Módulo de CPU

1. Primero se deben de importar las librerías necesarias para poder interactuar con los diferentes procesos e informaciones del CPU

```c
#include <linux/proc_fs.h>
#include <linux/seq_file.h>
#include <linux/module.h>
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/sched/signal.h>
#include <linux/sched.h>
#include <linux/fs.h>
```

2. Posteriormente se deben de incluir las referencias a los struct que tienen la información de las tareas

```c

struct task_struct *task; //estructura definida en sched.h para tareas/procesos
struct task_struct *childtask; //estructura necesaria para iterar a travez de procesos secundarios
struct list_head *list; // estructura necesaria para recorrer cada lista de tareas tarea->estructura de hijos
```

3. Posteriormente se procede a escribir un archivo que nos servirá para poder ver los procesos que se tiene en nuestro sistema, para ello se utiliza `seq_printf` para poder escribir en el archivo con formato json, `for_each_process` para recorrer cada proceso existente

```c
static int escribir_archivo(struct seq_file * archivo, void *v){
    seq_printf(archivo,"{\n");
    for_each_process( task ){
        seq_printf(archivo,"{\n");
        seq_printf(archivo,"\"pid\": %d, \n",task->pid);
        seq_printf(archivo,"\"nombre\": \"%s\", \n",task->comm);
        seq_printf(archivo,"\"usuario\": \"root\", \n");
        seq_printf(archivo,"\"estado\": %ld, \n",task->state);
        seq_printf(archivo,"\"hijo\":\n");
        seq_printf(archivo,"\t[\n");

        list_for_each( list,&task->children ){
            seq_printf(archivo,"\t{\n");
            childtask= list_entry( list, struct task_struct, sibling );
            seq_printf(archivo,"\"pid\": %d, \n",childtask->pid);
            seq_printf(archivo,"\"nombre\": \"%s\", \n",childtask->comm);
            seq_printf(archivo,"\"usuario\": \"root\", \n");
            seq_printf(archivo,"\"estado\": %ld, \n",childtask->state);
            seq_printf(archivo,"\t},\n");
        }
        seq_printf(archivo,"\t]\n");
        seq_printf(archivo,"},\n");
    }
    seq_printf(archivo,"}\n");
    return 0;
}
```

4. Luego creamos una función que indica que al abrir se ejecutará la función de escribir_archivo

```
40
41  static int al_abrir(struct inode *inode, struct file*file){
42      return single_open(file, escribir_archivo, NULL);
43  }
```

5. De igual forma al creamos una función inicial que escribirá dentro de la consola del kernel los mismos procesos padres que podrán leerse con el comando dmseg para ello se utiliza el código `printk para poder escribir dentro de la consola del kernel`

```
51  int iniciar(void){ //modulo de inicio
52      proc_create("cpu_201503608", 0, NULL, &operaciones);
53      printk(KERN_INFO "%s", "CARGANDO MODULO");
54      printk(KERN_INFO "201503608 JUAN LUIS ROBLES MOLINA");
55
56      for_each_process( task ){
57          printk(KERN_INFO "PADRE DE PID: %d PROCESO: %s ESTADO: %ld",task->pid,task->comm,task->state);
58          list_for_each( list,&task->children ){
59              childtask= list_entry( list, struct task_struct, sibling );
60
61              printk(KERN_INFO "HIJO DE %s[%d]PID: %d PROCESO: %s ESTADO: %ld",task->comm,task->pid,childtask->pid, childtask->comm, childtask->state);
62          }
63          printk("*******************************************");
64      }
65      return 0;
66  }
```

6. Luego creamos una función de salida que nos servirá cuando nosotros removamos el módulo de la carpeta **/proc**

```
void salir(void){
    remove_proc_entry("cpu_201503608",NULL);
    printk(KERN_INFO "%s","REMOVIENDO MODULO");
    printk(KERN_INFO "Curso: sistemas operativos 1\n");
}
```

7. Luego indicaremos con qué funciones iniciaremos y finalizamos con este módulo, junto con la información de licencia y autor

```
74  module_init(iniciar);
75  module_exit(salir);
76
77  MODULE_LICENSE("GPL");
78  MODULE_AUTHOR("201503608");
```

Módulo de RAM

1. Primero se deben de importar las librerías necesarias para poder interactuar con la información de la RAM

```c
#include <linux/proc_fs.h>
#include <linux/seq_file.h>
#include <asm/uaccess.h>
#include <linux/hugetlb.h>
#include <linux/module.h>
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/fs.h>
```

2. De igual forma que con el cpu creamos un archivo que contiene la información de la RAM junto con los datos personales.

```c
static int escribir_archivo(struct seq_file * archivo, void *v){
    si_meminfo(&inf);
    long total_memoria =(inf.totalram * 4);
    long memoria_libre =(inf.freeram*4);
    seq_printf(archivo, "*****************\n");
    seq_printf(archivo, "*LABORATORIO DE SISTEMAS OPERATIVOS 1      *\n");
    seq_printf(archivo, "*DICIEMBRE 2020                           *\n");
    seq_printf(archivo, "*JUAN LUIS ROBLES MOLINA 201503608        *\n");
    seq_printf(archivo, "*PROYECTO 1 MODULO DE RAM                  *\n");
    seq_printf(archivo, "*****************\n");
    seq_printf(archivo, "Memoria Total : \t %8lu MB\n",total_memoria/1024);
    seq_printf(archivo, "Memoria libre : \t %8lu MB\n",memoria_libre/1024);
    seq_printf(archivo, "Memoria en uso : \t %8lu MB\n",(total_memoria-memoria_libre)/1024);
    seq_printf(archivo, "Porcentaje de Memoria : \t %8lu \n",((total_memoria-memoria_libre)/total_memoria*100));
    return 0;
}
```

3. Luego creamos una función que indica que al abrir se ejecutará la función de escribir_archivo

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

4. De igual forma al creamos una función inicial que escribirá dentro de la consola del kernel los mismos procesos padres que podrán leerse con el comando dmseg para ello se utiliza el código `printk` para poder escribir dentro de la consola del kernel

```c
static int iniciar(void){ //modulo de inicio
    proc_create("memo_201503608", 0, NULL, &operaciones);
    printk(KERN_INFO "%s", "CARGANDO MODULO");
    printk(KERN_INFO "201503608 JUAN LUIS ROBLES MOLINA");
    return 0;
}
```

5. Luego creamos una función de salida que nos servirá cuando nosotros removamos el módulo de la carpeta **/proc**

```
48
49  static void salir(void){
50      remove_proc_entry("memo_201503608",NULL);
51      printk(KERN_INFO "%s","REMOVIENDO MODULO");
52      printk(KERN_INFO "SISTEMAS OPERATIVOS 1\n");
53  }
54
```

6. Luego indicaremos con qué funciones iniciaremos y finalizamos con este módulo, junto con la información de licencia y autor

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

```
MODULE_LICENSE("GPL");
MODULE_AUTHOR("201503608");
```