Universidad San Carlos de Guatemala

Facultad de ingeniería

Escuela de Ciencias y Sistemas

Sistemas Operativos 2

Manual Tecnico

ID	Nombre
201701133	Selvin Lisandro Aragón Pérez
201700529	Cristian Manases Juarez Juarez

Manual técnico

En este manual se explicara de manera detalla el uso, implemetacion de modulos kernel en el sistema.

IMPLEMENTACION DE MODULOS KERNEL

¿Dónde están y cómo los veo?

Los módulos son almacenados en un directorio especial

- Ver módulos almacenados

```
$ /lib/modules/[nombre_kernel]
```

- Ver la versión del kernel

```
$ uname -r
```

- Revisar módulos instalados, esto listará todos los módulos instalados en el kernel

```
$ lsmod
```

- Revisamos o "buscamos" unicamente un módulo

```
$ lsmod | grep "[nombre_modulo]"
```

MAKEFILE

Aparte, necesitaremos de un archivo especial llamado Makefile; este contendrá el siguiente contenido:

- `Archivo: Makefile`

```
obj-m += timestamps.o # Definir el nombre del archivo que esperamos de salidaall:
```

- `Definir que se hará cuando se compile`

```
make -C /lib/modules/$(shell uname -r)/build M=$(shell pwd)
```

- `Definir que se hará cuando se limpie el módulo`

```
modulesclean:make -C /lib/modules/$(shell uname -r)/build M=$(shell pwd) clean
```

Pasos a seguir para montar un módulo

- Instalar linux headers

```
$ sudo apt-get install linux-headers-$(uname -r)
```

- # Instalar dependencias para compilar c y c++

```
$ sudo apt-get install build-essential
```

- Compilar archivos

```
$ make
```

- Montar el modulo

```
$ sudo insmod [nombre_modulo].ko
```

- Para revisar el log es con

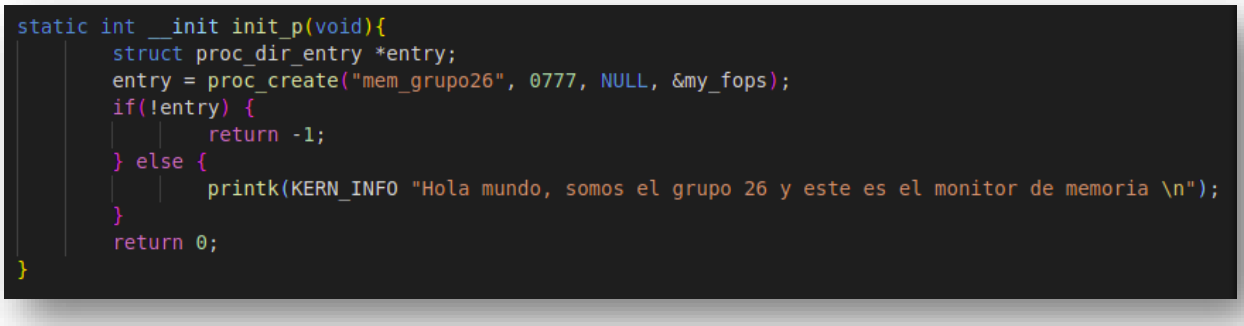
```
$ sudo dmesg
```

- Para desmontar un modulo

```
$ sudo rmmod "[nombre_modulo]"
```

Código para el modulo Ram

Cuando uno quiere mostrar mostrar algo ejecutar algo al momento de montar un modulo se ejecuta en _init



```c
static int __init init_p(void){
        struct proc_dir_entry *entry;
        entry = proc_create("mem_grupo26", 0777, NULL, &my_fops);
        if(!entry) {
                return -1;
        } else {
                printk(KERN_INFO "Hola mundo, somos el grupo 26 y este es el monitor de memoria \n");
        }
        return 0;
}
```

Cuando uno quiere mostrar mostrar algo ejecutar algo al momento de desmontar un modulo se ejecuta en _exit

```c
static void __exit exit_p(void){
        remove_proc_entry("mem_grupo26",NULL);
        printk(KERN_INFO "Sayonara mundo, somos el grupo 26 y este fue el monitor de memoria \n");
}
```

El codigo para obtener los datos de la memoria es el siguiente:

- Sysinfo: Nos permite obtener los datos de la memoria como un struct
- Freeram: Obtenemos la memoria libre en el sistema
- Totalram: Obtenemos el total de la memoria

```c
static int my_proc_show(struct seq_file *m, void *v){
    //Memoria en MB
    struct sysinfo inf;
    int32_t ramLibre;
    int32_t ramTotal;
    int32_t ramocupado;
    int32_t porcentaje;
    si_meminfo(&inf);
    ramLibre = ((inf.freeram*inf.mem_unit)/(1024*1024));
    ramTotal = ((inf.totalram*inf.mem_unit)/(1024*1024));
    ramocupado = ramTotal - ramLibre;
    porcentaje = ((ramocupado * 100) / ramTotal) ;
    seq_printf(m, "{ \"total\": %d , \"consumida\": %d , \"porcentaje\": %d}", ramTotal, ramocupado, p
    return 0;
}
```

Código para el modulo de Procesos

En el caso del modulo de procesos al montar y desmontar el modulo se efectua de la misma manera (_init y _exit).

Para la obtencion de los datos de los procesos en el sistema se utiliza task_struct

```
//Procesos
struct task_struct *task;
int32_t registered = 0;
int32_t running = 0;
int32_t sleeping = 0;
int32_t stopped = 0;
int32_t zombie = 0;
unsigned long rss;
```

Para obtener el estado los estados de los procesos se utilizan la funcion task -> state. En el for each se recorre cada una de las tareas y las compara para obtener el timpo de estado en el que se encuentra el proceso.

```
for_each_process(task){
    get_task_struct(task);
    registered++;
    if(task -> state == TASK_RUNNING || task -> state == TASK_TRACED ){
        running++;
    }else if(task -> state == TASK_INTERRUPTIBLE || task -> state == TASK_UNINTERRUPTIBLE ){
        sleeping++;
    }else if(task -> state ==  __TASK_STOPPED || task -> state ==  TASK_STOPPED ){
        stopped++;
    }else if(task -> state == EXIT_ZOMBIE || task -> state ==  EXIT_DEAD){
        zombie++;
    }else{
        registered--;
    }
}
```

Luego de obtener todos los estados se procede a obtener los campos id, nombre, estado, porcentaje de ram, usuario, memoria ocupara,con las siguientes funciones:

- Pid: Nos permite obtener el id del proceso
- Comm: Nos permite obtener el nombre del proceso.
- State: Nos permite obtener el estado del proceso
- End_code: Obtenemos el fin del segmento de memoria
- Start_code: Obtenemos el inicio del segmento de memoria
- Cred-> uid.val: Obtenemos el usuario que ejecuto el proceso.

```
task->pid, task->comm, task->state, rss, task->mm->end_code - task->mm->start_code, task->cred->uid.va
```