

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
```

```
cristian@cristian-VirtualBox:~$ sudo apt-get install systemtap
[sudo] contraseña para cristian:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  systemtap-common systemtap-runtime
Paquetes sugeridos:
  systemtap-doc vim-addon-manager
Se instalarán los siguientes paquetes NUEVOS:
  systemtap systemtap-common systemtap-runtime
0 actualizados, 3 nuevos se instalarán, 0 para eliminar y 250 no actualizados.
Se necesita descargar 2,061 kB de archivos.
Se utilizarán 9,971 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] S
Des:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 systemtap-runtime am
d64 4.6-2 [89.1 kB]
Des:2 http://archive.ubuntu.com/ubuntu jammy/universe amd64 systemtap-common amd
64 4.6-2 [691 kB]
Des:3 http://archive.ubuntu.com/ubuntu jammy/universe amd64 systemtap amd64 4.6-
2 [1,281 kB]
```

- 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()) }
```

```
≡ profiler.stp ×
≡ profiler.stp
1  probe timer.profile {
2      printf("Proceso: %s\n", execname())
3      printf("ID del proceso: %d\n", pid())
4  }
```

- c. Ejecute su archivo usando el siguiente comando:

```
sudo stap profiler.stp
```

```
Proceso: at-spi2-registr
ID del proceso: 1247
Proceso: stapio
ID del proceso: 3723
Proceso: stapio
ID del proceso: 3723
Proceso: WRScene~derLP#1
ID del proceso: 1764
Proceso: cinnamon
ID del proceso: 1400
Proceso: stapio
ID del proceso: 3723
Proceso: csd-keyboard
ID del proceso: 1259
Proceso: kworker/u10:2
ID del proceso: 3733
Proceso: code
ID del proceso: 2665
Proceso: swapper/0
ID del proceso: 0
Proceso: swapper/3
ID del proceso: 0
Proceso: WRWorkerLP#0
ID del proceso: 1764
Proceso: gnome-terminal-
ID del proceso: 1730
Proceso: swapper/0
ID del proceso: 0
Proceso: glean.dispatche
ID del proceso: 1764
```

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?** Primero que nada, podemos ver el proceso que se está ejecutando y su ID, al momento de realizar diferentes acciones en el sistema mientras este comando se está ejecutando podemos ver algunos procesos como:
 - stapio es el proceso de SystemTap
 - cinnamon es el entorno de escritorio de Linux Mint
 - swapper/0 es el proceso del kernel que maneja la paginación de la memoria
 - glean.disptache es un proceso relacionado con el controlador de gráficos
 - gnome-terminal es la terminal que estás usando para ejecutar el profiler.stp

Me pareció interesante el proceso swapper ya que es un proceso del sistema operativo que se encarga de manejar la memoria virtual. Su función principal es la de intercambiar páginas de memoria RAM entre la memoria física y el disco duro, cuando hay escasez de memoria física disponible.

- **¿Para qué sirve SystemTap?** Básicamente es una herramienta de diagnóstico con la cual podemos analizar el sistema operativo en tiempo real. Es decir que nos puede servir para recopilar datos del sistema operativo en tiempo de ejecución sin interrupciones, lo que significa que no es necesario detener o reiniciar el sistema para realizar un diagnóstico o análisis. SystemTap se utiliza comúnmente para depurar problemas de rendimiento, seguimiento de errores, monitoreo de eventos del sistema, seguimiento de llamadas al sistema, entre otros.
- **¿Qué es una *probe*?** Es una instrucción que se inserta en el código fuente del programa o en el kernel del sistema operativo para obtener información sobre su comportamiento en tiempo de ejecución. Una probe puede ser vista como una especie de "punto de control" en el programa o kernel que permite a SystemTap recopilar datos y estadísticas sobre el funcionamiento del sistema, como la cantidad de veces que se llama a una función, el tiempo que tarda en ejecutarse una sección de código, entre otras cosas.
- **¿Cómo funciona SystemTap?**
SystemTap funciona inyectando código en tiempo de ejecución en el kernel del sistema operativo y proporcionando una forma de instrumentar y medir el rendimiento del sistema en tiempo real. Para hacer esto, SystemTap utiliza un lenguaje de programación de script llamado SystemTap Script Language (SSTL), que es similar a C.

SystemTap utiliza un enfoque de "sondeo dinámico" que permite a los desarrolladores agregar y quitar instrumentación en tiempo real sin tener que reiniciar el sistema. Las sondas de SystemTap, también conocidas como "probes", son puntos en el código del kernel que los desarrolladores pueden usar para insertar código adicional para la instrumentación. SystemTap proporciona un conjunto de herramientas y utilidades que permiten a los desarrolladores crear y depurar scripts de SystemTap de manera eficiente.
- **¿Qué es hacer *profiling* y qué tipo de *profiling* se hace en este ejercicio?** El *profiling* es el proceso de medir el rendimiento y el comportamiento de un sistema o una aplicación. Se realiza para identificar cuellos de botella, ineficiencias o problemas que puedan estar afectando el rendimiento.

En este ejercicio, se realiza profiling para identificar qué procesos están en ejecución y su ID correspondiente. La `probe timer.profile` es utilizada para medir el tiempo que tarda cada proceso en ejecutarse y mostrar su nombre y su ID en la salida. Este tipo de profiling se enfoca en la identificación de procesos en ejecución y su información relacionada, lo que puede ser útil para analizar el rendimiento y la carga del sistema en un momento dado.

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.
 - ¿Cuál es la diferencia en C entre un método que no recibe parámetros y uno que recibe `void`?
- 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`? son dos funciones de salida (o `print`) en el kernel de Linux. `printk` es una función de salida que se utiliza específicamente para la depuración en el kernel de Linux, mientras que `printf` es una función de la biblioteca estándar de C que se utiliza para imprimir en la consola en el espacio de usuario.
- ¿Qué es y para qué sirve `KERN_INFO`? es una macro utilizada en el kernel de Linux para especificar el nivel de severidad de un mensaje de registro. Se utiliza para definir el nivel de severidad como informativo. En otras palabras, es una forma de indicar que la información registrada es simplemente informativa y no indica un problema crítico. Se

puede utilizar para registrar información útil o para ayudar en la depuración de problemas.

- 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>");
```

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
```

- ¿Qué es una goal definition o definición de meta en un Makefile, y qué se está haciendo con la definición de meta obj-m? En un Makefile, una goal definition o definición de meta es una regla que indica al sistema de construcción qué objetivo (o "meta") debe construirse y cómo. En este caso, la definición de meta obj-m indica que el objetivo es construir el módulo del kernel "simple.o" a través del comando "make".
- ¿Qué función tienen las líneas all: y clean:? Las líneas "all:" y "clean:" son reglas que se utilizan para compilar y limpiar el proyecto, respectivamente. La primera regla, "all:", indica que cuando se ejecute el comando "make" sin argumentos, se debe construir el objetivo "simple.o" especificado en la definición de meta obj-m. La segunda regla, "clean:", indica que cuando se ejecute el comando "make clean", se debe limpiar el directorio de construcción de los archivos generados por la compilación.
- ¿Qué hace la opción -C en este Makefile? La opción "-C" en este Makefile indica el directorio en el que se deben buscar los archivos de origen y el archivo Makefile para compilar el proyecto. En este caso, se especifica que el directorio de construcción debe ser "/lib/modules/\$(shell uname -r)/build", lo que significa que se utilizarán los archivos de origen y el archivo Makefile en ese directorio para compilar el proyecto.
- ¿Qué hace la opción M en este Makefile? La opción "M" en este Makefile indica el directorio en el que se encuentran los archivos de origen del módulo del kernel que se está compilando. En este caso, se especifica que los archivos de origen se encuentran en el directorio actual ("\$(shell pwd)").

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

```
cristian@cristian-VirtualBox:~/Escritorio/SistemasOperativos/Lab4$ make
make -C /lib/modules/5.15.0-56-generic/build M=/home/cristian/Escritorio/Sistema
sOperativos/Lab4 modules
make[1]: se entra en el directorio '/usr/src/linux-headers-5.15.0-56-generic'
CC [M] /home/cristian/Escritorio/SistemasOperativos/Lab4/simple.o
MODPOST /home/cristian/Escritorio/SistemasOperativos/Lab4/Module.symvers
CC [M] /home/cristian/Escritorio/SistemasOperativos/Lab4/simple.mod.o
LD [M] /home/cristian/Escritorio/SistemasOperativos/Lab4/simple.ko
BTF [M] /home/cristian/Escritorio/SistemasOperativos/Lab4/simple.ko
Skipping BTF generation for /home/cristian/Escritorio/SistemasOperativos/Lab4/si
mple.ko due to unavailability of vmlinux
make[1]: se sale del directorio '/usr/src/linux-headers-5.15.0-56-generic'
```

- h. Ejecute los siguientes comandos:

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

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

```
cristian@cristian-VirtualBox:~/Escritorio/SistemasOperativos/Lab4$ sudo insmod s
imple.ko dmesg
[sudo] contraseña para cristian:
cristian@cristian-VirtualBox:~/Escritorio/SistemasOperativos/Lab4$
```

- ¿Para qué sirve `dmesg`? El comando "`dmesg`" muestra los mensajes del kernel que se han generado desde el inicio del sistema, incluyendo información sobre hardware, drivers y otros eventos importantes.
 - ¿Qué hace la función `simple_init` en su programa `simple.c`? La función `simple_init` es una función de inicialización del módulo. En este caso se utiliza para imprimir un mensaje de carga del módulo en el registro del kernel a través de la función `printk`.
- i. Ahora ejecute los siguientes comandos:

```
sudo rmmod simple
```

```
cristian@cristian-VirtualBox:~/Escritorio/SistemasOperativos/Lab4$ sudo rmmod si
mple
```

```
dmesg
```

```
tion="profile_load" profile="unconfined" name="man_groff" pid=610 comm="apparmor_parser"
[ 8.898567] audit: type=1400 audit(1681360656.772:8): apparmor="STATUS" operation="profile_load" profile="unconfined" name="libreoffice-oosplash" pid=617 comm="apparmor_parser"
[ 8.898623] audit: type=1400 audit(1681360656.776:9): apparmor="STATUS" operation="profile_load" profile="unconfined" name="/usr/bin/redshift" pid=615 comm="apparmor_parser"
[ 8.899782] audit: type=1400 audit(1681360656.776:10): apparmor="STATUS" operation="profile_load" profile="unconfined" name="tcpdump" pid=616 comm="apparmor_parser"
[ 8.908385] audit: type=1400 audit(1681360656.784:11): apparmor="STATUS" operation="profile_load" profile="unconfined" name="libreoffice-senddoc" pid=618 comm="apparmor_parser"
[ 9.885357] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX
[ 9.888098] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link becomes ready
[ 2030.630362] simple: module verification failed: signature and/or required key missing - tainting kernel
[ 2030.631856] simple: unknown parameter 'dmesg' ignored
[ 2030.631908] Loading module
Que onda
```

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`? La función `simple_exit` es una función de descarga del módulo. En este caso se utiliza para imprimir un mensaje de descarga del módulo en el registro del kernel a través de la función `printk`.
- Usted ha logrado crear, cargar y descargar un módulo de Linux. ¿Qué poder otorga el ejecutar código de esta forma? Al ejecutar código de esta forma se obtiene el poder de modificar y extender el comportamiento del kernel de Linux, permitiendo agregar nuevas funcionalidades y dispositivos a través de módulos personalizados.

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

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



```
cristian@cristian-VirtualBox:~$ sudo apt-get --purge install lilo grub-legacy-
[sudo] contraseña para cristian:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
El paquete «grub-legacy» no está instalado, no se eliminará
Se instalarán los siguientes paquetes NUEVOS:
  lilo
0 actualizados, 1 nuevos se instalarán, 0 para eliminar y 255 no actualizados.
Se necesita descargar 263 kB de archivos.
Se utilizarán 715 kB de espacio de disco adicional después de esta operación.
Des:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 lilo amd64 1:24.2-5.
1 [263 kB]
Descargados 263 kB en 1s (205 kB/s)
Preconfigurando paquetes ...
Seleccionando el paquete lilo previamente no seleccionado.
(Leyendo la base de datos ... 541968 ficheros o directorios instalados actualmen
te.)
Preparando para desempaquetar .../lilo_1%3a24.2-5.1_amd64.deb ...
Desempaquetando lilo (1:24.2-5.1) ...
Configurando lilo (1:24.2-5.1) ...
Procesando disparadores para man-db (2.10.2-1) ...
cristian@cristian-VirtualBox:~$
```

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`.

```
cristian@cristian-VirtualBox:~$ cd /dev/disk/by-id
cristian@cristian-VirtualBox:/dev/disk/by-id$ ls -Al
total 0
lrwxrwxrwx 1 root root 9 abr 12 22:37 ata-VBOX_CD-ROM_VB2-01700376 -> ../../sr0
lrwxrwxrwx 1 root root 9 abr 12 22:37 ata-VBOX_HARDDISK_VB5f363feb-af4300f6 ->
../../sda
lrwxrwxrwx 1 root root 10 abr 12 22:37 ata-VBOX_HARDDISK_VB5f363feb-af4300f6-par
t1 -> ../../sda1
lrwxrwxrwx 1 root root 10 abr 12 22:37 ata-VBOX_HARDDISK_VB5f363feb-af4300f6-par
t2 -> ../../sda2
lrwxrwxrwx 1 root root 10 abr 12 22:37 ata-VBOX_HARDDISK_VB5f363feb-af4300f6-par
t3 -> ../../sda3
cristian@cristian-VirtualBox:/dev/disk/by-id$
```

- 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 sy`

```
# /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/sda3 during installation
UUID=82b432d7-5223-4e60-b3be-d929498e996d / ext4 errors=remoun
# /boot/efi was on /dev/sda2 during installation
UUID=A77D-49ED /boot/efi vfat umask=0077 0 1
/swapfile none swap sw
```

- ¿Qué es y para qué sirve el archivo `fstab`? El archivo `fstab` es un archivo de configuración utilizado por el sistema operativo Linux para definir cómo y dónde se deben montar los sistemas de archivos en el sistema durante el proceso de arranque. Contiene información sobre los dispositivos de almacenamiento y las particiones, así como sobre la forma en que se deben montar en el sistema de archivos.
 - ¿Qué almacena el directorio `/etc`? ¿En Windows, quién (hasta cierto punto) funge como `/etc`? El directorio `/etc` es un directorio de configuración en sistemas Unix y Linux que contiene archivos de configuración para el sistema y las aplicaciones. Algunos ejemplos de archivos almacenados en `/etc` son `fstab`, `passwd`, `group`, `hosts`, `resolv.conf`, entre otros. En Windows, el directorio equivalente a `/etc` es el registro del sistema.
 - ¿Qué se almacena en `/dev` y en `/dev/disk`? El directorio `/dev` contiene los dispositivos de hardware y virtual de un sistema Linux. Los dispositivos se representan como archivos en este directorio, y se pueden acceder como si fueran archivos normales. El directorio `/dev/disk` contiene los dispositivos de almacenamiento como discos duros, unidades de CD/DVD, tarjetas SD, entre otros.
- 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).

```
cristian@cristian-VirtualBox: /etc
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
GNU nano 6.2 lilo.conf
boot=ata-VBOX_HARDDISK_VB5f363feb-af4300f6
compact
default=Linux
delay=40
install=menu
large-memory
lba32
map=/boot/map
root="UUID=82b432d7-5223-4e60-b3be-d929498e996d"
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
```

¿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? El uso de la dirección completa del link hacia sda, en lugar de /dev/sda, se debe a que el archivo lilo.conf se procesa en el arranque del sistema antes de que el sistema de archivos raíz esté montado. Por lo tanto, en ese momento, los dispositivos pueden no estar accesibles mediante /dev/sda y solo se pueden acceder a través de los enlaces simbólicos en /dev/disk/by-id.

El programa udev es responsable de gestionar los dispositivos en sistemas Linux. Cuando se conecta un dispositivo, udev detecta y asigna un nombre de dispositivo único y persistente a ese dispositivo, lo que permite que los programas del sistema identifiquen y utilicen el dispositivo de manera consistente.

¿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 dispositivo de almacenamiento de bloques, que es capaz de leer y escribir bloques de datos. El formato de dirección sdxN se utiliza para representar dispositivos de almacenamiento. La letra x representa una unidad física, mientras que N representa una partición lógica en la unidad física.

El Master Boot Record (MBR) es una sección especial del disco que se utiliza para arrancar sistemas operativos antiguos basados en BIOS. El Volume Boot Record (VBR) es un registro de arranque que se encuentra en el sector de arranque del sistema de archivos de una partición. UEFI es el reemplazo de BIOS y se utiliza en sistemas más modernos. En lugar de MBR, UEFI utiliza GPT (GUID Partition Table) para particionar los discos.

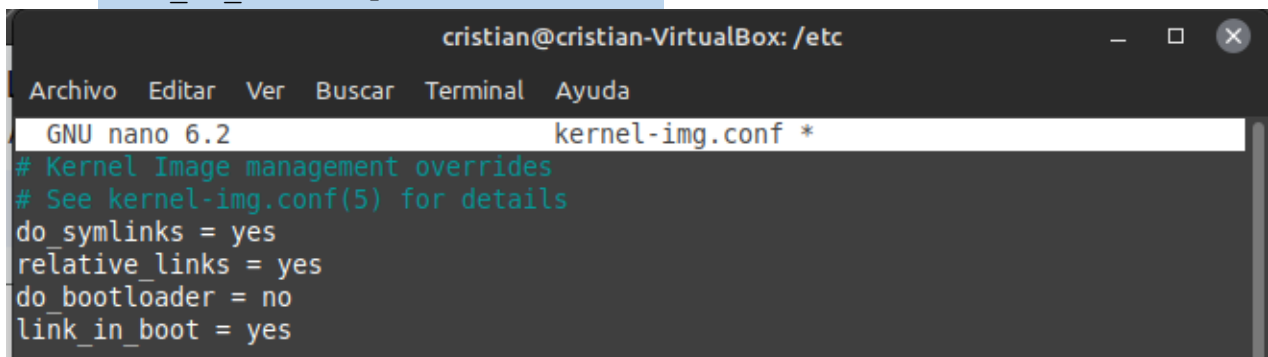
¿Qué es hacer *chain loading*?

El chain loading es un método de arranque en el que un bootloader carga otro bootloader. Por ejemplo, el bootloader del sistema operativo primario puede cargar el bootloader de otro sistema operativo instalado en una partición diferente.

¿Qué se está indicando con la configuración `root=<el file system anotado>`? La configuración `root=<el file system anotado>` indica la ruta del sistema de archivos raíz que el bootloader debe usar para cargar el sistema operativo. En el archivo `lilo.conf`, la línea `root` especifica el sistema de archivos raíz que corresponde al dispositivo que se va a utilizar para el arranque.

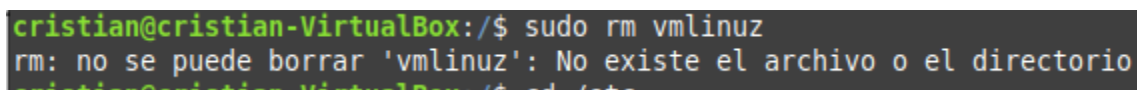
- f. Abra, en el mismo directorio `/etc`, el archivo `kernel-img.conf`, y asegúrese de que incluya las siguientes líneas (*i.e.*, modifique y agregue según sea necesario):

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



```
cristian@cristian-VirtualBox: /etc
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
GNU nano 6.2 kernel-img.conf *
# Kernel Image management overrides
# See kernel-img.conf(5) for details
do_symlinks = yes
relative_links = yes
do_bootloader = no
link_in_boot = yes
```

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



```
cristian@cristian-VirtualBox:/$ sudo rm vmlinuz
rm: no se puede borrar 'vmlinuz': No existe el archivo o el directorio
```

- 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`).

```
cristian@cristian-VirtualBox:/boot$ ls
coffee.bmp          initrd.img-5.15.0-56-generic
config-5.15.0-56-generic  initrd.img.old
debian.bmp          inside.bmp
debian-de.bmp       onlyblue.bmp
debianlilo.bmp      System.map-5.15.0-56-generic
efi                 tuxlogo.bmp
grub                vmlinuz
initrd.img          vmlinuz-5.15.0-56-generic
```

¿Qué es vmlinuz? **vmlinuz** es el nombre que se le da al kernel de Linux compilado en sistemas basados en el kernel de Linux. Es un archivo ejecutable que contiene el código del kernel, los controladores de dispositivos y otros componentes necesarios para que el sistema operativo funcione correctamente. El archivo **vmlinuz** es esencial para el arranque del sistema y es cargado en la memoria RAM durante el proceso de inicio del sistema. Después de que se carga el kernel, se carga el sistema de archivos y se inicializan los controladores de dispositivos para permitir que el sistema operativo pueda interactuar con el hardware del sistema.

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

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

```
cristian@cristian-VirtualBox:/boot$ sudo rm -r /boot/grub
```

- 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.

```
cristian@cristian-VirtualBox:/boot$ cd /etc/kernel
cristian@cristian-VirtualBox:/etc/kernel$ ls
header_postinst.d  install.d  postinst.d  postrm.d  preinst.d  prerm.d
cristian@cristian-VirtualBox:/etc/kernel$ cd header_postinst.d/
cristian@cristian-VirtualBox:/etc/kernel/header_postinst.d$ ls
dkms
cristian@cristian-VirtualBox:/etc/kernel/header_postinst.d$ cd ..
cristian@cristian-VirtualBox:/etc/kernel$ cd install.d/
cristian@cristian-VirtualBox:/etc/kernel/install.d$ ls
dkms
cristian@cristian-VirtualBox:/etc/kernel/install.d$ cd /etc/kernel/postinst.d
cristian@cristian-VirtualBox:/etc/kernel/postinst.d$ ls
dkms          pm-utils      zz-runlilo    zz-update-grub
initramfs-tools  xx-update-initrd-links  zz-shim
cristian@cristian-VirtualBox:/etc/kernel/postinst.d$ sudo rm -r zz-update-grub
cristian@cristian-VirtualBox:/etc/kernel/postinst.d$ cd /etc/kernel/postrm.d/
cristian@cristian-VirtualBox:/etc/kernel/postrm.d$ ls
initramfs-tools  zz-runlilo    zz-update-grub
cristian@cristian-VirtualBox:/etc/kernel/postrm.d$ sudo rm -r zz-update-grub
cristian@cristian-VirtualBox:/etc/kernel/postrm.d$ cd /etc/kernel/preinst.d/
cristian@cristian-VirtualBox:/etc/kernel/preinst.d$ ls
intel-microcode
cristian@cristian-VirtualBox:/etc/kernel/preinst.d$ cd /etc/kernel/prerm.d/
cristian@cristian-VirtualBox:/etc/kernel/prerm.d$ ls
dkms
cristian@cristian-VirtualBox:/etc/kernel/prerm.d$
```

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

```
cristian@cristian-VirtualBox:/etc/kernel/prerm.d$ cd /etc/initramfs/post-update.d
cristian@cristian-VirtualBox:/etc/initramfs/post-update.d$ ls
runlilo
cristian@cristian-VirtualBox:/etc/initramfs/post-update.d$
```

- l. Ejecute el siguiente comando:

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

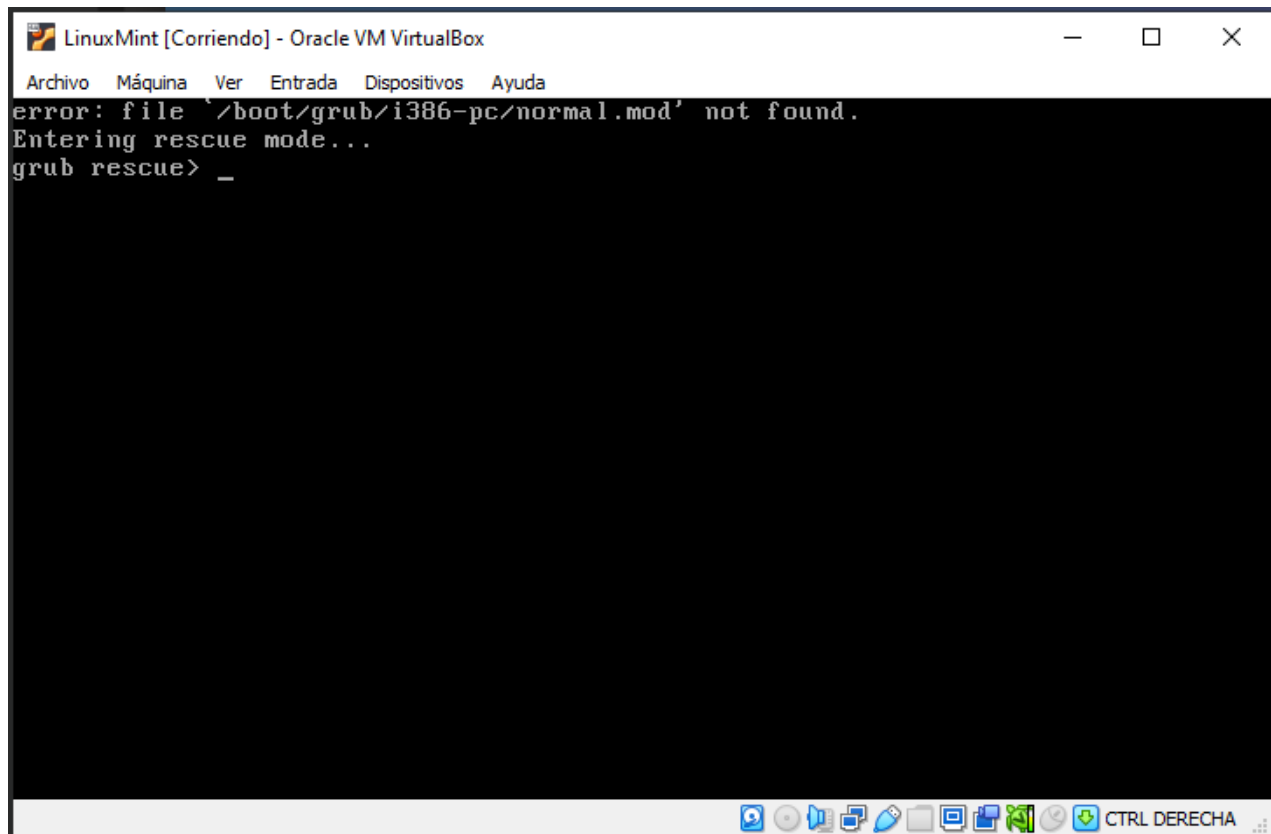
Primero debo encontrar el nombre correcto del paquete de imagen del kernel en mi sistema con el comando:

```
cristian@cristian-VirtualBox:/etc/initramfs/post-update.d$ dpkg -l | grep linux-image
ii  linux-image-5.15.0-56-generic  5.15.0-56.62      amd64
    Signed kernel image generic
ii  linux-image-generic            5.15.0.56.54      amd64
    Generic Linux kernel image
```

Luego ejecuto el comando `dpkg-reconfigure` con el nombre completo del paquete de imagen de mi kernel

```
cristian@cristian-VirtualBox:/etc/initramfs/post-update.d$ sudo dpkg-reconfigure linux-image-5.15.0-56-generic
/etc/kernel-img.conf:4: W: ignoring unknown parameter relative_links
I: /boot/vmlinuz.old is now a symlink to vmlinuz-5.15.0-56-generic
Procesando disparadores para linux-image-5.15.0-56-generic (5.15.0-56.62) ...
/etc/kernel/postinst.d/dkms:
 * dkms: running auto installation service for kernel 5.15.0-56-generic
...done.
/etc/kernel/postinst.d/initramfs-tools:
update-initramfs: Generating /boot/initrd.img-5.15.0-56-generic
/etc/kernel/postinst.d/zz-runlilo:
Fatal: raid_setup: stat("ata-VBOX_HARDDISK_VB5f363feb-af4300f6")
run-parts: /etc/kernel/postinst.d/zz-runlilo exited with return code 1
dpkg: error al procesar el paquete linux-image-5.15.0-56-generic (--configure):
 el subprocesso instalado paquete linux-image-5.15.0-56-generic script post-installation devolvió e
l código de salida de error 1
Se encontraron errores al procesar:
 linux-image-5.15.0-56-generic
cristian@cristian-VirtualBox:/etc/initramfs/post-update.d$
```

- 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.



- Mencione tres diferencias funcionales entre GRUB y LILO.
1. Capacidad de detección de sistemas operativos: GRUB es capaz de detectar automáticamente otros sistemas operativos instalados en el mismo disco, lo que permite al usuario seleccionar entre diferentes sistemas operativos para arrancar. LILO no tiene esta capacidad de detección automática, por lo que el usuario debe agregar manualmente la información del sistema operativo en el archivo de configuración.
 2. Soporte de sistema de archivos: GRUB es compatible con una amplia variedad de sistemas de archivos, incluyendo ext2, ext3, ext4, NTFS, FAT y más. LILO, por otro lado, solo es compatible con sistemas de archivos Linux como ext2 y ext3.
 3. Ubicación de la configuración: En GRUB, la configuración del gestor de arranque se almacena en un archivo de texto plano llamado grub.cfg. En LILO, la configuración se guarda en el archivo /etc/lilo.conf. Además, LILO requiere que se ejecute el comando "lilo" después de realizar cambios en la configuración, mientras que en GRUB los cambios se aplican automáticamente.