Taller de Sistemas Operativos

Módulos del kernel

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 los programadores tienden agregar funcionalidades como módulos ya que se pueden cargar a demanda.

Módulos

- El *kernel* debe realizar dos grandes tareas con respecto a los módulos:
  - Asegurarse que el resto del *kernel* pueda acceder a los símbolos globales del módulo y que el módulo pueda acceder a los símbolos del *kernel*. Las referencias son resueltas en el momento de carga del módulo.
  - Llevar la cuenta del uso de módulos para que no se pueda descargar un módulo mientras otra parte del *kernel* lo está usando.

Implementación de módulos

- Los módulos son guardados en el *filesystem* como archivos objeto (con extensión .ko) y son linkeditados 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.

Implementación de módulos

- El campo `state` puede tener los siguientes valores:
  - MODULE_STATE_LIVE – El módulo está activo.
  - MODULE_STATE_COMING – El módulo se está inicializando.
  - MODULE_STATE_GOING – El módulo se está eliminando.
- Cada módulo tiene un conjunto de contadores (uno por CPU) almacenado en el campo `ref`.
- El contador es incrementado cada vez que una función del módulo se inicializa y decrementado cada vez que finaliza.
- Un módulo puede ser descargado una vez que la suma de los contadores de uso sea 0.

Implementación de módulos

- Cuando se carga un módulo, todas las referencias a símbolos del *kernel* deben ser resueltas. Esto lo realiza el comando `insmod`.
- Algunas tablas de símbolos especiales dentro del *kernel* son usadas para guardar los símbolos accedidos por módulos junto con sus correspondientes direcciones.
- Los módulos cargados pueden exportar sus símbolos así otros módulos pueden accederlos.
- Un módulo A puede utilizar símbolos de un módulo B. Para cargar el módulo A, B ya debe estar cargado.
- El campo `modules_which_use_me` del objeto module correspondiente a A es la cabeza de la lista de dependencias de A. Esta lista debe actualizarse dinámicamente cuando se carga un módulo que usa A.
- El módulo A no puede descargarse si su lista de dependencias no es vacía.

Carga de módulos del kernel

- En la carga del módulo se realizan las siguientes operaciones:
  1. Lee de la línea de comando el nombre de módulo a cargar.
  2. Busca el archivo que implementa el módulo (/lib/modules).
  3. Lee de disco el archivo que contiene el código del módulo.
  4. Invoca a la `syscall init_module()` pasándole:
     - El buffer de usuario conteniendo el código objeto del módulo.
     - El largo del código.
     - El área de memoria conteniendo los parámetros (user mode).
  5. Termina.

- La mayor parte del trabajo real es `sys_init_module()`.

Carga de módulos del kernel

1. Verifica que el usuario tenga permisos de cargar módulos.
2. Reserva memoria temporal para los *buffers* pasados como parámetro a la *syscall* (y copia la información).
3. Busca en la `modules_list` que el módulo no se encuentre ya cargado.
4. Reserva memoria para la porción ejecutable del código y el área de inicialización del mismo (y copia la información).
5. Guarda en los campos `module_code` y `module_init` del objeto module las direcciones de memoria cargadas en (4).
6. Inicializa la lista `modules_which_use_me` y los contadores de uso en cero salvo el de la CPU que ejecuta este código (1).
7. Usando las *kernel symbol tables* y las *module symbol tables* resuelve las referencias en el código del módulo.

Carga de módulos del kernel

8. Carga los valores de las variables según los argumentos que recibe en el comando `insmod`.
9. Libera las áreas temporales de memoria (paso 2).
10. Agrega el objeto module a la `modules_list`.
11. Carga el estado del módulo en MODULE_STATE_COMING.
12. Si se definió, ejecuta el método `init` del objeto module.
13. Carga el estado del módulo a MODULE_STATE_LIVE.
14. Termina retornando cero (éxito).

Descarga de módulos del kernel

- Un usuario puede descargar un módulo mediante el comando `rmmod`.

Este realiza las siguientes operaciones:

1. Lee de la línea de comando el nombre de módulo a descargar.
2. Abre el archivo `/proc/modules` donde se listan todos los módulos del *kernel* y busca que efectivamente se encuentre cargado.
3. Invoca a la `syscal delete_module()` pasándole el nombre del módulo.
4. Termina.

La mayor parte del trabajo real es `sys_delete_module()`.

Descarga de módulos del kernel

1. Verifica que el usuario tenga permisos para descargar el módulo.
2. Copia el nombre del módulo en un buffer del *kernel*.
3. Busca el objeto module en la lista modules.
4. Verifica las dependencias. Si `module_which_use_me` no es vacía retorna un código de error.
5. Verifica el estado del módulo. Si no es MODULE_STATE_LIVE retorna un código de error.
6. Verifica que el módulo tenga un método de init personalizado. Si es así debe tener un método de `exit` personalizado. De no ser así se retorna con código de error.
7. Para evitar “race conditions” se detienen las actividades en todos los CPU’s excepto el que ejecuta `sys_delete_module()`.
8. Carga el estado del módulo a MODULE_STATE_GOING.
9. Si la suma de los contadores de referencia es mayor que cero, retorna un código de error.

Descarga de módulos del kernel

10. Se ejecuta el método de `exit` del módulo.
11. Se da de baja el objeto module de la lista `modules`
12. Se remueve el objeto `module` de la lista de dependencias de los módulos que estaban usando.
13. Libera la memoria usada por el módulo (objeto module, código, etc.).
14. Retorna cero (éxito).

Carga de módulos bajo demanda

- Un módulo puede ser cargado automáticamente cuando la funcionalidad que provee es requerida y descargado automáticamente al finalizar.
- Para cargar un módulo automáticamente, el *kernel* crea un *kernel thread* para ejecutar el binario `modprobe` que toma en cuenta las posibles complicaciones debido a la dependencia de módulos.
- El comando `modprobe` es similar a `insmod` pero carga en forma recursiva todos los módulos dependientes.
- Otra herramienta se llama `depmod`. Es ejecutada durante el boot del sistema. Busca los módulos compilados para el *kernel* (usualmente en `/lib/modules`). Luego escribe las dependencias en el archivo `modules.dep`.

Implementación de módulos

```
#include <linux/init.h>
#include <linux/module.h>
MODULE_LICENSE("Dual BSD/GPL");

static int hello_init(void) {
    printk(KERN_ALERT "Hello, world\n");
    return 0;
}


static void hello_exit(void) {
    printk(KERN_ALERT "Goodbye, cruel world\n");
}
module_init(hello_init);
module_exit(hello_exit);
```