

---

Fecha de Entrega: 23 de marzo, 2022.

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. Debe entregar en Canvas un archivo de texto con sus respuestas a las preguntas planteadas y con las capturas de pantalla solicitadas.

Materiales: se recomienda la máquina virtual OSC-2016 para el ejercicio 2, aunque la instalación y remoción de un módulo por medio de un programa debería ser logable en versiones más recientes de Linux con instrucciones similares.

El ejercicio 3 requiere reemplazar el sistema de arranque GRUB por el sistema de arranque LILO. Las instrucciones garantizan esta meta si se trabaja sobre la máquina OSC-2016. De trabajarse en otro sabor o versión de Linux, el objetivo debe ser instalar LILO **manualmente**, por lo que debería dejarse registro de los pasos tomados, principalmente de las diferencias que haya con respecto a las instrucciones presentadas en este documento.

El ejercicio 1 puede desarrollarse en cualquier sistema Linux mientras se pueda instalar SystemTap.

#### 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
```

```
sudo stap profiler.stp
Proceso: swapper/0
ID del proceso: 0
Proceso: swapper/1
ID del proceso: 0
Proceso: swapper/2
ID del proceso: 0
Proceso: swapper/3
ID del proceso: 0
Proceso: swapper/2
ID del proceso: 0
Proceso: swapper/0
ID del proceso: 0
Proceso: swapper/3
ID del proceso: 0
Proceso: swapper/1
```

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.).

- ¿Qué puede ver en el *output* cuando realiza estas acciones?  
Lo que se puede observar es que al hacer cosas como abrir otros programas, archivos etc, mientras que esta el profiler.stp, se muestra la información sobre los procesos que están siendo ejecutados. Identificándolos con un nombre e Id
- ¿Para qué sirve SystemTap?  
Es una herramienta de diagnóstico y monitorio de Linux que puede recopilar información sobre el rendimiento y el comportamiento del sistema para solucionar problemas de inmediato.
- ¿Qué es una *probe*?  
Es una parte de código que se inserta en el Kernel de Linux para recopilar información específica.
- ¿Cómo funciona SystemTap?  
Funciona mediante la inserción de código en el kernel de Linux en tiempo de ejecución. Este código se ejecuta en el contexto del SO y tiene acceso a toda la información de manera detallada sobre el comportamiento del mismo e inclusive su rendimiento.
- ¿Qué es hacer *profiling* y qué tipo de *profiling* se hace en este ejercicio?  
Este es un proceso que se dedica a recopilar y analizar datos sobre el rendimiento y el comportamiento de un sistema o aplicación; todo ello con la meta de identificar áreas de mejora. Como vemos en este ejemplo que realizamos se está haciendo un proceso que ejecuta el SO en tiempo real, denominándose profiling dinámico.

## Ejercicio 2 (30 puntos)

- Abra su máquina virtual y tómela una *snapshot*.
- 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>`
- 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.

```
// Mariana David  
// Carnet: 201055  
  
#include <linux/init.h>  
#include <linux/module.h>  
#include <linux/kernel.h>  
#include <linux/list.h>  
  
int simple_init(void)  
{  
    printk(KERN_INFO "Loading Module\nHello World");  
    return 0;  
}  
  
void simple_exit(void)  
{  
    printk(KERN_INFO "Removing Module\nGoodbye World");  
}
```

- ¿Cuál es la diferencia en C entre un método que no recibe parámetros y uno que recibe `void`?  
La diferencia es que el método que no recibe parámetros no especifica ningún tipo de declaración, haciendo referencia a que el método puede aceptar cualquier numero y argumento; por otro lado el método que recibe el `void` especifica que no esera ningún argumente, es decir que si intenta llamar a la función será un error.
- 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`?

La diferencia radica en que `printk` es una función a nivel de kernel que es capaz de imprimir en diferentes niveles de registros definidos, mientras que el `printf` siempre imprime en un descriptor de archivos. `-std_out`.

- ¿Qué es y para qué sirve `KERN_INFO`?

Es un macro que se utiliza en kernel para indicar el nivel de severidad de un mensaje que esta siendo impreso. Ese se utiliza con el `printk` para imprimir mensajes en el registro del kernel. Prácticamente ayuda porque especifica la importancia del mensaje que quiere ser impreso.

- 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("<Descripcion>");
MODULE_AUTHOR("<Su nombre>");

// Mariana David
// Carnet: 201055

#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/list.h>

int simple_init(void)
{
    printk(KERN_INFO "Loading Module\nHello World!");
    return 0;
}

void simple_exit(void)
{
    printk(KERN_INFO "Removing Module\nGoodbye World!");
}

module_init(simple_init);
module_exit(simple_exit);
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Me gustan los animales");
MODULE_AUTHOR("MarianaDavid");
```

Grabe su programa.

- 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
```

```
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 definición de objeto en un *Makefile* especifica un objeto a construir con sus respectivas dependencias. En este caso el `obj-m` es una variable que especifica los archivos objeto que se deben construir como módulos del Kernel

- ¿Qué función tienen las líneas `all:` y `clean:`?  
La función del `all` es la acción predeterminada que se tomara cuando se ejecute el comando `make` sin argumento y en el caso del `clean` especifica la acción que se debe tomar para limpiar los artefactos de compilación.
- ¿Qué hace la opción `-C` en este *Makefile*?  
Lo que hace es que especifica el directorio donde `make` debe buscar el *Makefile* para realizar las acciones. En este caso, cambia el directorio del sistema y de compilación del Kernel para la versión del kernel que se está ejecutando actualmente.
- ¿Qué hace la opción `M` en este *Makefile*?  
Este especifica el directorio que contiene el código fuente del módulo del kernel externo. En este caso, está configurado en el directorio de trabajo actual permitiendo que *Makefile* pueda encontrar el archivo `simple.c` y así poder generar el archivo `simple.o` en el mismo directorio de trabajo.

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

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

```
sudo insmod simple.ko
[sudo] password for Mari:
```

```
[ 1490.777335] hid-generic 0003:80EE:0021.0002: input,hidraw0: USB HID v1.10 Mou
se [VirtualBox USB Tablet] on usb-0000:00:06.0-1/input0
[ 5889.266504] Loading Module
Hello World!
[ 6884.905083] Removing Module
Goodbye World!
```

Tome una captura de pantalla de los resultados de ambos comandos e inclúyala en sus entregables.

- ¿Para qué sirve `dmesg`?

`dmesg` es un comando utilizado en sistemas operativos Unix y Linux que permite al usuario ver los mensajes del kernel del sistema. Estos mensajes pueden proporcionar información importante sobre eventos que han ocurrido en el sistema, como errores de hardware, problemas de configuración de dispositivos, advertencias de seguridad y otros eventos importantes del sistema.

- ¿Qué hace la función `simple_init` en su programa `simple.c`?

Esta función es llamada cuando el módulo del kernel se descarga del kernel. En este ejercicio, se ve que imprime un mensaje en el registro del kernel utilizando la función `printk` para indicar que el modulo que tenemos esta siendo eliminado.

- i. Ahora ejecute los siguientes comandos:

```
sudo rmmod simple
dmesg
```

```
ACPI: Sleep Button [SLPF]
ACPI: Video Device [GFX0] (multi-head: yes rom: no post: no)
input: Video Bus as /devices/LNXSYSTM:00/LNXXSYB:00/0000:00:02:00/input/input6
snd_intel8x0 0000:00:02:0: disable (unknown or VT-d) VM optimization
Adding 392198k swap on /dev/sda5. Priority:-1 extents:1 across:392198k FS
alg: No test for crc32 (crc32-pclmul)
pmouse serio: alps: Unknown ALPS touchpad: E7=10 00 64, EC=10 00 64
input: IntelPS/2 Generic Explorer Mouse as /devices/platform/18042/serial/input/input7
EXT4-fs (sda1): re-mounted. Opts: errors=remount-ro
systemd-journald[107]: Received request to flush runtime journal from PID 1
snd_intel8x0 0000:00:02:0: white list rate for 1028:0177 is 48000
RPC: Registered named UNIX socket transport module.
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
RPC: Registered tcp NFSv4.1 backchannel transport module.
FS-Cache: Loaded
FS-Cache: Netfs 'nfs' registered for caching
Installing knfsd (copyright (C) 1996 okir@sonad.swb.de).
cfs00211: Calling CRDA to update world regulatory domain
cfs00211: World regulatory domain updated.
cfs00211: DFS Master region: unset
cfs00211: (start_freq - end_freq @ bandwidth), (max_antenna_gain, max_wirp), (dfs_cac_time)
cfs00211: (2402000 KHz - 2472000 KHz @ 40000 KHz), (N/A, 2000 mBm), (N/A)
cfs00211: (2457000 KHz - 2482000 KHz @ 40000 KHz), (N/A, 2000 mBm), (N/A)
cfs00211: (2474000 KHz - 2494000 KHz @ 20000 KHz), (N/A, 2000 mBm), (N/A)
cfs00211: (5170000 KHz - 5250000 KHz @ 80000 KHz, 160000 KHz AUTO), (N/A, 2000 mBm), (N/A)
cfs00211: (5250000 KHz - 5330000 KHz @ 80000 KHz, 160000 KHz AUTO), (N/A, 2000 mBm), (0 s)
cfs00211: (5490000 KHz - 5730000 KHz @ 160000 KHz), (N/A, 2000 mBm), (N/A)
cfs00211: (5735000 KHz - 5835000 KHz @ 80000 KHz), (N/A, 2000 mBm), (N/A)
cfs00211: (57240000 KHz - 63720000 KHz @ 2160000 KHz), (N/A, 0 mBm), (N/A)
floppy: no floppy controllers found
work still pending
ipvl: ADDRCONF(NETDEV_UP): eth0: link is not ready
e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX
Loading Module
valizado
Removing Module
```

Tome una nueva captura de pantalla de los resultados de ambos comandos e inclúyala en sus entregables.

- ¿Qué hace la función `simple_exit` en su programa `simple.c`?

Nos dice cuando se esta removiendo el módulo imprimiendo al buffer del kernel

- Usted ha logrado crear, cargar y descargar un módulo de Linux. ¿Qué poder otorga el ejecutar código de esta forma?

Nos permite integrar nuevas partes al kernel sin necesidad de compilar todo nuevamente.

### Ejercicio 3 (40 puntos)

- 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.**
- Ejecute el siguiente comando en una terminal (note el guion al final):

```
sudo apt-get --purge install lilo grub-legacy-
```

```
:-$ sudo apt-get --purge install lilo grub-legacy-
Get:1 http://ftp.us.debian.org/debian/ jessie/main lilo 1:24.1-1
Fetched 275 kB in 0s (464 kB/s)
Preconfiguring packages ...
Selecting previously unselected package lilo.
(Reading database ... 155058 files and directories currently installed)
Preparing to unpack .../lilo_1%3a24.1-1_1386.deb ...
Unpacking lilo (1:24.1-1) ...
Processing triggers for man-db (2.7.0.2-5) ...
Setting up lilo (1:24.1-1) ...
```

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”.

- c. Vaya al directorio `/dev/disk/by-id` y ejecute el comando `ls -Al`. El resultado le mostrará varios *links* simbólicos, 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`.

```
/dev/disk/by-id$ ls -Al
total 0
lrwxrwxrwx 1 root root 9 Mar 28 17:55 ata-VBOX_CD-ROM_VB2-01700376 -> .././sr0
lrwxrwxrwx 1 root root 9 Mar 28 17:55 ata-VBOX_HARDDISK_VB50deffe8-78c5b36b -> .././sda
lrwxrwxrwx 1 root root 10 Mar 28 17:55 ata-VBOX_HARDDISK_VB50deffe8-78c5b36b-part1 -> .././sda1
lrwxrwxrwx 1 root root 10 Mar 28 17:55 ata-VBOX_HARDDISK_VB50deffe8-78c5b36b-part2 -> .././sda2
lrwxrwxrwx 1 root root 10 Mar 28 17:55 ata-VBOX_HARDDISK_VB50deffe8-78c5b36b-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>`.

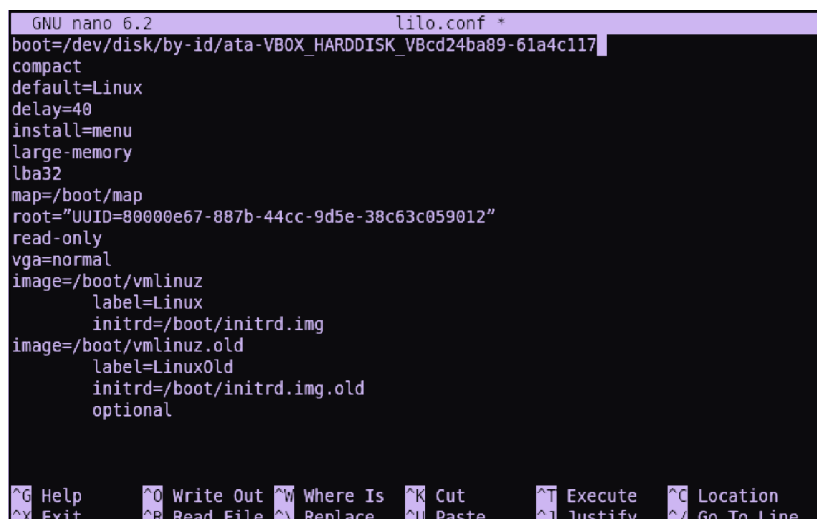
```
:/etc$ cat fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/sda1 during installation
UUID=5f2e2232-4e47-4fe8-ae94-45ea749a5c92 / ext4 errors=remount-ro 0 1
# swap was on /dev/sda5 during installation
UUID=ce96e707-6ab8-4556-aafe-5799e9a86292 none swap sw 0 0
/dev/sr0 /media/cdrom0 udf,iso9660 user,noauto 0 0
```

- ¿Qué es y para qué sirve el archivo `fstab`?  
Es una configuración de sistema que prácticamente genera un listado de todas las particiones disponibles del disco y otro tipo de file system y data sources e indica como inicializan.
- ¿Qué almacena el directorio `/etc`? ¿En Windows, quién (hasta cierto punto) funge como `/etc`?  
Es el directorio que contiene los archivos de configuración, que son los archivos locales que se utilizan para el control de operaciones de un programa.

- ¿Qué se almacena en `/dev` y en `/dev/disk`?  
Se almacenan los archivos especiales de dispositivos. Además da más información sobre las particiones en el sistema.
- 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=Linux
delay=40
install=menu
large-memory
lba32
map=/boot/map
root="<el file system anotado>"
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
```

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).



```
GNU nano 6.2          lilo.conf *
boot=/dev/disk/by-id/ata-VBOX_HARDDISK_VBcd24ba89-61a4c117
compact
default=Linux
delay=40
install=menu
large-memory
lba32
map=/boot/map
root="UUID=80000e67-887b-44cc-9d5e-38c63c059012"
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

^G Help  ^O Write Out  ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit  ^R Read File  ^N Replace   ^U Paste     ^I Justify   ^_ Go To Line
```



- ¿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?

Se utiliza la dirección completa del enlace hacia sda en lugar de dev/sda. Esto para asegurar el lugar correcto. Además el programa udev es responsable de administrar los archivos de dispositivos en /dev y crear enlaces simbólicos que apuntan a los dispositivos utilizando identificadores únicos, permitiendo referenciar dispositivos de manera mas correcta.

- ¿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.

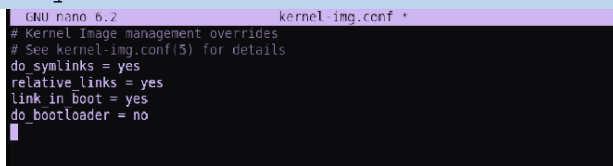
Es un tipo de dispositivo que almacena datos en bloques y permite el acceso aleatorio a estos bloques. Los discos duros y las unidades USB son ejemplos de este. Por otro lado el master boot record, es el primer sector de un dispositivos de almacenamiento y contiene información sobre como están organizadas las particiones en el dispositivo y código para iniciar el proceso de arranque. Además, contiene código para cargar el sistema operativo.

- ¿Qué es hacer *chain loading*?  
Es cargar un cargador de arranque desde otro cargador de arranque. Esto permite encadenar varios cargadores para seleccionar entre múltiples sistemas operativos o versiones del kernel.
- ¿Qué se está indicando con la configuración `root="<el file system anotado>"`?

Se indica a que partición del disco se quiere apuntar

- 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 raíz y elimine los *links* simbólicos llamados `vmlinuz` e `initrd.img`.

```
rm /vmlinuz
symbolic link '/vmlinuz'? y
rm /initrd.img
symbolic link '/initrd.img'? y
clear
```

- h. Vaya al directorio /boot y cree *links* simbólicos hacia `vmlinuz-3.16.0-4-686-pae` e `initrd.img-3.16.0-4-686-pae` 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`).

```
ln -s vmlinuz-3.16.0-4-686-pae vmlinuz
ls -Al
```

- ¿Qué es `vmlinuz`?

Es el kernel de Linux o bien el núcleo de sistema operativo Linux.

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

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

```
new search terminal loop
:/boot# sudo rm -r /boot/grub/
:/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.

```
:/boot# cd /etc/kernel
:/etc/kernel# ls
.d postrm.d
:/etc/kernel# cd postinst.d/
:/etc/kernel/postinst.d# ls
removal initramfs-tools zz-runlilo zz-update-grub
:/etc/kernel/postinst.d# rm zz-update-grub
rm regular file 'zz-update-grub'? y
:/etc/kernel/postinst.d# cd ../post
../post: No such file or directory
:/etc/kernel/postinst.d# ls
removal initramfs-tools zz-runlilo
:/etc/kernel/postinst.d# cd ..
:/etc/kernel# ls
.d postrm.d
:/etc/kernel# cd postrm.d/
:/etc/kernel/postrm.d# ls
s-tools zz-runlilo zz-update-grub
:/etc/kernel/postrm.d# rm zz-update-grub
rm regular file 'zz-update-grub'? y
:/etc/kernel/postrm.d# ls
s-tools zz-runlilo
:/etc/kernel/postrm.d# cd ../
:/etc/kernel# ls
.d postrm.d
:/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.
- l. Ejecute el siguiente comando:

```
sudo dpkg-reconfigure linux-image-3.16.0-4-686-pae
```

```
/etc# sudo dpkg-reconfigure linux-image-3.16.0-4-686-pae
/etc/kernel/postinst.d/initramfs-tools:
update-initramfs: Generating /boot/initrd.img-3.16.0-4-686-pae
/etc/kernel/postinst.d/zz-runlilo:
Warning: /etc/lilo.conf should be owned by root
Added Linux *
Skipping /boot/vmlinuz.old
One warning was issued.
```

- 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.

```
GNU GRUB  version 2.06

*Start Linux Mint 21.1 Cinnamon 64-bit
Start Linux Mint 21.1 Cinnamon 64-bit (compatibility mode)
OEM install (for manufacturers)
Test memory

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.
```

```
LILO 24.2 Loading Linux.....
BIOS data check successful
```

- Mencione tres diferencias funcionales entre GRUB y LILO.
  1. Lilo no tiene interfaz de comando interactiva.
  2. Lilo almacena información sobre la localización del kernel u otro SO debe ser cargado en el MBR.
  3. Lilo no puede leerlas particiones
  4. El soporte de arranque desde una red: GRUB admite el arranque desde una red, mientras que Lilo no.