Universidad del Valle de Guatemala
Sistemas Operativos
Docentes: Erick Pineda; Tomás Gálvez P.
Semestre I, 2021

Laboratorio #4

Fecha de Entrega: 28 de marzo, 2021.

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 lograble 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
```

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?
- ¿Para qué sirve SystemTap?

- ¿Qué es una *probe*?
- ¿Cómo funciona SystemTap?
- ¿Qué es hacer *profiling* y qué tipo de *profiling* se hace en este ejercicio?

Ejercicio 2 (30 puntos)

a. Abra su máquina virtual y tómele 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`?
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`?
- ¿Qué es y para qué sirve `KERN_INFO`?

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`?
- ¿Qué función tienen las líneas `all:` y `clean:`?
- ¿Qué hace la opción `-C` en este *Makefile*?
- ¿Qué hace la opción `M` en este *Makefile*?

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
```

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

- ¿Para qué sirve `dmesg`?
- ¿Qué hace la función `simple_init` en su programa `simple.c`?

i. Ahora ejecute los siguientes comandos:

```
sudo rmmod simple
dmesg
```

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

Ejercicio 3 (40 puntos)

a. Si todo ha salido bien con los demás ejercicios, tómele 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-
```

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`.

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>`.
   - ¿Qué es y para qué sirve el archivo `fstab`?
   - ¿Qué almacena el directorio `/etc`? ¿En Windows, quién (hasta cierto punto) funge como `/etc`?
   - ¿Qué se almacena en `/dev` y en `/dev/disk`?

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).

- ¿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?
- ¿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 Rercord* (VBR), y su relación con UEFI.
- ¿Qué es hacer *chain loading*?
- ¿Qué se está indicando con la configuración `root="<el file system anotado>"`?

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`.
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`).
    - ¿Qué es `vmlinuz`?
i. En este mismo directorio elimine el subdirectorio `grub` con el siguiente comando:

```
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.
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
```

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.