Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Sistemas Operativos 2

Documentación Proyecto

Robin Armando Salvatierra Bautista

200915428

5 de julio de 2019

INDICE

Contenido

INTRODUCCIÓN

El siguiente documento tiene como objetivo dar a conocer como las especificaciones y conocimientos básicos en cuanto a la aplicación de módulos de kernel y el comando strace que fueron utilizados para la realización del proyecto.

MODULOS A IMPLEMENTAR

Para el primer módulo de kernel deberá interceptar la llamada al sistema que el comando rm utiliza para luego realizar el backup del archivo.

Para el segundo módulo de kernel se deberá de monitorear el directorio root "/" y registrar cualquier borrado, modificación o creación en el archivo /var/proyecto/#carnet_files.log

LOG DE SEÑALES DE LA TERMINAL

Usar el comando strace para crear un log de todas las señales enviadas por el proceso de la terminal, incluyendo los hijos y guardarlo en /var/log/signals.log.

INTRODUCCION A MODULOS DE KERNEL

Modulo de ejemplo:

Primero de todo deberemos incluir algunos **headers** que contienen definiciones que vamos a necesitar:

```
#include <linux/init.h>
#include <linux/module.h>
```

A continuación deberemos definir las funciones de inicialización i destrucción del modulo en las cuales para este sencillo ejemplo haremos un simple **printk**

```
static int hello_world_init(void)
{
        printk(KERN_ALERT "Hello World!\n");
        return 0;
}
```

En la función de destrucción haremos lo mismo:

```
static void hello_world_exit(void)
{
        printk(KERN_ALERT "Bye World!\n");
}
```

Finalmente deberemos indicar como hemos llamado a las funciones mediante

```
module_init(hello_world_init);
module_exit(hello_world_exit);
```

Para poder compilar se crea el archivo de **Makefile**:

```
obj-m += helloworld.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
```

Es obligatorio que antes de los comandos "**make**" haya un tabulador y no un conjunto de espacios, sino no va a reconocer el formato del **Makefile**.

Con el comando **make** vamos a compilar el **modulo del kernel**:

```
# make
make -C /lib/modules/2.6.18-53.1.21.el5PAE/build M=/home/jordi/helloworld
modules
make[1]: Entering directory `/usr/src/kernels/2.6.18-53.1.21.el5-PAE-
i686'
  CC [M]  /home/jordi/helloworld/helloworld.o
  Building modules, stage 2.
  MODPOST
  CC      /home/jordi/helloworld/helloworld.mod.o
  LD [M]  /home/jordi/helloworld/helloworld.ko
```

```
make[1]: Leaving directory `/usr/src/kernels/2.6.18-53.1.21.el5-PAE-i686'
```

Mediante un **ls** podemos ver los ficheros que ha generado:

```
# ls
helloworld.c  helloworld.ko  helloworld.mod.c  helloworld.mod.o
helloworld.o  Makefile  Module.symvers
```

Podemos cargar el módulo mediante **insmod**:

```
insmod helloworld.ko
```

A continuación mediante el **dmesg** podremos ver el **Hello World**:

```
Hello World!
```

Para ver el "Bye World" lo descargamos mediante **rmmod**:

```
rmmod helloworld.ko
```

Y de nuevo, con **dmesg**, podremos ver el mensaje de despedida:

```
helloworld!
Bye World!
```

MODULO 1

El algoritmo del modulo 1 consiste en los siguientes pasos:

- Se cambian los permisos de la pagina para escritura
- Se guarda el valor de memoria de la llamada original
- Se inserta la nueva función a la dirección de memoria de la llamada original
- En esta función se realiza los cambios requeridos por el modulo a realizar y se retorna la llamada original
- Se cambia la dirección de memoria a modo de escritura
- Se regresa la función original a la dirección de la memoria
- Se proteger la pagina contra escritura es decir se cambian de nuevo los permisos de la pagina a modo lectura

MODULO 2

El algoritmo del modulo 2 consiste en los siguientes pasos:

- Se inicializa el monitor de eventos Inotify
- Se agrega un observador inicializado con el directorio
- Dentro de un bucle infinito se hace la validación para verificar que no se tengan errores en la lectura
- En cada iteración según el evento ya sea creación modificación o eliminación se escribe dentro del log la actividad realizada

LOG DE SEÑALES

Strace

strace es una herramienta de línea de comandos útil para diagnosticar, dar instrucciones y ejecutar tarea de depuración. Los administradores del sistema encontramos en strace una herramienta practica para resolver problemas con programas para los cuales la fuente no está disponible ya que no necesitan ser recompilados para rastrearlos.

La herramienta strace captura y registra todas las llamadas al sistema realizadas por un proceso y las señales recibidas por el proceso, strace se encarga de mostrar el nombre de cada llamada al sistema junto con sus argumentos entre paréntesis y su valor de retorno a error estándar, opcionalmente, será posible redirigirlo a un archivo.

Se solicia crear un log de todas las señales enviadas por el proceso de la terminal, incluyendo los hijos y guardarlo en /var/log/signals.log.

El cual se realizo con el siguiente comando

strace -fp 2939 -o /var/log/signal.log -e trace=signal

Donde los parámetros del comando significan

-p

Adjunte al proceso con el ID de proceso pid y comience a rastrear.

FILTRADO

-e trace = signal

Rastrear todas las llamadas del sistema relacionadas con la señal

RASTREO

-f

Realice el seguimiento de los procesos secundarios a medida que se crean mediante los procesos actualmente rastreados como resultado de las llamadas del sistema fork (2), vfork (2) y clone (2).

FORMATO DE SALIDA

-o

Escriba la salida de rastreo en el nombre de archivo del archivo en lugar de en stderr.

CODIGO FUENTE

https://github.com/robinprousac/SO2_P1

Referencias

https://www.systutorials.com/docs/linux/man/1-strace/

https://linux.die.net/man/1/strace

https://www.tldp.org/LDP/lkmpg/2.6/html/x121.html

http://www.haifux.org/lectures/22/syscall_logger/syscall_logger_module.c

https://en.wikipedia.org/wiki/System.map

https://bbs.archlinux.org/viewtopic.php?id=236875

https://askubuntu.com/questions/47515/any-way-to-search-for-text-within-nano