

Sistemas Operativos 2

Segundo Semestre 2019

Práctica 1

Manual de Usuario

Jhosef Omar Cáceres Aguilar

201513696



Módulos en linux

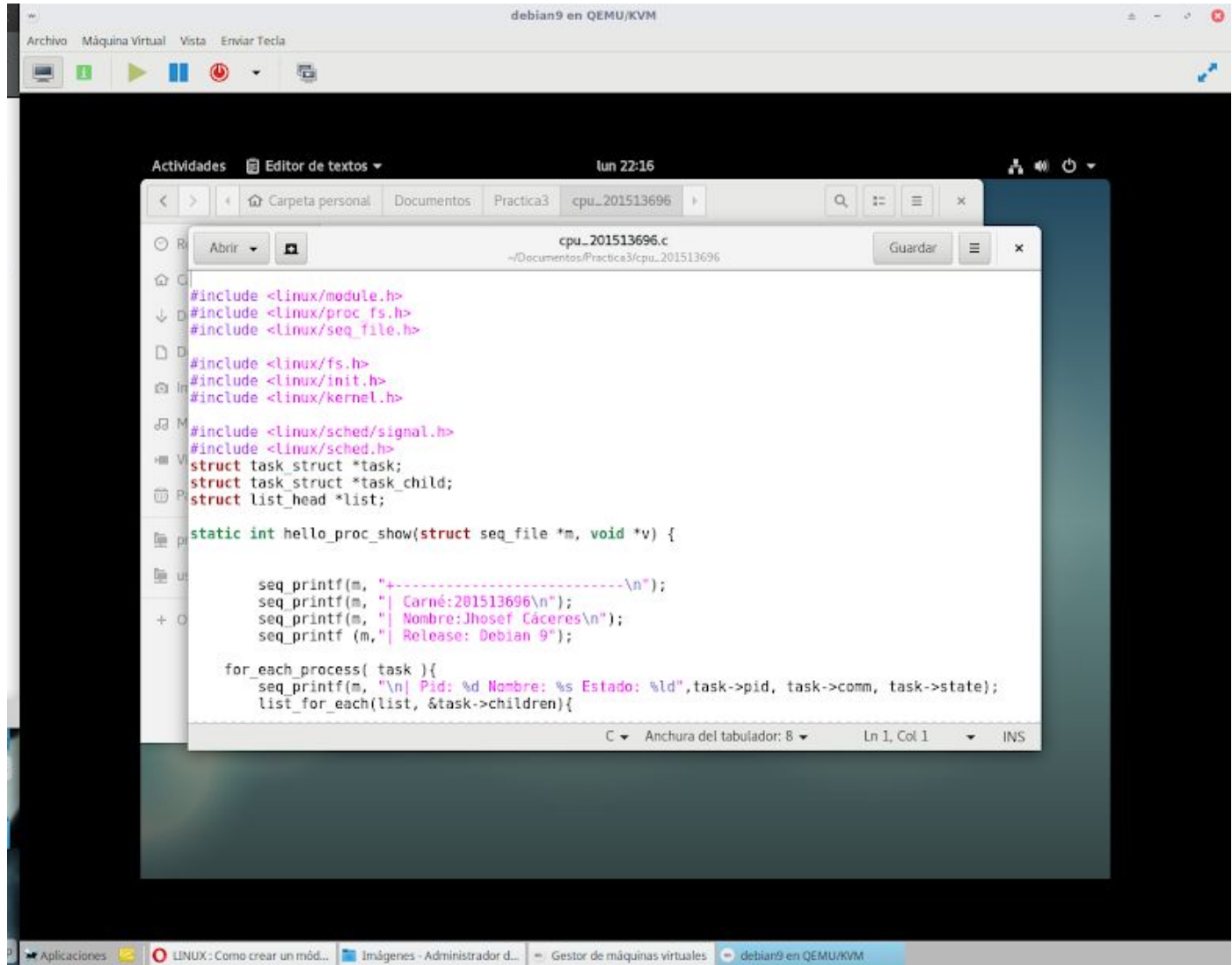
Carga dinámica de módulos

El kernel de Linux es modular ya que permite la inserción y eliminación dinámica de código en el kernel en tiempo de ejecución.

- Las subrutinas asociadas, datos, puntos de entrada y salida son agrupadas en una única imagen binaria llamada módulo del kernel.
- Para permitir esto, el kernel debe estar compilado con la opción `CONFIG_MODULES`
- Cuando se agrega una nueva funcionalidad al kernel siempre está la pregunta; Como módulo ? O estático en el kernel ?
- Muchas veces se tienden agregar funcionalidades como módulos ya que se pueden cargar a demanda.
- Los módulos son guardados en el filesystem como archivos objeto (con extensión `.ko`) y son linkeados con el kernel con el programa `insmod`.
- El sistema define la estructura `struct module` para representar los módulos.
- Cada objeto `module` describe un módulo.
- Linux mantiene una lista doblemente encadenada que agrupa todos los objetos `module`.
- La cabeza de la lista es guardada en la variable `modules` mientras los punteros a los elementos adyacentes se guardan en el campo `list` de cada `module`.

Proceso de creación de módulos

Se escribe un archivo en c.



```
#include <linux/module.h>
#include <linux/proc_fs.h>
#include <linux/seq_file.h>

#include <linux/fs.h>
#include <linux/init.h>
#include <linux/kernel.h>

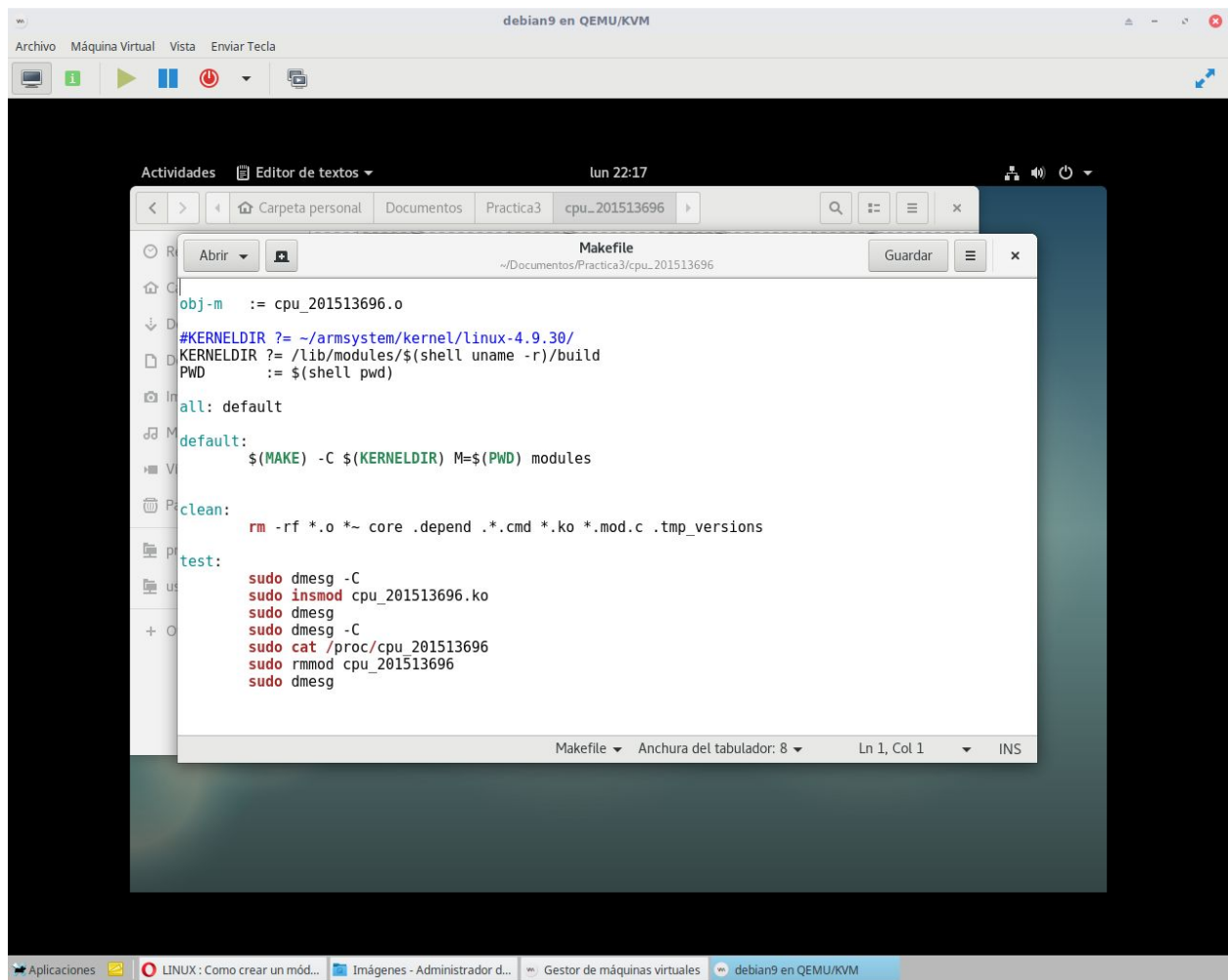
#include <linux/sched/signal.h>
#include <linux/sched.h>
struct task_struct *task;
struct task_struct *task_child;
struct list_head *list;

static int hello_proc_show(struct seq_file *m, void *v) {

    seq_printf(m, "-----\n");
    seq_printf(m, "Carné:201513696\n");
    seq_printf(m, "Nombre:Jhosef Cáceres\n");
    seq_printf(m, "Release: Debian 9");

    for_each_process( task ){
        seq_printf(m, "\n| Pid: %d Nombre: %s Estado: %ld", task->pid, task->comm, task->state);
        list_for_each(list, &task->children){
```

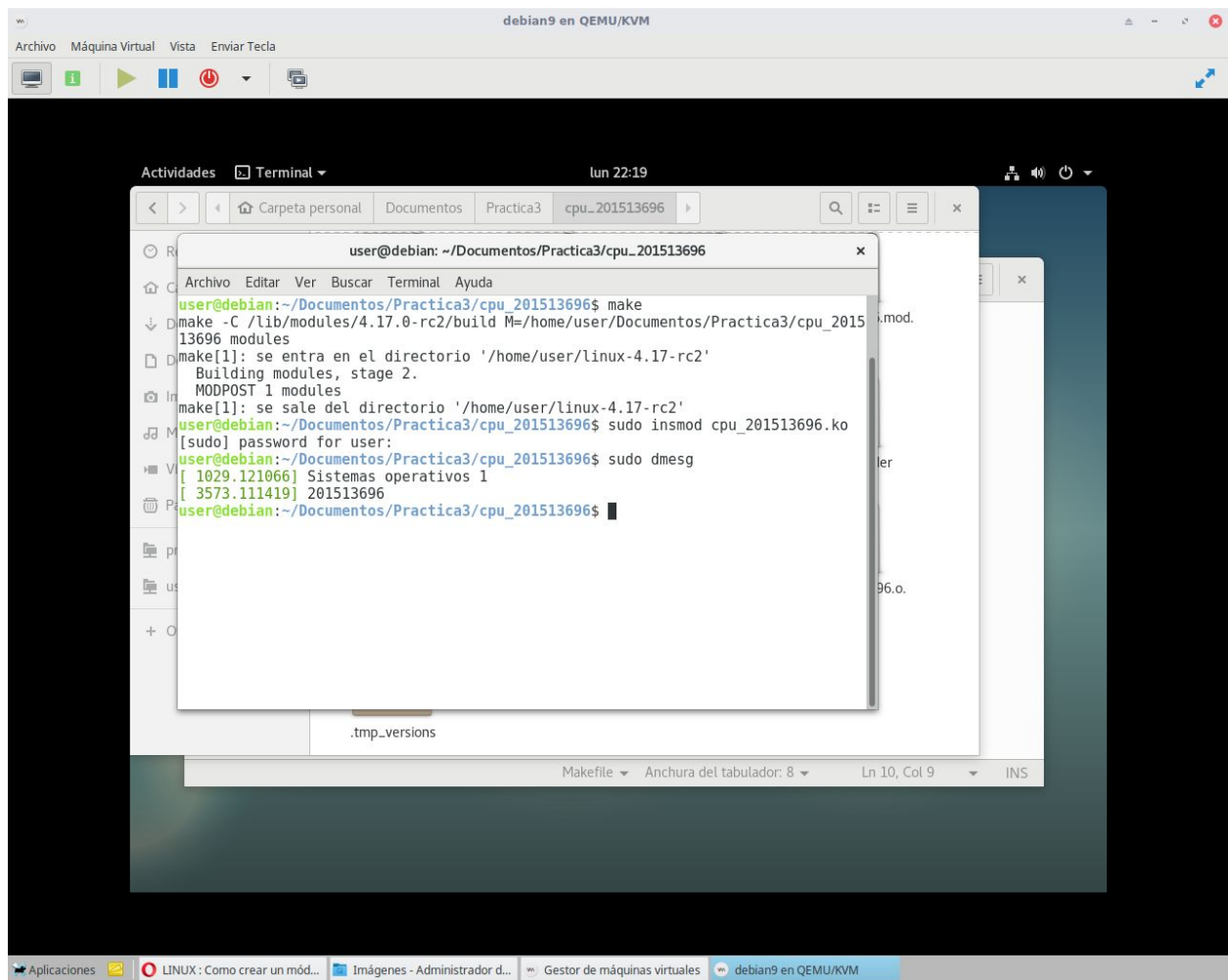
Luego se crea un archivo MakeFile



Se utiliza el comando `make` para compilar el módulo

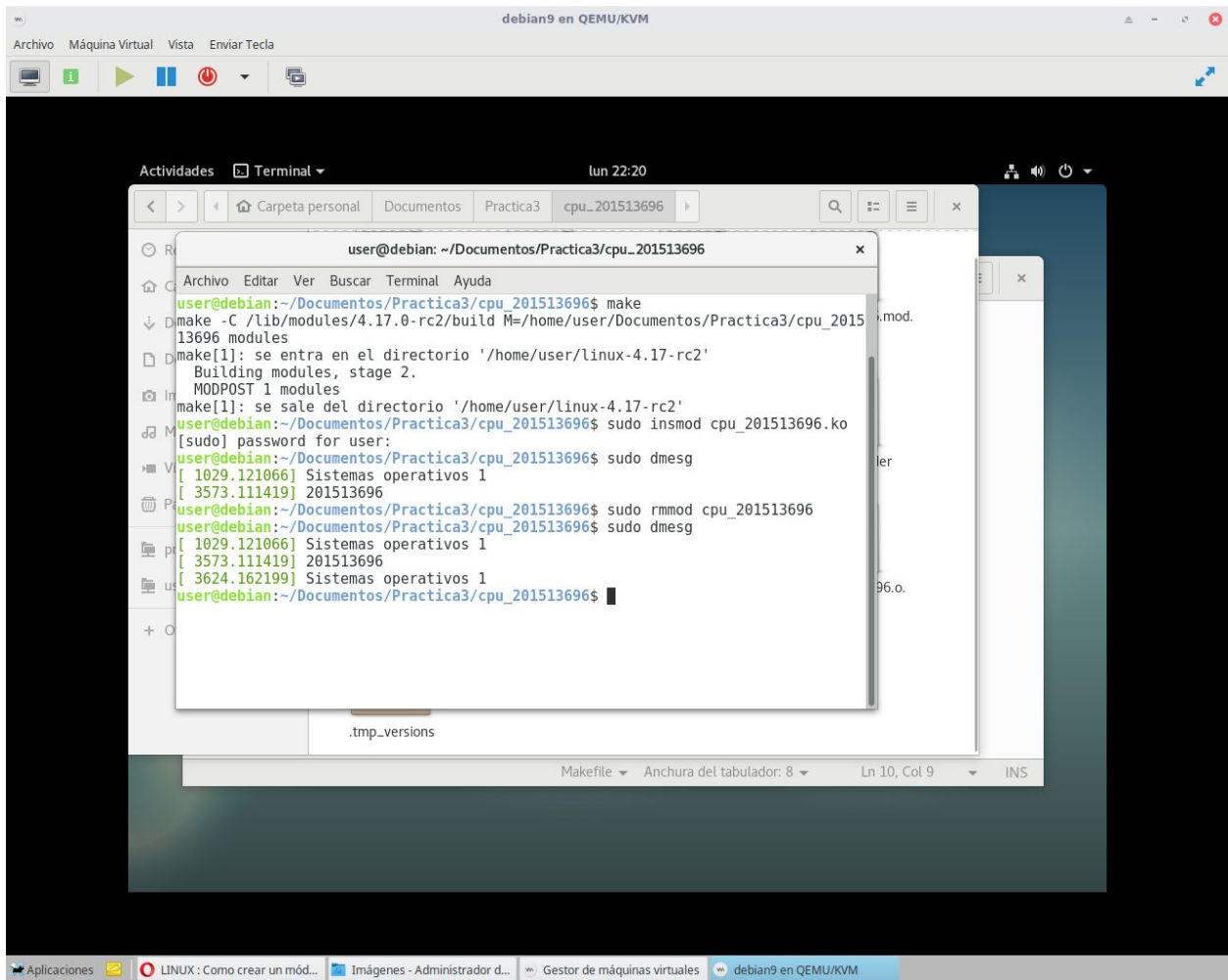
El comando `dmesg -C` para imprimir el log del módulo al cargarlo.

Para cargar el módulo ya compilado se usa el comando `insmod`



Ahora para quitarlo se usa el comando.

rmmod



Struct usados para obtener información

Estructura para la memoria RAM

```
struct sysinfo {
```

```
    unsigned long loads[3]; /* Segundos desde el arranque */
```

```
    unsigned long totalram; /* Promedios de carga de 1, 5 y 15 minutos */
```

```

unsigned long freeram; /* Tamaño de memoria principal utilizable */
unsigned long sharedram; /* Tamaño de memoria disponible */
unsigned long bufferram;; /* Cantidad de memoria compartida */
bufferram largo sin firmar; /* Memoria utilizada por buffers */
unsigned long totalswap; /* Tamaño total del espacio de intercambio */
unsigned long freeswap; /* espacio de intercambio todavía disponible */
unsigned short procs; /* Número de procesos actuales */
char _f[22]; /* Pads estructura a 64 bytes */
};

```

Estructura para los procesos.

La estructura de datos task_struct describe un proceso o tarea en el sistema.

```

struct task_struct {
    /* estos están codificados, no toques */

    estado largo volátil; /* -1 no ejecutable, 0 ejecutable, > 0 detenido */

    largo contador

    largo prioridad

    señal largo sin firmar;

    sin firmar largo bloqueado; /* mapa de bits de señales enmascaradas */

    banderas largas sin firmar; /* por banderas de proceso, definidas a continuación */

    int errno

    debugreg largo [8]; /* Registros de depuración de hardware */

```

```

struct exec_domain * exec_domain;

/* varios campos */

struct linux_binfmt * binfmt;

struct task_struct * next_task, * prev_task;

struct task_struct * next_run, * prev_run;

unsigned long saved_kernel_stack;

sin firmar largo kernel_stack_page;

int exit_code, exit_signal;

/* ??? */

larga personalidad sin firmar;

volcado int: 1;

int did_exec: 1;

int pid;

int pgrp;

int tty_old_pgrp;

int sesión;

/* valor booleano para el líder del grupo de sesión */

int líder;

grupos int [NGROUPS];

/*

* Indicadores al proceso original (padre), hijo menor, hermano menor,

* hermano mayor, respectivamente. (p-> padre puede ser reemplazado por

* p-> p_pptr-> pid)

* /

```



```

struct task_struct * p_opptr, * p_pptr, * p_cptr,
    * p_ysptr, * p_osptr;

struct wait_queue * wait_chldexit;

uid corto sin firma, euid, suid, fsuid;

gid corto sin firmar, egid, sgid, fsgid;

sin tiempo de espera largo, política, rt_priority;

sin signo largo it_real_value, it_prof_value, it_virt_value;

sin signo largo it_real_incr, it_prof_incr, it_virt_incr;

struct timer_list real_timer;

largo utime, stime, cutime, cstime, start_time;

/* mm error e información de intercambio: esto puede discutirse como
   mm-específico o hilo-específico */

sin firmar long min_flt, maj_flt, nswap, cmin_flt, cmaj_flt, cnsnap;

int swappable: 1;

swap_address largo sin firmar;

sin firmar long_maj_flt; /* antiguo valor de maj_flt */

sin firmar largo dec_flt; /* página cuenta de fallos de la última vez */

swap_cnt largo sin firmar; /* número de páginas a intercambiar en la próxima pasada */

/* límites */

struct rlimit rlim [RLIM_NLIMITS];

unsigned short used_math;

char comm [16];

/* información del sistema de archivos */

int link_count;

```

```

struct tty_struct * tty; /* NULL si no hay tty */

/* ipc stuff */

struct sem_undo * semundo;

struct sem_queue * semsleeping;

/* Idt para esta tarea - utilizado por Wine. Si es NULL, se usa default_idt */

struct desc_struct * Idt;

/* tss para esta tarea */

struct thread_struct tss;

/* información del sistema de archivos */

struct fs_struct * fs;

/* Abrir información del archivo */

struct files_struct * files;

/* información de gestión de memoria */

struct mm_struct * mm;

/* manejadores de señales */

struct signal_struct * sig;

#ifdef __SMP__

procesador int

int last_processor;

int lock_depth; /* Profundidad de bloqueo.

Podemos contextualizar el cambio dentro y fuera

de mantener un bloqueo del núcleo del syscall ... */

#terminara si

};

```

—