UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERIA
ESCUELA DE CIENCIAS Y SISTEMAS
ING.

SISTEMAS OPERATIVOS
SECCIÓN
A

MANUAL TECNICO

NOMBRE:	CARNET:
Edgar Orlando Guamuch Zárate	**201314632**
Marcos Alberto Santos Aquino	**201700328**
Jurgen Andoni Ramirez Ramirez	**201404179**

Contenido

REQUISITOS DEL SISTEMA:

Requisitos Mínimos

- Procesador 700 MHz
- 512 GB de memoria RAM
- 10 GB Disco Duro
- Puerto USB para la instalación

Lenguajes Utilizados

- Go 1.13
- C GCComplier

Módulo de RAM

Para obtener la información de la memoria RAM, se debe consultar por medio de la función my_proc_show, el cual es cargado a través de la función si_mimfo(). Esta función recibe como parámetro un apuntador a un struct, y lo rellena con la información de la memoria. A continuación se muestra cada una de las propiedades del struct previamente mencionado.

```c
static int my_proc_show(struct seq_file *m, void *v) {
    long memoria_total = 0;
    long memoria_libre = 0;
    long memoria_usada = 0;

    struct sysinfo si;
    si_meminfo(&si);

    memoria_total = (si.totalram * 4);
    memoria_libre = (si.freeram * 4);
    memoria_usada = memoria_total - memoria_libre;
    seq_printf(m, "{\"ram_total\": %8lu", (memoria_total));
    seq_printf(m, ", \"ram_free\": %8lu", (memoria_libre));
    seq_printf(m, ", \"ram_usage\": %8lu", (memoria_usada));

    seq_printf(m, ", \"ram_usage_percent\":%8lu }", (memoria_usada*100/memoria_total));
    return 0;
}
```

Para utilizar las librerías necesarias en el módulo de memoria RAM se debe importar de la siguiente manera:
#include <nombre_libreria>.

Librerias

Nombre	**Descripción**
linux/proc_fs.h	Librería que proporcional as funciones para poder crear y escribir un archivo en el directorio /proc.
seq_file.h	Librería que proporciona una función que agrega un buffer al archivo en el directorio /proc.
module.h	Librería que proporciona lo necesario para definir el módulo como uno módulo de Linux.
init.h	Librería que proporciona los macros para iniciar un módulo, además de macros para finalizarlo.
mm.h	Librería que se utiliza para la manipulación de áreas de memoria. Provee la función get_mm_rss(), la cual calcula la memoria de cada proceso.

Inicialización del Módulo:

Es la función por la cual se inicializa la ejecución del módulo creado, es decir, es la principal del módulo, y desde la cual se procede a llamar al resto de código que se ejecutara. Esta función es llamada al realizar la carga del módulo por primera vez a través del comando insmod. Esta función también es la encargada de crear el fichero en el /proc.

```
static int __init test_init(void) {
    struct proc_dir_entry *entry;
    entry = proc_create("mem_grupo19", 0777, NULL, &my_fops);
    if(!entry) {
        printk(KERN_INFO "adios_grupo19");
        return -1;
    } else {
        printk(KERN_INFO "Hola mundo, somos el grupo grupo19 y este es el monitor de memoria\n");
    }
    return 0;
}
```

Salida de Módulo

Esta función se encarga de limpiar la memoria que fue utilizada por el módulo, esta es ejecutada al realizar la descarga del módulo. Aquí es donde se elimina el archivo creado en /proc.

```
static void __exit test_exit(void) {
    remove_proc_entry("mem_grupo19", NULL);
    printk(KERN_INFO ""Sayonara mundo, somos el grupo grupo19 y este fue el monitor de memoria \n");
}
```

Para cada una de las funciones previamente mencionadas es necesario llamarlas al final del archivo de C de la siguiente manera.

```
module_init(test_init);
module_exit(test_exit);
```

Escritura del Archivo en la Ruta /proc

La función a continuación es llamada cada vez que se desea obtener la información de la memoria RAM, esta se guardará en un archivo en la ruta */proc*.

```
static int my_proc_show(struct seq_file *m, void *v) {
    long memoria_total = 0;
    long memoria_libre = 0;
    long memoria_usada = 0;

    struct sysinfo si;
    si_meminfo(&si);

    memoria_total = (si.totalram * 4);
    memoria_libre = (si.freeram * 4);
    memoria_usada = memoria_total - memoria_libre;
    seq_printf(m, "{\"ram_total\": %8lu", (memoria_total));
    seq_printf(m, ", \"ram_free\": %8lu", (memoria_libre));
    seq_printf(m, ", \"ram_usage\": %8lu", (memoria_usada));

    seq_printf(m, ", \"ram_usage_percent\":%8lu }", (memoria_usada*100/memoria_total));
    return 0;
}
```

La siguiente función se encarga de abrir el archivo en la ruta /proc y retorna el inodo que identifica el archivo.

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

Módulo de Procesos

Para utilizar las librerías necesarias en el módulo de memoria RAM se debe importar de la siguiente manera:
#include <nombre_libreria>.

Librerias

Nombre	**Descripción**
linux/proc_fs.h	Librería que proporcional as funciones para poder crear y escribir un archivo en el directorio /proc.
linux/seq_file.h	Librería que proporciona una función que agrega un buffer al archivo en el directorio /proc.
asm/uaccess.h	Librería que proporciona el acceso del usuario.
linux/hugetlb.h	Librería que permite utilizar páginas de memoria virtual de gran tamaño.
linux/module.h	Librería que proporciona lo necesario para definir el módulo como uno módulo de Linux.
linux/init.h	Librería que proporciona los macros para iniciar un módulo, además de macros para finalizarlo.
linux/kernel.h	Librería que indica el módulo del Kernel.
linux/fs.h	Librería que proporciona las estructuras de apoyo para manejar.
linux/sched.h	Librería que proporciona la estructura para tareas o procesos.
linux/sched/signal.h	Libreria que proporciona Macros para realizar iteraciones.
linux/mm.h	Librería que se utiliza para la manipulación de áreas de memoria. Provee la función get_mm_rss(), la cual calcula la memoria de cada proceso.

Estructuras Importantes

Las siguientes estructuras se utilizaron para acceder a las tareas o procesos y de igual manera para acceder a los hijos.

```
struct task_struct *task_list;
struct task_struct *task_list_child;
struct list_head *list;
```

Escritura del Archivo en la Ruta /proc

La función a continuación es llamada cada vez que se desea obtener la información de los procesos, esta se guardará en un archivo en la ruta */proc*.

```
static int escribir_a_proc(struct seq_file * file_proc, void *v) {

    seq_printf(file_proc, "[");
    for_each_process(task_list) {
            seq_printf(file_proc,"{\"pid\":%d,\"nombre\":\"%s\",\"estado\":%ld,\"ram\":%ld,\"usuario\":%u,\"hijos\":[", task_list->pid,task_list->comm,task_list->state,(task_l
            list_for_each(list, &task_list->children){
                task_list_child = list_entry( list, struct task_struct, sibling );
                seq_printf(file_proc,"{\"pid\":%d,\"nombre\":\"%s\",\"estado\":%ld,\"ram\":%ld,\"usuario\":%u},",task_list_child->pid, task_list_child->comm, task_list_child->
            }
            seq_printf(file_proc, "]},");
    }
    seq_printf(file_proc, "]");
    return 0;
}
```

La siguiente función se encarga de abrir el archivo en la ruta /proc y retorna el inodo que identifica el archivo.

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

Inicialización de Módulo

Es la función por la cual se inicializa la ejecución del módulo creado, es decir, es la principal del módulo, y desde la cual se procede a llamar al resto de código que se ejecutara. Esta función es llamada al realizar la carga del módulo por primera vez a través del comando insmod. Esta función también es la encargada de crear el fichero en el /proc.

```
static int __init cpu_grupo19_init(void)
{
    proc_create("cpu_grupo19", 0, NULL, &archivo_operaciones);
    printk(KERN_INFO "cpu_grupo19\n");
    printk(KERN_INFO "Sayonara mundo, somos el grupo 19 y este fue el monitor cpu and process\n");

    return 0;
}
```

Salida de Módulo

Esta función se encarga de limpiar la memoria que fue utilizada por el módulo, esta es ejecutada al realizar la descarga del módulo. Aquí es donde se elimina el archivo creado en /proc.

```
static void __exit cpu_grupo19_cleanup(void)
{
    remove_proc_entry("cpu_grupo19", NULL);
    printk(KERN_INFO "Sistemas Operativos 2 - Grupo 19\n");
    printk(KERN_INFO "Sayonara mundo, somos el grupo 19 y este fue el monitor cpu and process\n");
}
```

Para cada una de las funciones previamente mencionadas es necesario llamarlas al final del archivo de C de la siguiente manera.

```
module_init(cpu_grupo19_init);
module_exit(cpu_grupo19_cleanup);
```