UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE SISTEMAS

PROYECTO 1

200915347 - 200915466

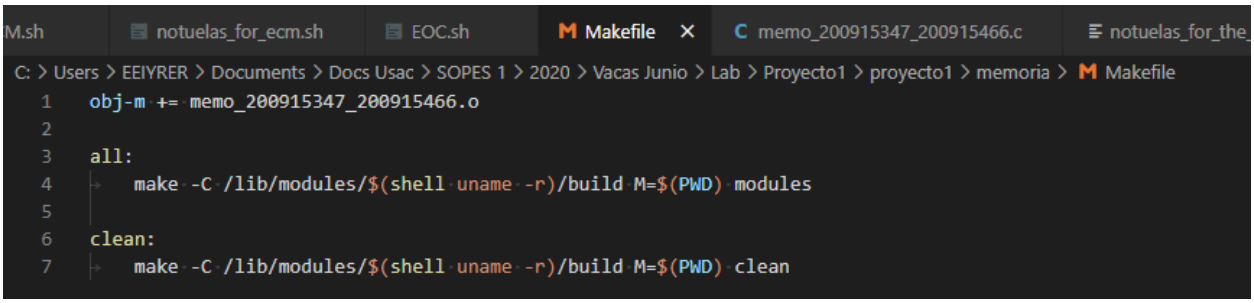
ERICK FERNANDO REYES MANCILLA
OLIVER ALEXANDER RODAS MENDOZA
SISTEMAS OPERATIVOS 1

INDICE

MÓDULOS

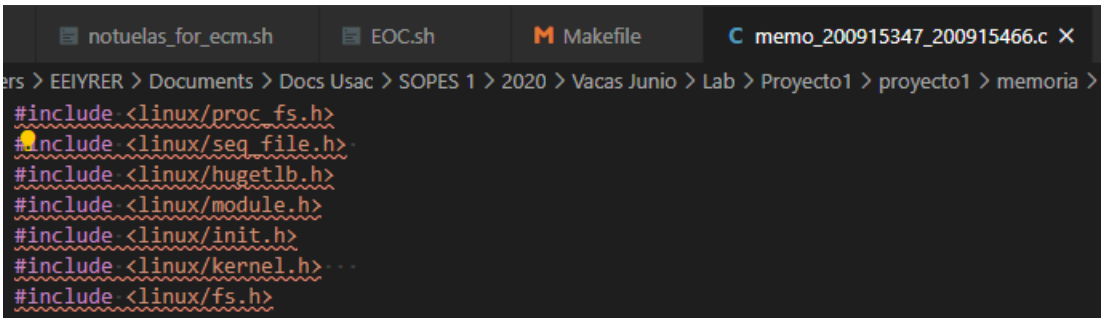
MÓDULO DE MEMORIA

En el archivo ***MakeFile*** para este módulo se coloca el nombre de nuestro archivo escrito en C con la extensión ".o", a continuación, colocamos la ruta en la cual se encuentran los módulos de nuestro sistema operativo debido a que existe diversidad de kernels es más fácil colocar en la ruta **`$(Shell uname -r)`** el cual es un comando que nos regresa la versión del kernel que tenemos. Se agrega el parámetro -> **`all`** que permite compilar el módulo del kernel y **`clean`** que permite borrar los archivos generados al compilar el módulo.



Se crea un archivo en C donde utilizaremos las bibliotecas que proporciona las bibliotecas que proporciona el kernel, el archivo contendrá la siguiente estructura:

Se importan las librerías proporcionadas por el kernel.



Descripción del módulo y definición de la estructura **`sysinfo`**.

Declaramos una función que recibe la referencia a nuestro archivo. Se carga la información de la memoria a la estructura de **`sysinfo`** de la cual obtendremos los datos relevantes de la memoria RAM de nuestro sistema operativo. También se utiliza la estructura **`seq_printf`** para imprimir en nuestro archivo.

```c
static int escribir_archivo(struct seq_file * archivo, void *v) {
    si_meminfo(&inf);

    seq_printf(archivo, "=============================================\n");
    seq_printf(archivo, "|  Laboratorio Sistemas Operativos 1        |\n");
    seq_printf(archivo, "|  Vacaciones junio 2020                    |\n");
    seq_printf(archivo, "|  Fernando Reyes  -  200915347             |\n");
    seq_printf(archivo, "|  Oliver Rodas    -  200915466             |\n");
    seq_printf(archivo, "|                                           |\n");
    seq_printf(archivo, "|    PROYECTO 1 (MODULO 1 - MEMORIA RAM)    |\n");
    seq_printf(archivo, "|                                           |\n");
    seq_printf(archivo, "=============================================\n");
    seq_printf(archivo, "  Sistema Operativo: Ubuntu 18.04\n");
    seq_printf(archivo, "  Memoria Total:  \t %8lu MB\n", (inf.totalram * 4) / 1024);
    seq_printf(archivo, "  Memoria Libre:  \t %8lu MB\n", (inf.freeram * 4) / 1024 );
    seq_printf(archivo, "  Memoria en uso: \t %8li %%\n", 100 - ((inf.freeram * 4 * 100) / (inf.totalram * 4)));

    return 0;
}
```

Se declaran los eventos que se disparan al abrir o cerrar el archivo que definimos en el paso anterior, de esta forma nuestro módulo se ejecuta cada vez que esto suceda.

```c
static int al_abrir(struct inode *inode, struct file *file) {
  return single_open(file, escribir_archivo, NULL);
}

static struct file_operations operaciones =
{
    .open = al_abrir,
    .read = seq_read
};

static int iniciar(void)
{
    proc_create("memo_200915347_200915466", 0, NULL, &operaciones);
    printk(KERN_INFO "Carné: 200915347_200915466\n");
    return 0;
}

static void salir(void)
{
    remove_proc_entry("memo_200915347_200915466", NULL);
    printk(KERN_INFO "Curso: Sistemas Operativos 1\n");
}

module_init(iniciar);
module_exit(salir);
```

Para la instalación del nuevo módulo nos dirigimos a la carpeta donde esta nuestro **Makefile** y el archivo C y a continuación ejecutamos en la terminal el comando **`make all`**.

```
PROBLEMS 5   OUTPUT   TERMINAL   DEBUG CONSOLE
nando@nando-so1:~/Documentos/proyecto1/memoria$ make all
make -C /lib/modules/5.3.0-28-generic/build M=/home/nando/Documentos/proyecto1/memoria modules
make[1]: se entra en el directorio '/usr/src/linux-headers-5.3.0-28-generic'
  CC [M]  /home/nando/Documentos/proyecto1/memoria/memo_200915347_200915466.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/nando/Documentos/proyecto1/memoria/memo_200915347_200915466.mod.o
  LD [M]  /home/nando/Documentos/proyecto1/memoria/memo_200915347_200915466.ko
make[1]: se sale del directorio '/usr/src/linux-headers-5.3.0-28-generic'
nando@nando-so1:~/Documentos/proyecto1/memoria$
```

Ahora instalamos el módulo con el comando: **`sudo insmod memo_200915347_200915466.ko`**

```
nando@nando-so1:~/Documentos/proyecto1/memoria$ sudo insmod memo_200915347_200915466.ko
nando@nando-so1:~/Documentos/proyecto1/memoria$
```

Revisamos los mensajes del log con el comando: **`dmesg`**.

```
nando@nando-so1:~/Documentos/proyecto1/memoria$ dmesg
[    0.000000] Linux version 5.3.0-28-generic (buildd@lcy01-amd64-009) (gcc version 7.4.0 (Ubuntu 7.4.0-1ubuntu1~18.04.1)) #30~18.04.1-Ubuntu SMP Fri Jan 17 06:14:09 UTC 2020 (Ubuntu 5.3.
28.30~18.04.1-generic 5.3.13)
[    0.000000] Command line: BOOT_IMAGE=/vmlinuz-5.3.0-28-generic root=UUID=fce57b93-b390-437c-9e2b-3ad0f8f61b00 ro quiet splash
[    0.000000] KERNEL supported cpus:
[    0.000000]   Intel GenuineIntel
[    0.000000]   AMD AuthenticAMD
[    0.000000]   Hygon HygonGenuine
[    0.000000]   Centaur CentaurHauls
[    0.000000]   zhaoxin   Shanghai
```

```
[22208.892570] Carné: 200915347_200915466
nando@nando-so1:~/Documentos/proyecto1/memoria$
```

Podemos ver que ha sido ejecutado con éxito al revisar el archivo creado en la carpeta proc con el comando: **`cat /proc/memo_200915347_200915466`**

```
nando@nando-so1:~/Documentos/proyecto1/memoria$ cat /proc/memo_200915347_200915466
=============================================
|  Laboratorio Sistemas Operativos 1        |
|  Vacaciones junio 2020                    |
|  Fernando Reyes   -   200915347           |
|  Oliver Rodas     -   200915466           |
|                                           |
|      PROYECTO 1 (MODULO 1 - MEMORIA RAM)  |
|                                           |
=============================================
  Sistema Operativo: Ubuntu 18.04
  Memoria Total:             3936 MB
  Memoria Libre:              889 MB
  Memoria en uso:              78 %
nando@nando-so1:~/Documentos/proyecto1/memoria$
```

Desinstalamos el módulo con el comando: **`sudo rmmod memo_200915347_200915466`**

```
PROBLEMS 5   OUTPUT   TERMINAL   DEBUG CONSOLE
nando@nando-so1:~/Documentos/proyecto1/memoria$ sudo rmmod memo_200915347_200915466
nando@nando-so1:~/Documentos/proyecto1/memoria$
```

Revisamos el log con el comando: **`dmesg`**.

```
PROBLEMS 5   OUTPUT   TERMINAL   DEBUG CONSOLE                                    1: bash
nando@nando-so1:~/Documentos/proyecto1/memoria$ dmesg
[    0.000000] Linux version 5.3.0-28-generic (buildd@lcy01-amd64-009) (gcc version 7.4.0 (Ubuntu 7.4.0-1ubuntu1~18.04.1)) #30~18.04.1-Ubuntu SMP Fri Jan 17 06:14:09 UTC 2020 (Ubuntu 5.3.0
-28.30~18.04.1-generic 5.3.13)
[    0.000000] Command line: BOOT_IMAGE=/vmlinuz-5.3.0-28-generic root=UUID=fce57b93-b390-437c-9e2b-3ad0f8f61b00 ro quiet splash
[    0.000000] KERNEL supported cpus:
[    0.000000]   Intel GenuineIntel
[    0.000000]   AMD AuthenticAMD
[    0.000000]   Hygon HygonGenuine
[    0.000000]   Centaur CentaurHauls
[    0.000000]   zhaoxin   Shanghai
[    0.000000] x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point registers'
```

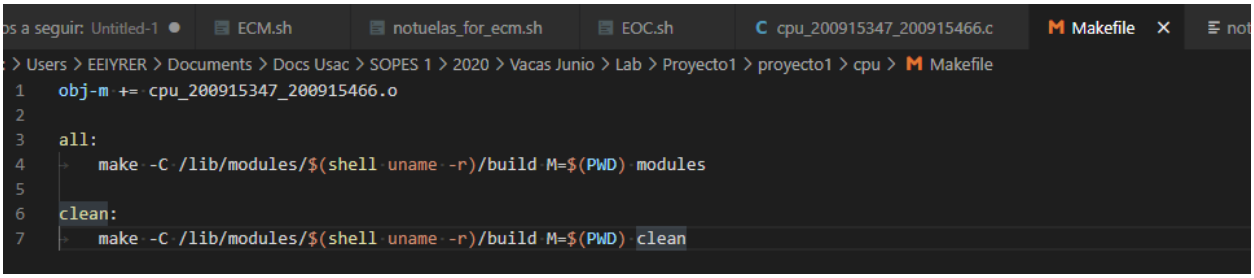
```
[22208.892570] Carné: 200915347_200915466
[22915.162941] Curso: Sistemas Operativos 1
nando@nando-so1:~/Documentos/proyecto1/memoria$
```

Se remueven todos los archivos generados del módulo con el comando**`: make clean`**.

```
PROBLEMS 5   OUTPUT   TERMINAL   DEBUG CONSOLE                                    1: bash
nando@nando-so1:~/Documentos/proyecto1/memoria$ make clean
make -C /lib/modules/5.3.0-28-generic/build M=/home/nando/Documentos/proyecto1/memoria clean
make[1]: se entra en el directorio '/usr/src/linux-headers-5.3.0-28-generic'
  CLEAN   /home/nando/Documentos/proyecto1/memoria/Module.symvers
make[1]: se sale del directorio '/usr/src/linux-headers-5.3.0-28-generic'
nando@nando-so1:~/Documentos/proyecto1/memoria$
```

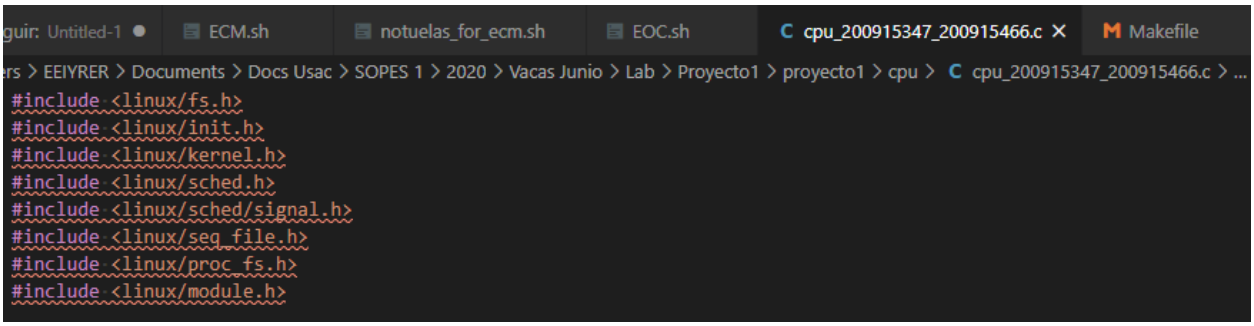
MÓDULO DE CPU

En el archivo ***MakeFile*** para este módulo se coloca el nombre de nuestro archivo escrito en C con la extensión ".o", a continuación, colocamos la ruta en la cual se encuentran los módulos de nuestro sistema operativo debido a que existe diversidad de kernels es más fácil colocar en la ruta **`$(Shell uname -r)`** el cual es un comando que nos regresa la versión del kernel que tenemos. Se agrega el parámetro -> **`all`** que permite compilar el módulo del kernel y **`clean`** que permite borrar los archivos generados al compilar el módulo.

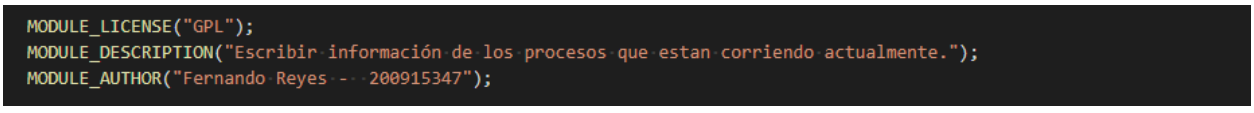


Se crea un archivo en C donde utilizaremos las bibliotecas que proporciona las bibliotecas que proporciona el kernel, el archivo contendrá la siguiente estructura:

Se importan las librerías proporcionadas por el kernel.



Descripción del módulo.



Declaramos una función que recibe la referencia a nuestro archivo. Se carga la información de los procesos y subprocesos haciendo uso de las estructuras propias del kernel **`task_struct`** y **`list_head.`**

```c
static int escribir_archivo(struct seq_file * archivo, void *v) {
    struct task_struct *task;
    struct task_struct *task_child;
    struct list_head *list;

    seq_printf(archivo, "=============================================\n");
    seq_printf(archivo, "|  Laboratorio Sistemas Operativos 1        |\n");
    seq_printf(archivo, "|  Vacaciones junio 2020                    |\n");
    seq_printf(archivo, "|  Fernando Reyes  -  200915347             |\n");
    seq_printf(archivo, "|  Oliver Rodas  -  200915466               |\n");
    seq_printf(archivo, "|                                           |\n");
    seq_printf(archivo, "|        PROYECTO 1 (MODULO 2 - CPU)        |\n");
    seq_printf(archivo, "|                                           |\n");
    seq_printf(archivo, "=============================================\n");
    seq_printf(archivo, "                                             \n");

    for_each_process(task){
        seq_printf(archivo, "---------------------------------------------\n");
        seq_printf(archivo, "  PID:  \t %d \n", task->pid);
        seq_printf(archivo, "  Nombre: \t %s \n", task->comm);
        seq_printf(archivo, "  Estado: \t ");
        switch(task->state){
            case 0:
                seq_printf(archivo, "Ejecutando\n");
                break;
            case 1:
                seq_printf(archivo, "Listo\n");
                break;
            case 2:
                seq_printf(archivo, "Durmiendo\n");
                break;
            case 4:
                seq_printf(archivo, "Zombie\n");
                break;
            case 8:
                seq_printf(archivo, "Detenido\n");
                break;
            case 32:
                seq_printf(archivo, "En Espera\n");
                break;
            default:
                seq_printf(archivo, "Desconocido\n");
                break;
        }
```

```c
        seq_printf(archivo, "  Hijos: \n");

        list_for_each(list, &task->children) {
            task_child=list_entry(list, struct task_struct, sibling);
            seq_printf(archivo, "      ---------------------------------------\n");
            seq_printf(archivo, "      PID:  \t %d \n", task_child->pid);
            seq_printf(archivo, "      Nombre: \t %s \n", task_child->comm);
            seq_printf(archivo, "      Estado: \t ");
            switch(task_child->state){
                case 0:
                    seq_printf(archivo, "Ejecutando\n");
                    break;
                case 1:
                    seq_printf(archivo, "Listo\n");
                    break;
                case 2:
                    seq_printf(archivo, "Durmiendo\n");
                    break;
                case 4:
                    seq_printf(archivo, "Zombie\n");
                    break;
                case 8:
                    seq_printf(archivo, "Detenido\n");
                    break;
                case 32:
                    seq_printf(archivo, "En Espera\n");
                    break;
                default:
                    seq_printf(archivo, "Desconocido\n");
                    break;
            }

        }

    }

    return 0;
}
```

Se declaran los eventos que se disparan al abrir o cerrar el archivo que definimos en el paso anterior, de esta forma nuestro módulo se ejecuta cada vez que esto suceda.

```
static int al_abrir(struct inode *inode, struct file *file) {
    return single_open(file, escribir_archivo, NULL);
}

static struct file_operations operaciones =
{
    .open = al_abrir,
    .read = seq_read
};

static int iniciar(void)
{
    proc_create("cpu_200915347_200915466", 0, NULL, &operaciones);
    printk(KERN_INFO "Carné: 200915347_200915466\n");
    return 0;
}

static void salir(void)
{
    remove_proc_entry("cpu_200915347_200915466", NULL);
    printk(KERN_INFO "Curso: Sistemas Operativos 1\n");
}

module_init(iniciar);
module_exit(salir);
```

Para la instalación del nuevo módulo nos dirigimos a la carpeta donde esta nuestro **Makefile** y el archivo C y a continuación ejecutamos en la terminal el comando **`make all`**.

```
nando@nando-so1:~/Documentos/proyecto1/cpu$ make all
make -C /lib/modules/5.3.0-28-generic/build M=/home/nando/Documentos/proyecto1/cpu modules
make[1]: se entra en el directorio '/usr/src/linux-headers-5.3.0-28-generic'
  CC [M]  /home/nando/Documentos/proyecto1/cpu/cpu_200915347_200915466.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/nando/Documentos/proyecto1/cpu/cpu_200915347_200915466.mod.o
  LD [M]  /home/nando/Documentos/proyecto1/cpu/cpu_200915347_200915466.ko
make[1]: se sale del directorio '/usr/src/linux-headers-5.3.0-28-generic'
nando@nando-so1:~/Documentos/proyecto1/cpu$
```

Ahora instalamos el módulo con el comando: **`sudo insmod cpu_200915347_200915466.ko`**

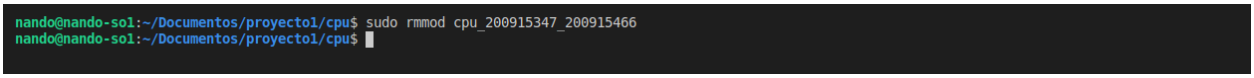
```
nando@nando-so1:~/Documentos/proyecto1/cpu$ sudo insmod cpu_200915347_200915466.ko
[sudo] contraseña para nando:
nando@nando-so1:~/Documentos/proyecto1/cpu$
```

Revisamos los mensajes del log con el comando: **`dmesg`**.

```
nando@nando-so1:~/Documentos/proyecto1/cpu$ dmesg
[    0.000000] Linux version 5.3.0-28-generic (buildd@lcy01-amd64-009) (gcc version 7.4.0 (Ubuntu 7.4.0-1ubuntu1~18.04.1)) #30~18.04.1-Ubuntu SMP Fri Jan 17 06:14:09 UTC 2020 (Ubuntu 5.3.0
-28.30~18.04.1-generic 5.3.13)
[    0.000000] Command line: BOOT_IMAGE=/vmlinuz-5.3.0-28-generic root=UUID=fce57b93-b390-437c-9e2b-3ad0f8f61b00 ro quiet splash
[    0.000000] KERNEL supported cpus:
[    0.000000]   Intel GenuineIntel
[    0.000000]   AMD AuthenticAMD
[    0.000000]   Hygon HygonGenuine
[    0.000000]   Centaur CentaurHauls
[    0.000000]   zhaoxin   Shanghai
[   36.576279] rfkill: input handler disabled
[  223.140753] Carné: 200915347_200915466
nando@nando-so1:~/Documentos/proyecto1/cpu$
```

Podemos ver que ha sido ejecutado con éxito al revisar el archivo creado en la carpeta proc con el comando: **`cat /proc/cpu_200915347_200915466`**

Desinstalamos el módulo con el comando: **`sudo rmmod cpu_200915347_200915466`**



Revisamos el log con el comando: **`dmesg`**.

Se remueven todos los archivos generados del módulo con el comando**: `make clean`**.