UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
SISTEMAS OPERATIVOS 1
SECCIÓN A

PROYECTO 1

MANUAL DE MODULOS

Carnet	Nombre
201801677	Julio Emiliano Cifuentes Zabala

Tabla de contenido

Manual Técnico

El manual y las librerías que se agregaron fueron utilizadas en la distribución de Linux Ubuntu 20.04 LTS.

Módulo de Memoria

Librerías utilizadas

Para el módulo de memoria se utilizaron las siguientes librerías del lenguaje C

```
#include <linux/module.h> // obligatorio para crear un modulo
#include <linux/kernel.h> // para usar KERN_INFO
#include <linux/init.h> // para los macros module_init y module_exit
#include <linux/proc_fs.h> // header para utilizar proc_fs
#include <asm/uaccess.h> // for copy_from_user
#include <linux/seq_file.h> // libreria seq_file y manejar el archivo en /proc/
#include <linux/hugetlb.h> // para utilizar si_meminfo
```

La información obtenida para crear el módulo de memoria fue a través de la estructura sysinfo que es provista por la librería kernel.h

El struct sysinfo contiene estadísticas y datos de la memoria de la computadora donde se encuentra instalada la distribución de Linux, como el uso es la memoria RAM y swap.

Contenido del Struct sysinfo

```
struct sysinfo {
        long uptime;          /* Seconds since boot */
        unsigned long loads[3];  /* 1, 5, and 15 minute load averages */
        unsigned long totalram;  /* Total usable main memory size */
        unsigned long freeram;   /* Available memory size */
        unsigned long sharedram; /* Amount of shared memory */
        unsigned long bufferram; /* Memory used by buffers */
        unsigned long totalswap; /* Total swap space size */
        unsigned long freeswap;  /* Swap space still available */
        unsigned short procs;    /* Number of current processes */
        unsigned long totalhigh; /* Total high memory size */
        unsigned long freehigh;  /* Available high memory size */
```

```
    unsigned int mem_unit;   /* Memory unit size in bytes */
    char _f[20-2*sizeof(long)-sizeof(int)];
                    /* Padding to 64 bytes */
};
```

La información de la memoria que provee esta estructura es en bytes y los valores que muestra no son los reales, por lo que se utiliza su atributo “mem_unit” para que al mutiplicarlo se obtenga el valor real. Este atributo es un múltiplo de los valores de memoria dados.

Funciones utilizadas

- Static int escribir_archivo(struct seq_file *archivo, void *v): Función donde se encuentran las instrucciones a ejecutar al leer el modulo que se escribe. En esta función se obtienen los datos de la estructura sysinfo y se escriben en el modulo a crear.
  La información escrita se encuentra en un formato tipo JSON para que sea más fácil visualizarlo.

- Static int abrir(struct inode *inode, struct file *file): Función que se ejecuta la función anterior escribir_archivo

- Static int _insert(void): Funcion que se ejecuta al escribir el comando insmod, se encarga de crear el modulo indicado e imprimir el mensaje de confirmación “201801677”.

- Static void _remove(void): Funcion que se ejecuta al escribir el comando rmmod, se encarga de eliminar el modulo indicado e imprimir el mensaje de confirmación “Sistemas Operativos 1”

- Module_init(_insert) y module_init(_remove): ejecutan las funciones para crear y eliminar el módulo.

Archivo Make

El archivo make contiene instrucciones para compilar el módulo escrito en C. Su contenido es el siguiente:

```
obj-m += memo_201801677.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
```

Comandos utilizados

- Make: comando para compilar y generar el archivo .ko
- Insmod: comando para montar el modulo en el directorio /proc
  Ejemplo:
    Sudo insmod memo_201801677.ko
- Rmmod: comando para desmontar/eliminar el modulo en el directorio /proc
  Ejemplo:
    Sudo rmmod memo_201801677
- Dmesg: comando para mostrar los mensajes del kernel
  Ejemplo:
    Dmesg (mostrar mensajes)
    Sudo dmesg -C (eliminar buffer de mensajes)

Módulo de CPU

Librerías utilizadas

Para el módulo de cpu se utilizaron las siguientes librerías del lenguaje C

```
#include <linux/module.h> // obligatorio para crear un modulo
#include <linux/kernel.h> // para usar KERN_INFO
#include <linux/init.h> // para los macros module_init y module_exit
#include <linux/proc_fs.h> // header para utilizar proc_fs
#include <asm/uaccess.h> // for copy_from_user
#include <linux/seq_file.h> // libreria seq_file y manejar el archivo en /proc/
#include <linux/mm.h>  // get_mm_rss()
#include <linux/sched.h>
```

La información obtenida para crear el módulo de cpu fue a través de la estructura task_struct que es provista por la librería /Linux/sched.h

El struct task_struct contiene estadísticas y datos de los procesos de la computadora donde se encuentra instalada la distribución de Linux, incluyendo su process id, su nombre, su estado, etc.

Contenido del Struct task_struct

Los atributos utilizados de esta estructura fueron los siguientes

proceso->state (para obtener el estado del proceso)

proceso-> comm (para obtener le nombre del proceso)

proceso->pid (id del proceso)

proceso->real_cred->uid (id del usuario que ejecutó el proceso)

get_mm_rss(proceso->mm) (% de ram del proceso)

proceso->children (para obtener los procesos hijos)

Funciones utilizadas

- Static int escribir_archivo(struct seq_file *archivo, void *v): Función donde se encuentran las instrucciones a ejecutar al leer el módulo que se escribe. En esta función se obtienen los datos de la estructura task_struct y se escriben en el módulo a crear.
  La información escrita se encuentra en un formato tipo JSON para que sea más fácil visualizarlo.

- Static int abrir(struct inode *inode, struct file *file): Función que se ejecuta la función anterior escribir_archivo

- Static int _insert(void): Funcion que se ejecuta al escribir el comando insmod, se encarga de crear el modulo indicado e imprimir el mensaje de confirmación “Julio Emiliano Cifuentes Zabala”.

- Static void _remove(void): Funcion que se ejecuta al escribir el comando rmmod, se encarga de eliminar el modulo indicado e imprimir el mensaje de confirmación “Diciembre 2021”

- Module_init(_insert) y module_init(_remove): ejecutan las funciones para crear y eliminar el módulo.

Archivo Make

El archivo make contiene instrucciones para compilar el modulo escrito en C. Su contenido es el siguiente:

```
obj-m += cpu_201801677.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
```

Comandos utilizados

- Make: comando para compilar y generar el archivo .ko
- Insmod: comando para montar el módulo en el directorio /proc
  Ejemplo:
  Sudo insmod cpu_201801677.ko
- Rmmod: comando para desmontar/eliminar el módulo en el directorio /proc
  Ejemplo:
  Sudo rmmod cpu_201801677
- Dmesg: comando para mostrar los mensajes del kernel
  Ejemplo:
  Dmesg (mostrar mensajes)
  Sudo dmesg -C (eliminar buffer de mensajes)