

Universidad del Valle de Guatemala

Sistemas operativos

Fredy Velásquez

201011

## Laboratorio #4

### Ejercicio 1 (30 puntos)

a. Descargue la herramienta SystemTap con el siguiente comando:

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

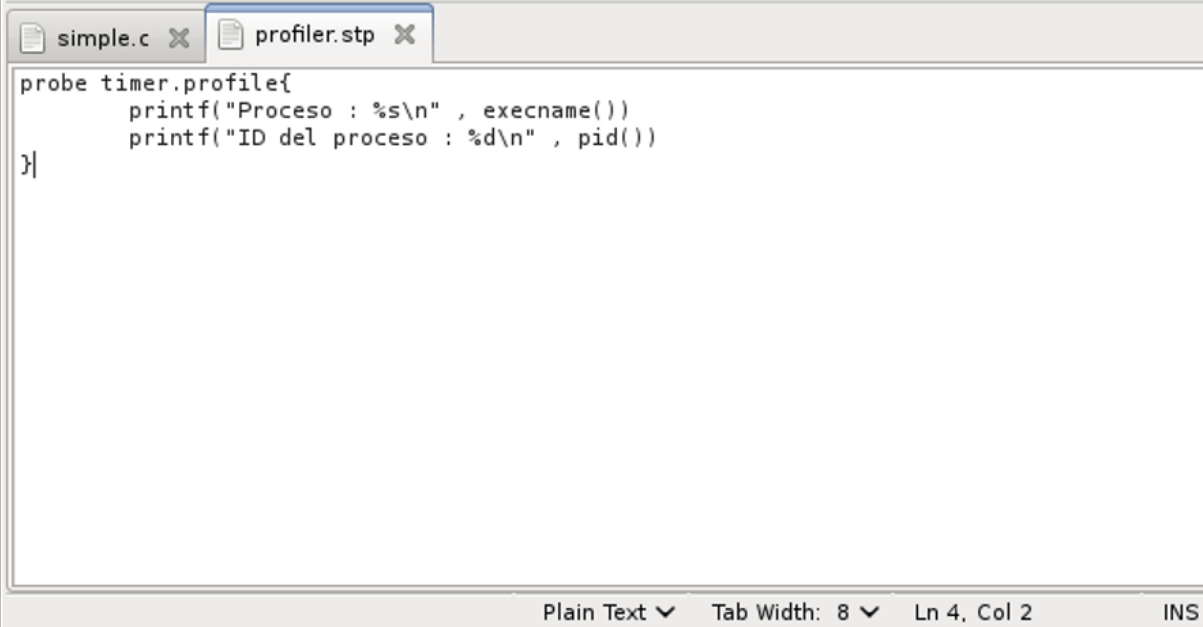
b. Cree un archivo llamado profiler.stp, con el siguiente código:

```
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.).



The screenshot shows a text editor window with two tabs: 'simple.c' and 'profiler.stp'. The 'profiler.stp' tab is active, displaying the following code:

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

The status bar at the bottom indicates 'Plain Text', 'Tab Width: 8', 'Ln 4, Col 2', and 'INS'.

- ¿Qué puede ver en el output cuando realiza estas acciones?  
Muestra en pantalla el tipo de proceso y el id que lo identifica

- ¿Para qué sirve SystemTap?

ayuda al usuario a hacer un diagnóstico de operaciones o actividades realizadas en el sistema operativo.

- ¿Qué es una probe?

Es un evento con su manejador correspondiente, al manejador también se le conoce como el cuerpo del probe.

- ¿Cómo funciona SystemTap?

Se definen eventos que ocurren en el sistema y se definen sub-rutinas que se deben ejecutar por el manejador al detectar ese evento. Systemtap traduce el script a C, lo corre en el compilador de C y crea un módulo kernel . después se carga el módulo y activan las probes.

- ¿Qué es hacer profiling y qué tipo de profiling se hace en este ejercicio?

Este es un profiling a nivel de sistema. Profiling consiste en hacer un diagnóstico del rendimiento monitoreando el comportamiento de un programa.

## Ejercicio 2 (30 puntos)

a. Abra su máquina virtual y tómese 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.

```
simple.c X
<linux/list.h>

//Metodo que retorna int
int simple_init(void) {
    printk(KERN_INFO "Loading Module\nprueba");
    return 0;
}

//metodo que no retorna nada
void simple_exit(void) {
    printk(KERN_INFO "Removing Module\nprueba2");
}

module_init(simple_init);
module_exit(simple_exit);
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Test personalizado qlg");
MODULE_AUTHOR("cou");
```

- ¿Cuál es la diferencia en C entre un método que no recibe parámetros y uno que recibe void?

Void explícitamente indica que no recibe parámetros mientras que un método que no recibe parámetros en realidad queda inespecificado, puede recibir lo que sea.

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?

Printf esta realizando la impresión en pantalla a nivel de aplicación, con modo usuario , mientras que printk esta realizando la impresión desde nivel de kernel, en modo kernel.

- ¿Qué es y para qué sirve KERN\_INFO?

Nos ofrece un punto de acceso al registro (logs) del kernel.

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?

goal definition permite definir el target que un makefile busca sobrescribir.

- ¿Qué función tienen las líneas all: y clean:?

all carga todos los módulos en kernel y clean limpia los módulos del kernel

- ¿Qué hace la opción -C en este Makefile?

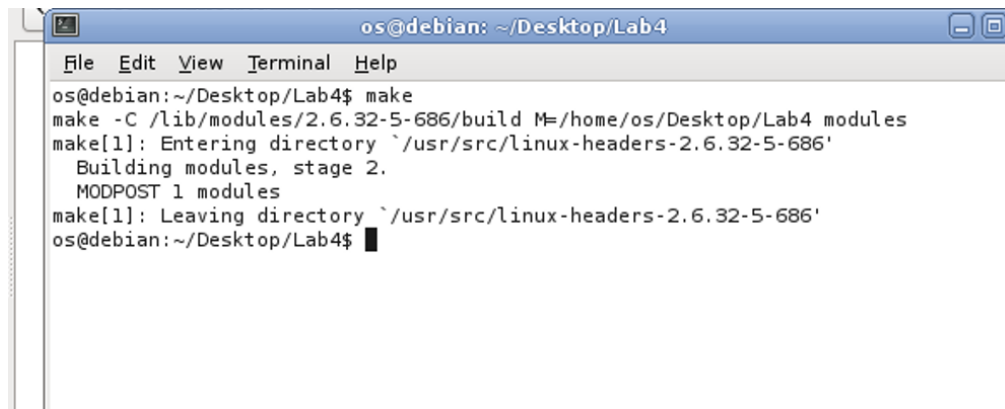
Retorna la ubicación del modulo

- ¿Qué hace la opción M en este Makefile?

ejecuta build en módulos externos.

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

Makefile.



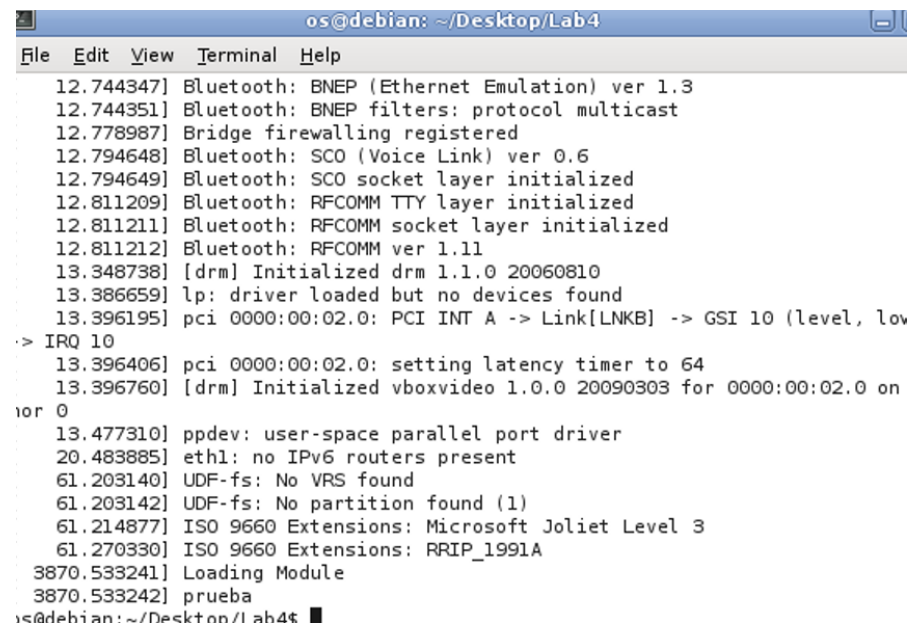
```
os@debian: ~/Desktop/Lab4
File Edit View Terminal Help
os@debian:~/Desktop/Lab4$ make
make -C /lib/modules/2.6.32-5-686/build M=/home/os/Desktop/Lab4 modules
make[1]: Entering directory `/usr/src/linux-headers-2.6.32-5-686'
Building modules, stage 2.
MODPOST 1 modules
make[1]: Leaving directory `/usr/src/linux-headers-2.6.32-5-686'
os@debian:~/Desktop/Lab4$
```

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.



```
os@debian: ~/Desktop/Lab4
File Edit View Terminal Help
12.744347] Bluetooth: BNEP (Ethernet Emulation) ver 1.3
12.744351] Bluetooth: BNEP filters: protocol multicast
12.778987] Bridge firewalling registered
12.794648] Bluetooth: SCO (Voice Link) ver 0.6
12.794649] Bluetooth: SCO socket layer initialized
12.811209] Bluetooth: RFCOMM TTY layer initialized
12.811211] Bluetooth: RFCOMM socket layer initialized
12.811212] Bluetooth: RFCOMM ver 1.11
13.348738] [drm] Initialized drm 1.1.0 20060810
13.386659] lp: driver loaded but no devices found
13.396195] pci 0000:00:02.0: PCI INT A -> Link[LNKb] -> GSI 10 (level, low
-> IRQ 10
13.396406] pci 0000:00:02.0: setting latency timer to 64
13.396760] [drm] Initialized vboxvideo 1.0.0 20090303 for 0000:00:02.0 on
ior 0
13.477310] ppdev: user-space parallel port driver
20.483885] eth1: no IPv6 routers present
61.203140] UDF-fs: No VRS found
61.203142] UDF-fs: No partition found (1)
61.214877] ISO 9660 Extensions: Microsoft Joliet Level 3
61.270330] ISO 9660 Extensions: RRIP_1991A
3870.533241] Loading Module
3870.533242] prueba
os@debian:~/Desktop/Lab4$
```

- ¿Para qué sirve `dmesg`?

muestra los mensajes de diagnostico en pantalla

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

carga en el kernel el modulo para que se pueda usar.

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.

```

[ 12.778987] Bridge firewalling registered
[ 12.794648] Bluetooth: SCO (Voice Link) ver 0.6
[ 12.794649] Bluetooth: SCO socket layer initialized
[ 12.811209] Bluetooth: RFCOMM TTY layer initialized
[ 12.811211] Bluetooth: RFCOMM socket layer initialized
[ 12.811212] Bluetooth: RFCOMM ver 1.11
[ 13.348738] [drm] Initialized drm 1.1.0 20060810
[ 13.386659] lp: driver loaded but no devices found
[ 13.396195] pci 0000:00:02.0: PCI INT A -> Link[LNKB] -> GSI 10 (level, low)
-> IRQ 10
[ 13.396406] pci 0000:00:02.0: setting latency timer to 64
[ 13.396760] [drm] Initialized vboxvideo 1.0.0 20090303 for 0000:00:02.0 on mi
nor 0
[ 13.477310] ppdev: user-space parallel port driver
[ 20.483885] eth1: no IPv6 routers present
[ 61.203140] UDF-fs: No VRS found
[ 61.203142] UDF-fs: No partition found (1)
[ 61.214877] ISO 9660 Extensions: Microsoft Joliet Level 3
[ 61.270330] ISO 9660 Extensions: RRIP_1991A
[ 3870.533241] Loading Module
[ 3870.533242] prueba
[ 4020.479941] Removing Module
[ 4020.479944] prueba2
os@debian:~/Desktop/Lab4$ █

```

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

Descarga y remueve los módulos del kernel

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

### 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
```

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”.

- 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`.

- 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?

Contiene información de configuración para poder montar y desmontar file systems del sistema.

- ¿Qué almacena el directorio /etc? ¿En Windows, quién (hasta cierto punto) funge como /etc?

similar a system32, contiene las configuraciones de las aplicaciones y programas del sistema operativo

- ¿Qué se almacena en /dev y en /dev/disk?

Dev contiene la información de todos los dispositivos conectados al equipo mientras que dev/disk contine la información específicamente de dispositivos de almacenamiento

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 Record (VBR), y su relación con UEFI.

- ¿Qué es hacer chain loading?

Se refiere a la carga de un programa sobre otro que está en ejecución c, como remplazo .

- ¿Qué se está indicando con la configuración root="<el file system anotado>"?

con root se indica la partición del disco duro que se debe usar como raíz.

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.

- Mencione tres diferencias funcionales entre GRUB y LILO

- Configuración: GRUB es más flexible que LILO en términos de configuración. Por ejemplo, GRUB puede detectar automáticamente los sistemas operativos instalados en un sistema y mostrarlos en un menú de arranque, lo que facilita la selección del sistema operativo deseado. Por otro lado, LILO requiere que el usuario edite manualmente el archivo de configuración cada vez que se añade o se quita un kernel.
- Soporte de sistemas de archivos: GRUB es capaz de leer una amplia variedad de sistemas de archivos, incluyendo ext2, ext3, ext4, NTFS, y FAT. Por otro lado, LILO sólo es capaz de leer sistemas de archivos Linux nativos como ext2 y ext3.



- Ubicación del registro de arranque: GRUB se instala en el registro de arranque maestro (MBR) del disco duro, mientras que LILO se instala en el primer sector del sistema de archivos raíz. Esto significa que si el sector de arranque de LILO se daña, el sistema no podrá arrancar, mientras que en el caso de GRUB, el daño se limita al MBR y se puede solucionar más fácilmente.