**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 modulo de kernel?
* Modulo de Kernel de Memoria
  + Librerias necesarias
  + Struct sysinfo
  + Funciones a utilizar
  + Archivo Makefile
  + Comandos para ejecutar módulo de kernel
* Módulo de Kernel de CPU
  + Librerias necesarias
  + Struct task
  + Funciones a utilizar
  + Archivo Makefile
  + Comandos para ejecutar módulo de kernel

**¿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**
* #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)/memoriatotal);
* 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>
* 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", task->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 NOMBRE: %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\*/