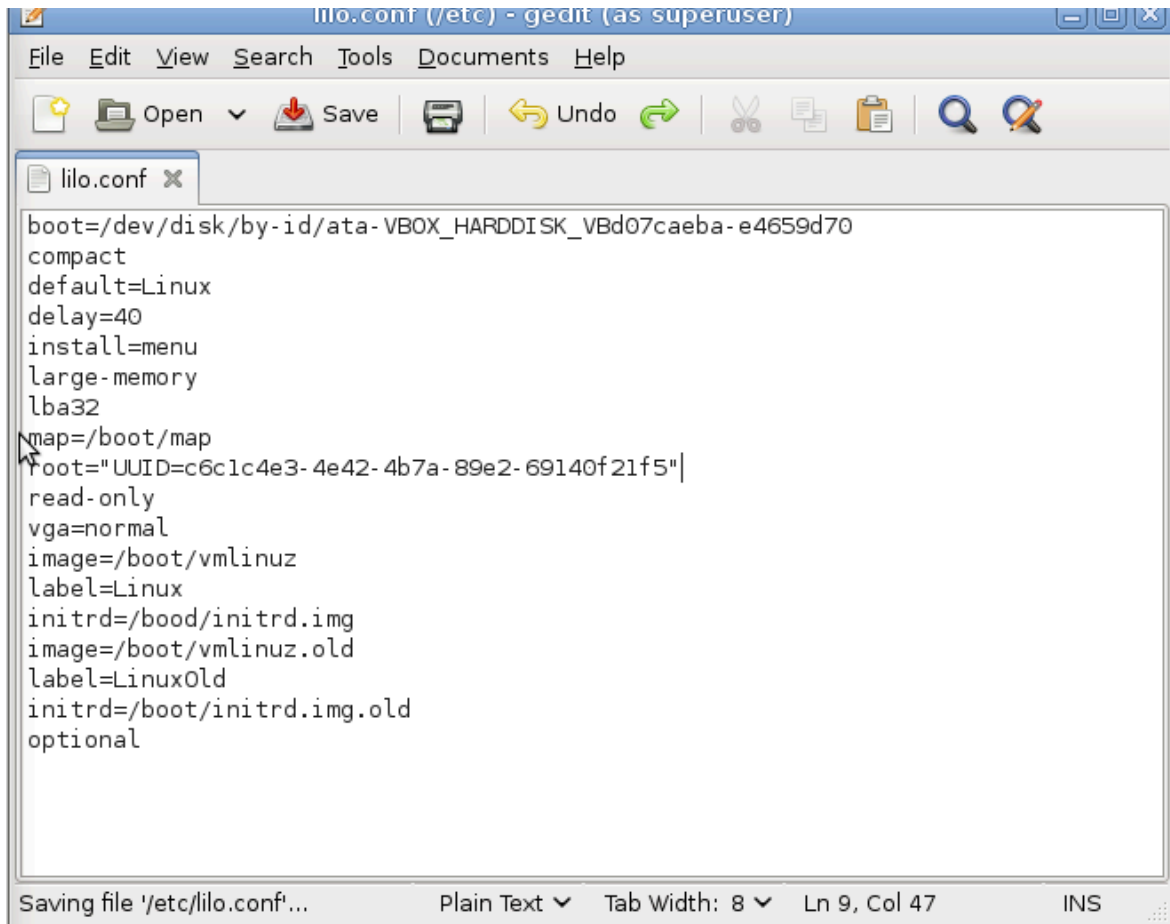
```
os@debian: /dev
File Edit View Terminal Help
root@debian:/etc# cat fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options> <dump> <pass>
proc /proc proc defaults 0 0
# / was on /dev/sda1 during installation
UUID=c6c1c4e3-4e42-4b7a-89e2-69140f21f585 / ext3 errors=remount-ro 0 1
# swap was on /dev/sda5 during installation
UUID=e2976f5e-f45d-4d31-8a8f-ac19d4f04620 none swap sw 0 0
/dev/scd0 /media/cdrom0 udf,iso9660 user,noauto 0 0
```

- ¿Qué es y para qué sirve el archivo `fstab`?  
Es un archivo de configuración que lista los sistemas de archivos que se montan automáticamente al iniciar el sistema. Sirve para especificar dispositivos, puntos de montaje, tipos de sistemas de archivos, y opciones de montaje.
- Usado por el sistema para montar particiones en el arranque o con comandos como `mount -a`.
  - ¿Qué almacena el directorio `/etc`? ¿En Windows, quién (hasta cierto punto) funge como `/etc`?  
Almacena archivos de configuración del sistema y de aplicaciones en Linux  
En Windows puede ser la carpeta `C:\Windows\System32\config`, ya que almacena configuraciones del sistema y aplicaciones.
  - ¿Qué se almacena en `/dev` y en `/dev/disk`?  
En `/dev` se almacenan archivos especiales para representar dispositivos físicos y virtuales, también se almacenan dispositivos de bloque como discos y particiones.  
En `/dev/disk` se organizan enlaces simbólicos a dispositivos de bloque para referencias discos de manera consistente.

- e. En ese mismo directorio `/etc` cree un archivo llamado `lilo.conf` que contenga lo siguiente:

```
boot=<la dirección completa del link  
hacia sda> compact  
default=Li  
nux  
delay=40  
install=me  
nu  
large-memo  
ry lba32  
map=/boot/  
map  
root="<el file system anotado>"  
read-only  
vga=normal  
image=/boot/vm  
linux  
label=Linux  
    initrd=/boot/initrd.img  
image=/boot/vmlinuz.old  
    label=LinuxOld  
    initrd=/boot/initrd.img  
    .old optional
```

En este archivo debe reemplazar <la dirección completa del link hacia sda> con la dirección **absoluta** hacia el *link* que anotó en el inciso c; y <el file system anotado> con lo que anotó en el inciso d (note que <el file system anotado> está rodeado de comillas).



```
lilo.conf (/etc) - gedit (as superuser)
File Edit View Search Tools Documents Help
Open Save Undo
lilo.conf x
boot=/dev/disk/by-id/ata-VBOX_HARDDISK_VBd07cae8a-e4659d70
compact
default=Linux
delay=40
install=menu
large-memory
lba32
map=/boot/map
root='UUID=c6c1c4e3-4e42-4b7a-89e2-69140f21f5'
read-only
vga=normal
image=/boot/vmlinuz
label=Linux
initrd=/boot/initrd.img
image=/boot/vmlinuz.old
label=LinuxOld
initrd=/boot/initrd.img.old
optional
Saving file '/etc/lilo.conf'... Plain Text Tab Width: 8 Ln 9, Col 47 INS
```

- ¿Por qué se usa <la dirección completa del link hacia sda> en lugar de sólo /dev/sda, y cuál es el papel que el programa udev cumple en todo esto?  
Es porque los nombres como /dev/sda pueden cambiar al reiniciar mientras que el nombre de la dirección es persistente.  
udev es el sistema de gestión de dispositivos, sirve para crear y mantener esos enlaces simbólicos en /dev/disk/by-id/

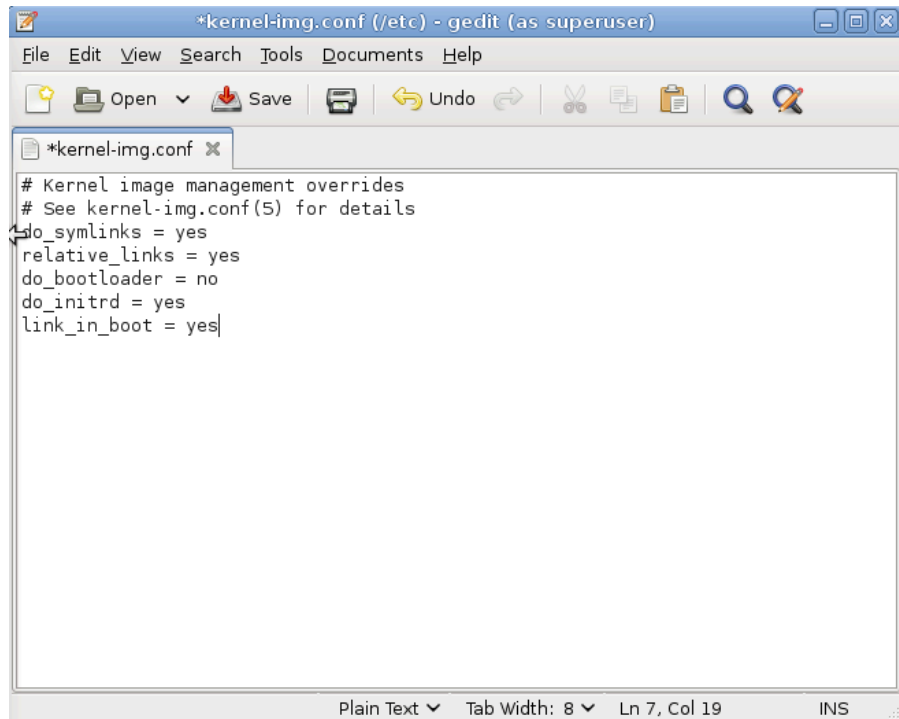
- ¿Qué es un *block device* y qué significado tiene *sdxN*, donde *x* es una letra y *N* es un número, en direcciones como */dev/sdb*? Investigue y explique los conceptos de *Master Boot Record* (MBR) y *Volume Boot Record* (VBR), y su relación con UEFI.
- Un block device es un tipo de dispositivo que maneja datos en bloques fijos como discos duros o particiones.
- En *sdxN*, *sd* significa un disco SCSI o SATA, *x* significa la letra que identifica el disco y *N* es el número que identifica en número de partición.

#### MBR y VBR:

- **Master Boot Record (MBR):**
  - Es un sector especial al inicio del disco (primeros 512 bytes) que contiene:
    - La tabla de particiones (hasta 4 particiones primarias).
    - Código de arranque inicial (bootloader como LILO o GRUB Legacy).
  - Usado en sistemas BIOS tradicionales.
- **Volume Boot Record (VBR):**
  - Es el sector de arranque de una partición específica.
  - Contiene código para cargar el sistema operativo de esa partición (ej. el cargador de Windows o Linux).
  - Usado por el MBR para pasar el control a la partición activa.
- **Relación con UEFI:**
  - UEFI reemplaza el MBR en sistemas modernos.
  - En lugar de un MBR, UEFI usa una **partición EFI System Partition (ESP)** que almacena cargadores de arranque (como GRUB o el firmware UEFI).
  - UEFI no depende de un VBR, ya que los cargadores están en archivos en la ESP (formato FAT).
  - MBR es limitado (2 TB, 4 particiones primarias); UEFI soporta discos grandes y más particiones con GPT.
- ¿Qué es hacer *chain loading*?
  - Chain loading es una técnica utilizada para que un cargador de arranque cargue otro cargador de arranque.
- ¿Qué se está indicando con la configuración `root="<el file system anotado>"`?
  - Es para especificar la partición raíz para cargar el kernel y el sistema operativo.

- f. Abra, en el mismo directorio `/etc`, el archivo `kernel-img.conf`, y asegúrese de que incluya las siguientes líneas (*i.e.*, modifique y agregue según sea necesario):

```
do_symlinks =  
yes  
relative_links =  
yes link_in_boot  
= yes
```



- g. Vaya al directorio `/boot` y elimine los *links* simbólicos llamados `vmlinuz` e `initrd.img`.

```
File Edit View Terminal Help
root@debian:/boot# ls
coffee.bmp          debian.bmp          initrd.img-2.6.32-5-686  sid.bmp            vmlinuz-2.6.32-5-686
config-2.6.32-5-686  debianlilo.bmp      initrd.img-2.6.39.4     System.map-2.6.32-5-686 vmlinuz-2.6.39.4
config-2.6.39.4      grub               sarge.bmp              System.map-2.6.39.4
root@debian:/boot# rm vmlinuz initrd.img
rm: cannot remove `vmlinuz': No such file or directory
rm: cannot remove `initrd.img': No such file or directory
```

- h. En el directorio `/boot` cree *links* simbólicos hacia `vmlinuz-<su versión de kernel>` e `initrd.img-<su versión de kernel>` con nombres `vmlinuz` e `initrd.img` respectivamente. Asegúrese del orden en el que se especifican los parámetros para crear un *link* simbólico (puede consultar `man ln`). En este paso debe reemplazar `<su versión de kernel>` por la versión de kernel que esté utilizando en su máquina virtual.

```
root@debian:/boot# ln -s vmlinuz-2.6.39.4 vmlinuz
root@debian:/boot# ls
coffee.bmp          debian.bmp          initrd.img-2.6.32-5-686  sid.bmp            vmlinuz
config-2.6.32-5-686  debianlilo.bmp      initrd.img-2.6.39.4     System.map-2.6.32-5-686 vmlinuz-2.6.32-5-686
config-2.6.39.4      grub               sarge.bmp              System.map-2.6.39.4   vmlinuz-2.6.39.4
root@debian:/boot# ln -s initrd.img-2.6.39.4 initrd.img
root@debian:/boot# ls
coffee.bmp          debian.bmp          initrd.img              sarge.bmp          System.map-2.6.39.4  vmlinuz-2.6.39.4
config-2.6.32-5-686  debianlilo.bmp      initrd.img-2.6.32-5-686  sid.bmp            vmlinuz
config-2.6.39.4      grub               initrd.img-2.6.39.4     System.map-2.6.32-5-686 vmlinuz-2.6.32-5-686
root@debian:/boot#
```

- ¿Qué es `vmlinuz`?

- i. En este mismo directorio elimine el subdirectorio `grub` con el siguiente comando:

```
sudo rm -r /boot/grub
```

```
File Edit View Terminal Help
root@debian:/boot# cd grub
bash: cd: grub: No such file or directory
root@debian:/boot#
```

- j. Vaya al directorio `/etc/kernel` y ejecute `ls`. Verá varios directorios. Acceda a cada uno y elimine los archivos que encuentre (si encuentra) que tengan “grub” en su nombre.

```
File Edit View Terminal Help
os@debian:/etc/kernel$ ls
header_postinst.d postinst.d postrm.d prerm.d
os@debian:/etc/kernel$
```

- k. Vaya al directorio `/etc/initramfs/post-update.d` y elimine los archivos que encuentre (si encuentra) que tengan “grub” en su nombre.

```
root@debian:/etc# cd initramfs/post-update.d/
root@debian:/etc/initramfs/post-update.d# ls
lilo
root@debian:/etc/initramfs/post-update.d#
```

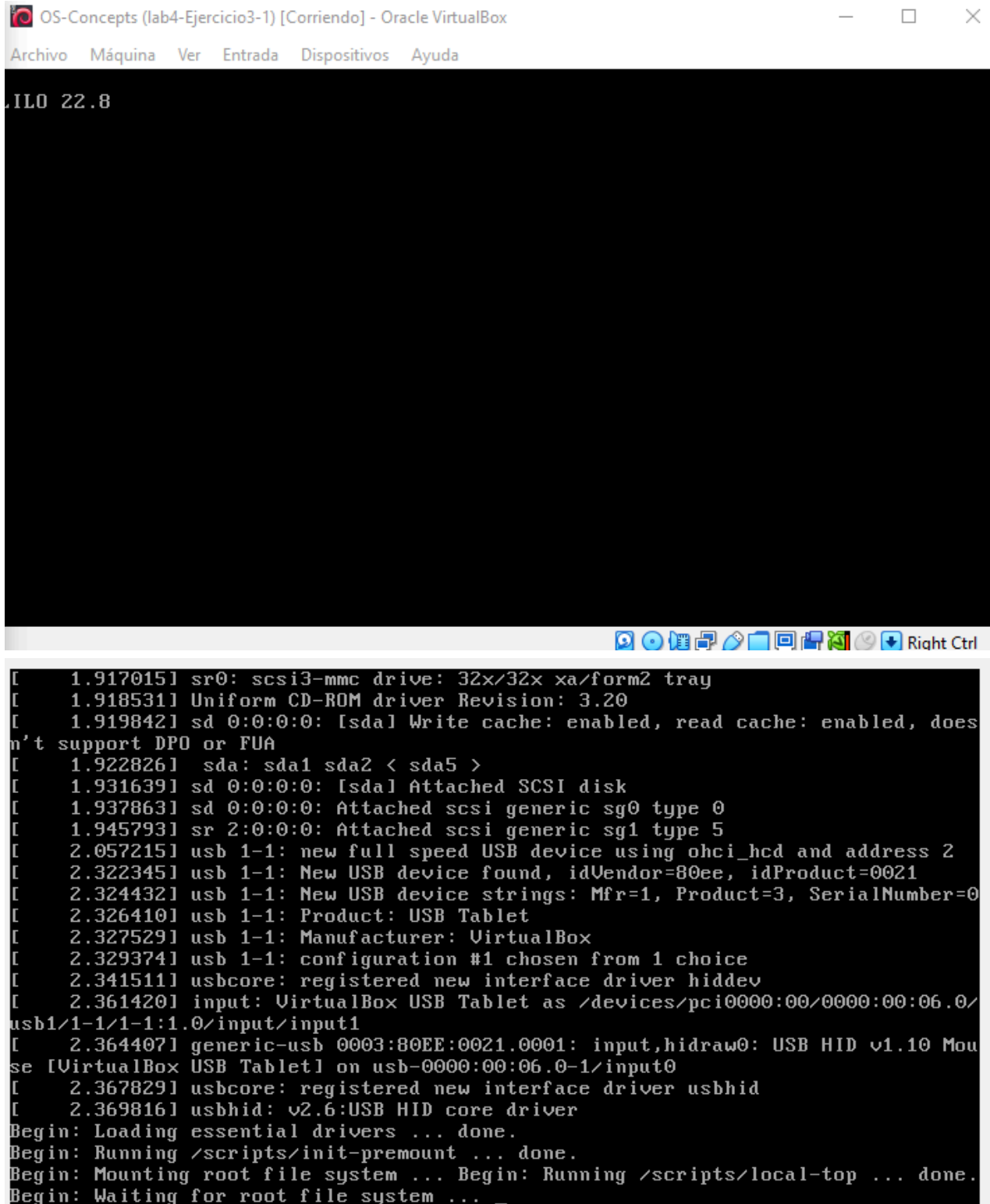
- l. Ejecute el siguiente comando:

```
sudo dpkg-reconfigure linux-image-<su versión de kernel>
```

- m. Si todo ha salido bien hasta ahora, reinicie su máquina virtual. Su sistema cargará el sistema operativo por medio de LILO en lugar de GRUB, y deberá iniciar sin pasar por el menú de selección de *kernel*. Cree una nueva *snapshot* de su máquina virtual y luego use esta y la *snapshot* anterior para tomar fotos del proceso de *booteo*, evidenciando el empleo de GRUB y LILO en cada caso. Incluya sus fotos o capturas con sus entregables.

```
root@debian:/etc# dpkg-reconfigure linux-image-2.6.32-5-686
Running depmod.
Running update-initramfs.
update-initramfs: Generating /boot/initrd.img-2.6.32-5-686
Examining /etc/kernel/postinst.d.
run-parts: executing /etc/kernel/postinst.d/dkms 2.6.32-5-686 /boot/vmlinuz-2.6.32-5-686
dkms: running auto installation service for kernel 2.6.32-5-686:
    virtualbox-ose-guest (3.2.10)...done.
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 2.6.32-5-686 /boot/vmlinuz-2.6.32-5-686
run-parts: executing /etc/kernel/postinst.d/pm-utils 2.6.32-5-686 /boot/vmlinuz-2.6.32-5-686
run-parts: executing /etc/kernel/postinst.d/update-notifier 2.6.32-5-686 /boot/vmlinuz-2.6.32-5-686
run-parts: executing /etc/kernel/postinst.d/zz-lilo 2.6.32-5-686 /boot/vmlinuz-2.6.32-5-686
Added Linux *
Skipping /boot/vmlinuz.old
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 2.6.32-5-686 /boot/vmlinuz-2.6.32-5-686
Generating grub.cfg ...
cat: /boot/grub/video.lst: No such file or directory
Found linux image: /boot/vmlinuz-2.6.39.4
Found initrd image: /boot/initrd.img-2.6.39.4
Found linux image: /boot/vmlinuz-2.6.32-5-686
Found initrd image: /boot/initrd.img-2.6.32-5-686
done
root@debian:/etc# █
```

## Reinicio con LILO:



```
OS-Concepts (lab4-Ejercicio3-1) [Corriendo] - Oracle VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda

LILO 22.8

[ 1.917015] sr0: scsi3-mmc drive: 32x/32x xa/form2 tray
[ 1.918531] Uniform CD-ROM driver Revision: 3.20
[ 1.919842] sd 0:0:0:0: [sda] Write cache: enabled, read cache: enabled, does
n't support DPO or FUA
[ 1.922826] sda: sda1 sda2 < sda5 >
[ 1.931639] sd 0:0:0:0: [sda] Attached SCSI disk
[ 1.937863] sd 0:0:0:0: Attached scsi generic sg0 type 0
[ 1.945793] sr 2:0:0:0: Attached scsi generic sg1 type 5
[ 2.057215] usb 1-1: new full speed USB device using ohci_hcd and address 2
[ 2.322345] usb 1-1: New USB device found, idVendor=80ee, idProduct=0021
[ 2.324432] usb 1-1: New USB device strings: Mfr=1, Product=3, SerialNumber=0
[ 2.326410] usb 1-1: Product: USB Tablet
[ 2.327529] usb 1-1: Manufacturer: VirtualBox
[ 2.329374] usb 1-1: configuration #1 chosen from 1 choice
[ 2.341511] usbcore: registered new interface driver hiddev
[ 2.361420] input: VirtualBox USB Tablet as /devices/pci0000:00/0000:00:06.0/
usb1/1-1/1-1:1.0/input/input1
[ 2.364407] generic-usb 0003:80EE:0021:0001: input,hidraw0: USB HID v1.10 Mou
se [VirtualBox USB Tablet] on usb-0000:00:06.0-1/input0
[ 2.367829] usbcore: registered new interface driver usbhid
[ 2.369816] usbhid: v2.6:USB HID core driver
Begin: Loading essential drivers ... done.
Begin: Running /scripts/init-premount ... done.
Begin: Mounting root file system ... Begin: Running /scripts/local-top ... done.
Begin: Waiting for root file system ... _
```

```
se [VirtualBox USB Tablet] on usb-0000:00:06.0-1/input0
[ 2.367829] usbcore: registered new interface driver usbhid
[ 2.369816] usbhid: v2.6:USB HID core driver
Begin: Loading essential drivers ... done.
Begin: Running /scripts/init-premount ... done.
Begin: Mounting root file system ... Begin: Running /scripts/local-top ... done.
Begin: Waiting for root file system ... done.
Gave up waiting for root device. Common problems:
- Boot args (cat /proc/cmdline)
- Check rootdelay= (did the system wait long enough?)
- Check root= (did the system wait for the right device?)
- Missing modules (cat /proc/modules; ls /dev)
ALERT! /dev/disk/by-uuid/c6c1c4e3-4e42-4b7a-89e2-69140f21f5 does not exist. Dr
opping to a shell!

BusyBox v1.17.1 (Debian 1:1.17.1-8) built-in shell (ash)
Enter 'help' for a list of built-in commands.

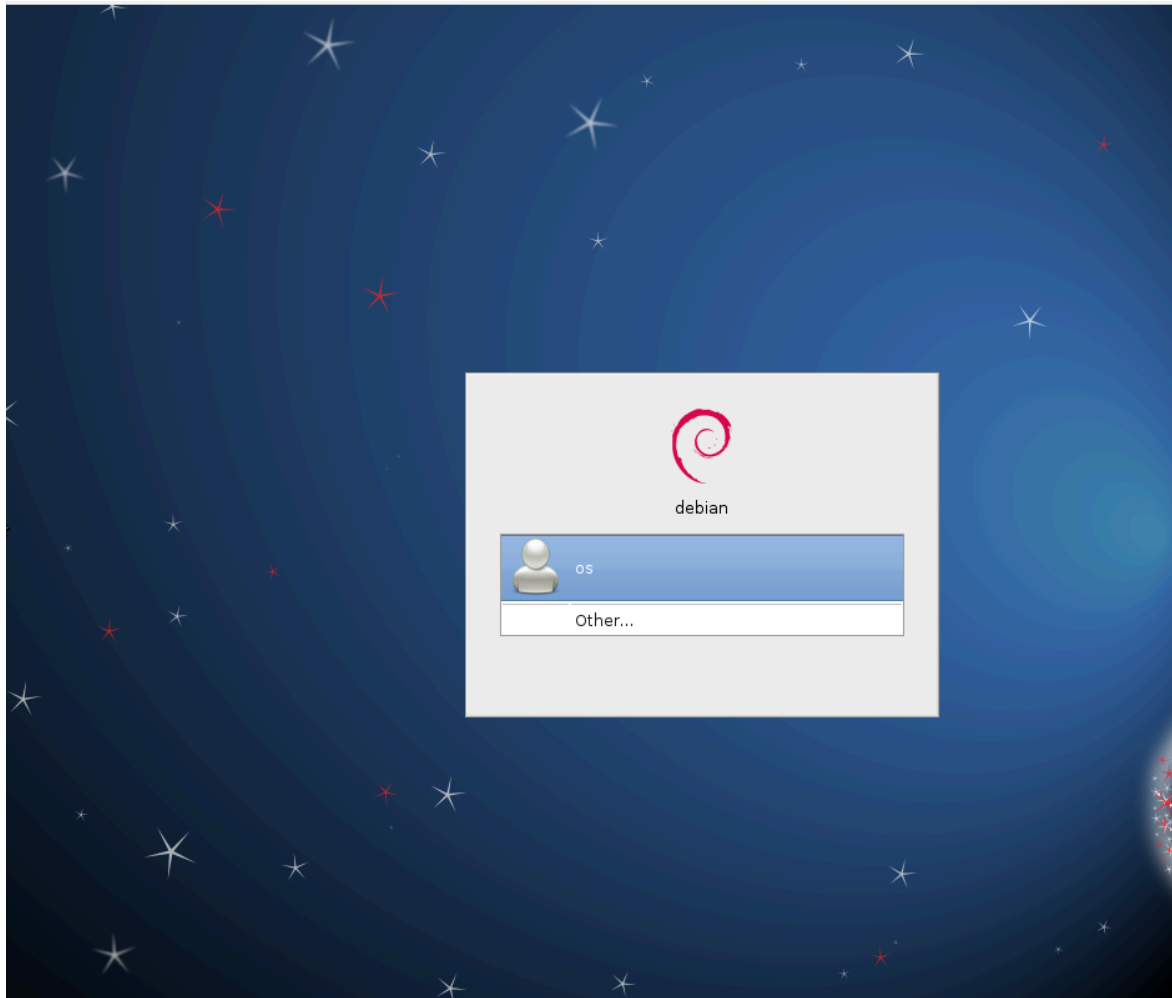
/bin/sh: can't access tty: job control turned off
(initramfs)
(initramfs) ls
bin      dev      init      proc      sbin      sys      var
conf     etc      lib      root      scripts   tmp
(initramfs)
```

## Reinicio con GRUB:

```
ata1 master: UBOX CD-ROM ATAPI-6 CD-ROM/DVD-ROM

iPXE (http://ipxe.org) E2:00.0 E2001.10 E200_
```





- Mencione tres diferencias funcionales entre GRUB y LILO.

1. GRUB ofrece un menú interactivo editable; LILO carga la entrada predeterminada sin interacción directa.
2. GRUB genera configuraciones automáticamente con update-grub; LILO requiere editar lilo.conf y ejecutar /sbin/lilo.
3. GRUB soporta más sistemas de archivos y UEFI; LILO está limitado a BIOS/MBR y sistemas de archivos básicos.