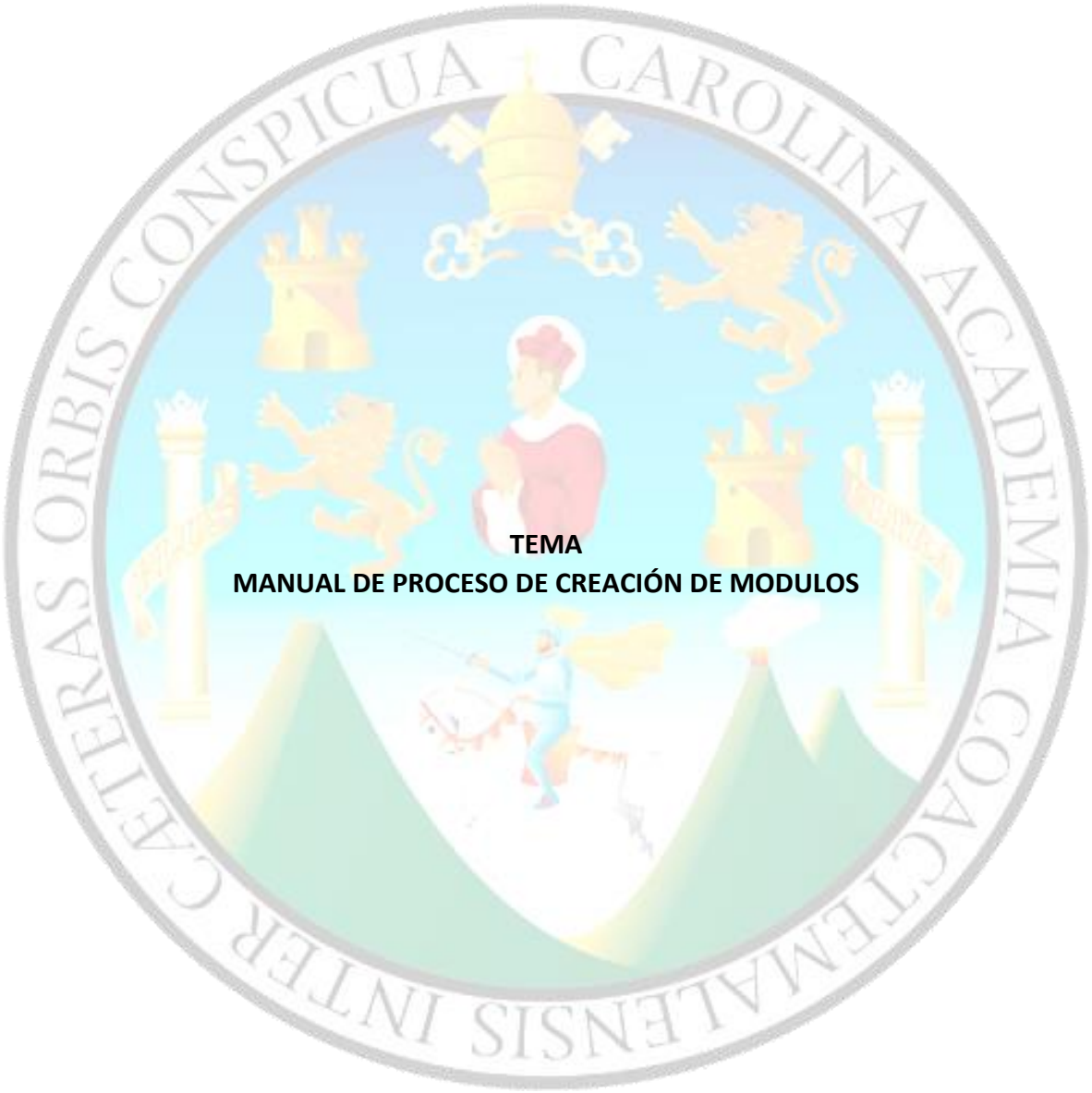
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERIA
ESCUELA DE CIENCIAS Y SISTEMAS
SISTEMAS OPERATIVOS 1
ING. SERGIO MÉNDEZ
AUX. BRYAN ORDOÑEZ



TEMA

MANUAL DE PROCESO DE CREACIÓN DE MODULOS

CESAR JAVIER SOLARES OROZCO --- 201313819
ERICK ALEXANDER LEMUS MORALES --- 201612097
FECHA DE ENTREGA: 15 DE JUNIO DE 2020

INDICE

TEMA

¿QUÉ ES UN MÓDULÑO DE KERNEL?

Un módulo del kernel es un fragmento de código o binarios que pueden ser cargado y eliminados del kernel según las necesidades de este. Tienen el objetivo de extender sus funcionalidades son fragmentos de código que pueden ser cargados y eliminados del núcleo bajo demanda. Extienden la funcionalidad del núcleo sin necesidad de reiniciar el sistema.

Esto es gracias a que el kernel tiene un diseño modular, cuando se instala un nuevo componente o se inicia la computadora los módulos son cargados de forma dinámica para que funcionen de forma transparente.

Módulo de Kernel de Memoria

- **Librerias necesarias:** a continuación se lista las librerías que son necesarias para el correcto funcionamiento de este módulo
  - <linux/proc_fs.h>
  - <linux/seq_file.h>
  - <asm/uaccess.h>
  - <linux/hugetlb.h>
  - <linux/module.h>
  - <linux/init.h>
  - <linux/kernel.h>
  - <linux/fs.h>
- **Struct Sysinfo:** es una estructura que manejan los sitemas linux para el control de la información de la memoria del sistema.
  - **A continuación la estructura.**

```
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 */
```

};

- **Funciones a utilizar:**
  - Write_file
  - Abrir
  - Iniciar
  - Salir
  - Module_init
  - Module_exit
- **Creando el módulo de kernel con código de C. con el siguiente contenido**

```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>

#define BUFSIZE     150

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Información de memoria");
MODULE_AUTHOR("Erick Lemus - 201612097\nJavier Solares - 201313819");

struct sysinfo inf;

static int write_file(struct seq_file * archivo, void *v){
    si_meminfo(&inf);
    long memoriatotal=(inf.totalram*4);
    long memorialibre=(inf.freeram*4);
    seq_printf(archivo,"\n");
    seq_printf(archivo,"      -------------------------------------\n");
    seq_printf(archivo,"      |   PROYECTO 1 - MODULO DE MEMORIA  |\n");
    seq_printf(archivo,"      | LABORATORIO SISTEMAS OPERATIVOS 1 |\n");
    seq_printf(archivo,"      |           JUNIO 2020              |\n");
    seq_printf(archivo,"      -------------------------------------\n");
    seq_printf(archivo,"\n");
    seq_printf(archivo,"      CESAR JAVIER SOLARES OROZCO - 201313819\n");
    seq_printf(archivo,"     ERICK ALEXANDER LEMUS MORALES - 201612097\n");
    seq_printf(archivo,"\n");
    seq_printf(archivo,"-----------------------------------------------\n");
    seq_printf(archivo,"| TOTAL MEMORIA  | %8lu kb - %8lu mb |\n", memoriatotal, memoriatotal/1024
);
    seq_printf(archivo,"| MEMORIA LIBRE  | %8lu kb - %8lu mb |\n", memorialibre, memorialibre/1024
);
    seq_printf(archivo,"| MEMORIA EN USO | %i %%                     |\n", (memorialibre * 100)/m
emoriatotal);
    seq_printf(archivo,"-----------------------------------------------\n");
    seq_printf(archivo,"\n");
    return 0;
}

static int abrir(struct inode  *inode, struct file *file){
    return single_open(file,write_file, NULL);
}

static struct file_operations ops =
```

```
    {
        .open=abrir,
        .read=seq_read
        /* data */
    };

    static int iniciar(void){
        proc_create("memo_201313819_201612097",0, NULL,&ops);
        printk("\nCarnet1: 201313819, Carnet2: 201612097\n");
        return 0;
    }

    static void salir(void){
        remove_proc_entry("memo_201313819_201612097",NULL);
        printk("\nSistemas Operativos 1\n");
    }

    module_init(iniciar);
    module_exit(salir);

```

- **Archivo Makefile:** es un archivo que crea el modulo de kernel a partir del comando make. Debemos crear un archivo llamada Makefile con el siguiente código

```
obj-m += memo_201313819_201612097.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 para ejecutar el modulo de memoria.**
    - make /*para ejecutar el archivo Makefile*/
    - Se crearan multiples archivos, debe crearse uno con la extensión .ko
    - sudo insmod miarchivo.ko /*inserta el modulo de kernel*/
    - cat myarchivo /*para ver el contenido de la ejecución del archivo de kernel*/
    - sudo rmmod miarchivo.ko /*para remover el modulo de kernel*/

Módulo de Kernel de CPU

Este modulo es utilizado para verificar los procesos que actualmente se están ejecutando dentro de nuestra máquina.

- **Librerias necesarias:**
  - <linux/kernel.h>
  - <linux/module.h>
  - <linux/init.h>
  - <linux/sched/signal.h>
  - <linux/sched.h>
  - <linux/list.h>
  - <linux/types.h>
  - <linux/slab.h>
  - <linux/string.h>
  - <linux/fs.h>
  - <linux/seq_file.h>
  - <linux/proc_fs.h>
  - <linux/mm.h>
- **Task_struct:** es una estructura que contiene la información completa de todos los proceso existentes dentro del sistema. La estructura es bastante extensa y contiene mucha información.
- **Funciones a utilizar:**
  - Iterate_init
  - Write_file
  - Abrir
  - Iniciar
  - Salir
  - Module_init
  - Module_exit
- **Creando el módulo de kernel con código de C. con el siguiente contenido**

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/init.h>
#include <linux/sched/signal.h>
#include <linux/sched.h>

#include <linux/list.h>
#include <linux/types.h>
#include <linux/slab.h>
#include <linux/string.h>
#include <linux/fs.h>
#include <linux/seq_file.h>
#include <linux/proc_fs.h>
#include <linux/mm.h>

```

```c
    struct task_struct *task;
    struct task_struct *task_child;
    struct list_head *list;

    MODULE_LICENSE("GPL");
    MODULE_DESCRIPTION("CPU");
    MODULE_AUTHOR("Javier Solares 201313819 --- Erick Lemus 201612097");

    int iterate_init(struct seq_file *archivo) /*    Init Module    */
    {

        seq_printf(archivo, "------------------------------------------------------------------------
    OBTENIENDO PROCESOS-------------------------------------------------------------------------\n");
        for_each_process(task)
        {
            char estadoq = 79; //otro estado
            if (task->state == TASK_RUNNING)
            {
                estadoq = 82;
            }
            else if (task->state == __TASK_STOPPED)
            {
                estadoq = 83;
            }
            else if (task->state == TASK_INTERRUPTIBLE)
            {
                estadoq = 73;
            }
            else if (task->state == TASK_UNINTERRUPTIBLE)
            {
                estadoq = 85;
            }
            else if (task->exit_state == EXIT_ZOMBIE)
            {
                estadoq = 90;
            }
            else if (task->state == TASK_DEAD)
            {
                estadoq = 68;
            }
            seq_printf(archivo, "\n\n\t\t| PID PROCESO ACTUAL: %d \t NOMBRE: %s \t ESTADO: %c |\n", ta
    sk->pid, task->comm, estadoq);
            int actual=1;
            list_for_each(list, &task->children)
            {
                task_child = list_entry(list, struct task_struct, sibling);

                char estado = 79; //otro estado
                if (task_child->state == TASK_RUNNING)
                {
                    estado = 82;
                }
                else if (task_child->state == __TASK_STOPPED)
                {
                    estado = 83;
                }
                else if (task_child->state == TASK_INTERRUPTIBLE)
                {
                    estado = 73;
                }
                else if (task_child->state == TASK_UNINTERRUPTIBLE)
```

```
                {
                    estado = 85;
                }
                else if (task_child->exit_state == EXIT_ZOMBIE)
                {
                    estado = 90;
                }
                else if (task_child->state == TASK_DEAD)
                {
                    estado = 68;
                }
                seq_printf(archivo, "\n%d) PPROCESO_PADRE: %s PID_PROCESO_PADRE%d || PID_HIJO: %d NOMB
    RE: %s ESTADO: %c \n", actual, task->comm, task->pid, task_child->pid, task_child->comm, estado);
                actual++;
            }
            seq_printf(archivo, "\n\n-----------------------------------------------------------------------
    ------------------------------------------------------------------------------------\n");
        }

        return 0;
    }

    static int write_file(struct seq_file *archivo, void *v)
    {
        seq_printf(archivo, "\n");
        seq_printf(archivo, "      -------------------------------------\n");
        seq_printf(archivo, "      |     PROYECTO 1 - MODULO DE CPU    |\n");
        seq_printf(archivo, "      | LABORATORIO SISTEMAS OPERATIVOS 1 |\n");
        seq_printf(archivo, "      |           JUNIO 2020              |\n");
        seq_printf(archivo, "      -------------------------------------\n");
        seq_printf(archivo, "\n");
        seq_printf(archivo, "      CESAR JAVIER SOLARES OROZCO - 201313819\n");
        seq_printf(archivo, "     ERICK ALEXANDER LEMUS MORALES - 201612097\n");
        seq_printf(archivo, "\n");
        seq_printf(archivo, "      -------------------------------------\n");
        seq_printf(archivo, "      |              ESTADOS              |\n");
        seq_printf(archivo, "      |  TASK_RUNNING         =        R  |\n");
        seq_printf(archivo, "      |  TASK_INTERRUPTIBLE   =        I  |\n");
        seq_printf(archivo, "      |  TASK_UNINTERRUPTIBLE =        U  |\n");
        seq_printf(archivo, "      |  TASK_STOPPED         =        S  |\n");
        seq_printf(archivo, "      |  TASK_ZOMBIE          =        Z  |\n");
        seq_printf(archivo, "      |  OTRO_TASK            =        0  |\n");
        seq_printf(archivo, "      -------------------------------------\n");
        seq_printf(archivo, "\n");
        return iterate_init(archivo);
    }

    static int abrir(struct inode *inode, struct file *file)
    {
        return single_open(file, write_file, NULL);
    }

    static struct file_operations ops =
        {
            .open = abrir,
            .read = seq_read

    };

    static int iniciar(void)
    {
```

```
    proc_create("cpu_201313819_201612097", 0, NULL, &ops);
    printk(KERN_INFO "\nCarnet1: 201313819, Carnet2: 201612097\n");
    return 0;
}

static void salir(void)
{
    remove_proc_entry("cpu_201313819_201612097", NULL);
    printk(KERN_INFO "\nSistemas Operativos 1\n");
}

void cleanup_exit(void)
{

    printk(KERN_INFO "%s", "REMOVING MODULE\n");

}

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

- **Archivo Makefile:** es un archivo que crea el modulo de kernel a partir del comando make. Debemos crear un archivo llamada Makefile con el siguiente código

```
obj-m += cpu_201313819_201612097.o
KDIR ?= /lib/modules/$(shell uname -r)/build

all:
    make -C $(KDIR) M=$(PWD) modules

clean:
    make -C $(KDIR) M=$(PWD) clean
```

- **Comandos para ejecutar el modulo de memoria.**
    - make /*para ejecutar el archivo Makefile*/
    - Se crearan multiples archivos, debe crearse uno con la extensión .ko
    - sudo insmod miarchivo.ko /*inserta el modulo de kernel*/
    - cat myarchivo /*para ver el contenido de la ejecución del archivo de kernel*/
    - sudo rmmod miarchivo.ko /*para remover el modulo de kernel*/