Universidad San Carlos de Guatemala
Facultad de Ingeniería
Curso Laboratorio Sistemas Operativos 1
Auxiliar Bernald Paxtor

Manual de Creación de Módulos

Juan Pablo Rojas Chinchilla
201900289
Vacaciones Diciembre 2021

Creación del módulo de memoria

Para la creación del módulo de memoria RAM se utilizó la “struct” sysinfo, la cual nos da información del sistema.
Para la obtención de la memoria total se utilizó el atributo “totalram” de esta estructura antes mencionada, para la obtención de la memoria libre se utilizaron tres atributos, estos fueron: “freeram”, “sharedram” y “bufferram”, para conseguir el dato de la ram libre debe restarse la memoria compartida y la memoria de buffer, a la memoria libre, con esto nos dará el dato crudo de memoria libre, para luego sumarle la memoria caché.
En cuanto a la recuperación de los datos se utilizó el formato json, ya que este facilita la obtención de objetos, ya que de esta manera pueden obtenerse todos sus atributos sin ninguna complicación.
Por último, tanto para el montaje como para el desmontaje de este módulo se escribe un mensaje, el cual tiene la función de mostrar cuando se monta y cuando se desmonta este módulo, para el inicio y final de este módulo se utilizaron los metodos: “module_init” y “module_exit”.

Creación del módulo de CPU

Para la creación del módulo de CPU se utilizaron dos estructuras principales, estas fueron: “task_struct” y “list_head”, además de estas estructuras, se utilizó la librería <linux/mm.h>, esta fue utilizada para la obtención de la memoria ram utilizada por cada proceso.
La obtención de los procesos y sus atributos se realizó con un ciclo, el cual recorría todos los ciclos, obteniendo tanto sus atributos como el nombre y id de sus procesos hijos, el único cálculo matemático realizado fue para la obtención de la memoria utilizada por cada proceso.
Luego de la obtención de todos los atributos de los procesos se separaron y agruparon la cantidad de procesos que pertenecían a “en ejecución”, “dormidos”, “zombies”, “detenidos” y “totales”.
Por último al igual que en el otro módulo se mostraron unos mensajes tanto al montar como al desmontar el módulo.

Código fuente

Memoria RAM

1. Obtención de datos sobre la memoria.

```c
static int proc_ram_data(struct seq_file * file, void *v){
    si_meminfo(&info);
    unsigned long totalRam = (info.totalram*4);
    unsigned long freeRam = (info.freeram*4)-(info.sharedram*4)-(info.bufferram*4);
    seq_printf(file, "{\n");
    seq_printf(file,"\"total_memory\": %lu,\n",totalRam/1024);
    seq_printf(file,"\"free_memory\": %lu,\n", freeRam/1024);
    seq_printf(file,"\"used_memory\": %lu\n", ((totalRam - freeRam)*100)/totalRam);
    seq_printf(file, "}\n");
    return 0;
}
```

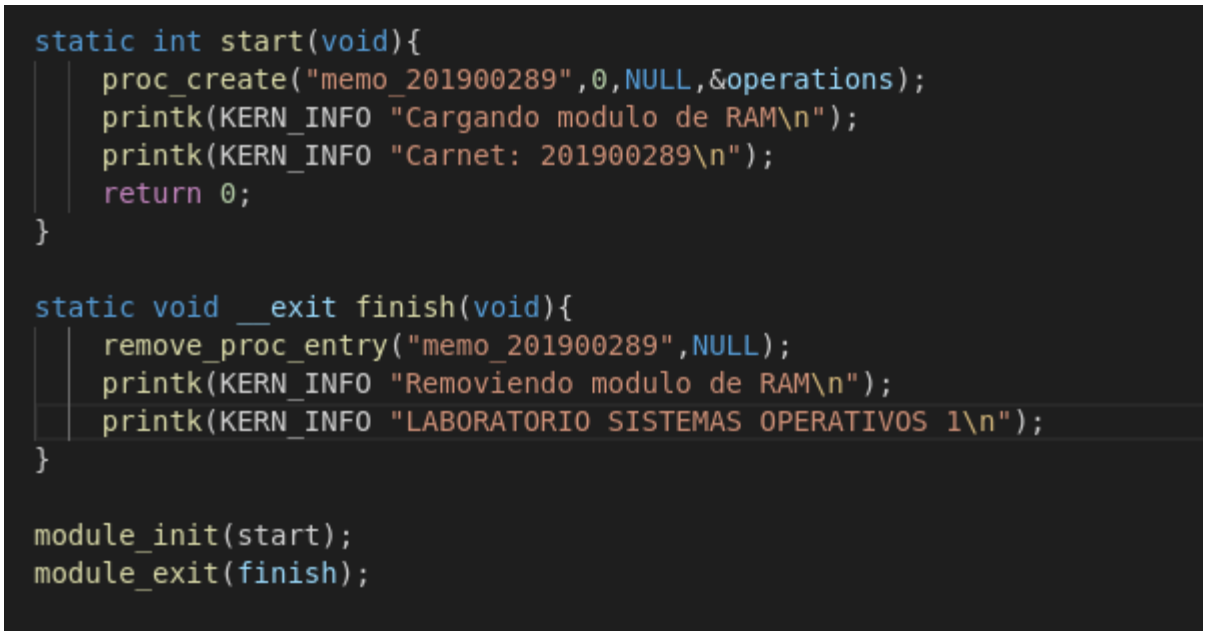
CPU

1. Ciclos para obtención de procesos

```
#ifndef CONFIG_MMU
    pr_err("No MMU, cannot calculate RSS.\n");
    return -EINVAL;
#endif

    seq_printf(file, "{\n\"processes\":[\n");
    int b = 0;
    for_each_process(task)
    {
        if (task->mm)
        {
            rss = get_mm_rss(task->mm) << PAGE_SHIFT;
        }
        else
        {
            rss = 0;
        }
        if (b == 0)
        {
            seq_printf(file, "{");
            b = 1;
        }
        else
        {
            seq_printf(file, ",{");
        }
        seq_printf(file, "\"pid\":%d,\n", task->pid);
        seq_printf(file, "\"name\":\"%s\",\n", task->comm);
        seq_printf(file, "\"user\": %d,\n", task->cred->uid);
        seq_printf(file, "\"state\":%ld,\n", task->__state);
        int porcentaje = (((rss / (1024 * 1024))) * 100) / (15685);
        seq_printf(file, "\"ram\":%d,\n", porcentaje);

        seq_printf(file, "\"child\":[\n");
        int a = 0;
        list_for_each(list, &(task->children))
        {
            taskChild = list_entry(list, struct task_struct, sibling);
            if (a != 0)
            {
                seq_printf(file, ",{");
                seq_printf(file, "\"pid\":%d,\n", taskChild->pid);
                seq_printf(file, "\"name\":\"%s\"\n", taskChild->comm);
                seq_printf(file, "}\n");
            }
            else
            {
                seq_printf(file, "{");
                seq_printf(file, "\"pid\":%d,\n", taskChild->pid);
                seq_printf(file, "\"name\":\"%s\"\n", taskChild->comm);
                seq_printf(file, "}\n");
                a = 1;
            }
        }
        a = 0;
        seq_printf(file, "\n]");

        if (task->__state == 0)
        {
            running += 1;
        }
        else if (task->__state == 1)
        {
            sleeping += 1;
        }
        else if (task->__state == 4)
        {
            zombie += 1;
        }
        else
        {
            stopped += 1;
        }
        seq_printf(file, "}\n");
    }
    b = 0;
    seq_printf(file, "],\n");
    seq_printf(file, "\"running\":%d,\n", running);
    seq_printf(file, "\"sleeping\":%d,\n", sleeping);
    seq_printf(file, "\"zombie\":%d,\n", zombie);
    seq_printf(file, "\"stopped\":%d,\n", stopped);
    seq_printf(file, "\"total\":%d\n", running + sleeping + zombie + stopped);
    seq_printf(file, "}\n");
    return 0;
}
```

General

1. Inicio y fin mostrando mensajes



```c
static int start(void){
    proc_create("memo_201900289",0,NULL,&operations);
    printk(KERN_INFO "Cargando modulo de RAM\n");
    printk(KERN_INFO "Carnet: 201900289\n");
    return 0;
}

static void __exit finish(void){
    remove_proc_entry("memo_201900289",NULL);
    printk(KERN_INFO "Removiendo modulo de RAM\n");
    printk(KERN_INFO "LABORATORIO SISTEMAS OPERATIVOS 1\n");
}

module_init(start);
module_exit(finish);
```