Práctica 2: Llamadas al sistema

LIN - Curso 2018-2019

J.C. Sáez
High Performance Computing Division
ArTeCs Group
Universidad Complutense de Madrid

Contenido

Contenido

Práctica 2: Llamadas al sistema

Objetivos

- Familiarizarse con:
  - Implementación de llamadas al sistema en Linux y su procedimiento de invocación
  - Compilación del kernel Linux

Contenido

Ejercicios

Ejercicio 1

- Estudiar la implementación del programa cpuinfo.c
  - Este programa imprime por pantalla el contenido de `/proc/cpuinfo` haciendo uso de las llamadas al sistema `open()` y `close()`, y las funciones `printf()` y `syscall()`.
  - ¿Qué llamada al sistema invoca el programa mediante `syscall()`?
  - Reescribir el programa anterior reemplazando las llamadas a `open()`, `close()` y `printf()` por invocaciones a `syscall()` que tengan el mismo comportamiento.

Ejercicios

- La entrada `/proc/cpuinfo` permite obtener información acerca de las CPUs del sistema

Terminal

```
kernel@debian:~$ cat /proc/cpuinfo
processor   : 0
vendor_id   : GenuineIntel
cpu family  : 6
model       : 23
model name  : Intel(R) Xeon(R) CPU           E5450  @ 3.00GHz
stepping    : 10
cpu MHz     : 2003.000
cache size  : 6144 KB
physical id : 0
siblings    : 4
core id     : 0
cpu cores   : 4
apicid      : 0
initial apicid  : 0
fpu     : yes
fpu_exception   : yes
cpuid level : 13
wp      : yes
flags       : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts
...
```

Ejercicios

Ejercicio 2

- Analizar la implementación del módulo del kernel `modleds.c` que interactúa con el driver de teclado de un PC para encender/apagar los LEDs
  - Al cargar el módulo se encienden los tres leds del teclado y al descargarlo se apagan
- Advertencia: No usar una shell SSH para cargar el módulo. Usar una ventana de terminal en la propia máquina virtual.

Ejercicios

Ejercicio 2 (cont.)

- Se ha de prestar especial atención a las siguientes funciones:
  - `get_kbd_driver_handler()`: Se invoca durante la carga del módulo para obtener un puntero al manejador del driver de teclado/terminal
  - `set_leds(handler,mask)`: Permite establecer el valor de los leds. Acepta como parámetro un puntero al manejador del driver y una máscara de bits que especifica el estado de cada LED.

Ejercicios

Ejercicio 2 (cont.)

- Significado de la máscara de bits de `set_leds()` (parámetro `mask`)
  - bit 0: scroll lock ON/OFF
  - bit 1: num lock ON/OFF
  - bit 2: caps lock ON/OFF
  - bit 3-31: se ignoran
- En cada bit..
  - Si 1 $\rightarrow$ LED ON
  - Si 0 $\rightarrow$ LED OFF

Contenido

Partes de la práctica

(Parte A.) Crear llamada al sistema "Hola Mundo" (`lin_hello`)

- Seguir instrucciones del tema "Llamadas al Sistema"
- Enseñar funcionamiento al profesor en el laboratorio
- Crear parche del kernel con las modificaciones realizadas

(Parte B.) Implementar llamada al sistema `ledctl()`

1. La llamada permitirá que los programas de usuario puedan encender/apagar los LEDs del teclado
   - Exige modificar el kernel para incluir llamada al sistema `ledctl()`
2. Además se ha de implementar el programa de usuario `ledctl_invoke` que permita invocar la llamada al sistema desde terminal

Especificación de `ledctl()` (I)

Llamada al sistema `ledctl()`

```
long ledctl(unsigned int leds);
```

- **Parámetro**: Máscara de bits que especifica qué LEDs se encenderán/apagarán
- **Valor de retorno**: 0 en caso de éxito; -1 en caso de fallo
  - Advertencia: La implementación en sí de la llamada (kernel) devolverá un número negativo que codifica el error
    - En caso de error la función `syscall()` devolverá -1 al programa de usuario, y el código de error quedará almacenado en la variable global `errno`

Especificación de `ledctl()` (II)

Formato parámetro ledctl

- `ledctl()` acepta como parámetro una máscara de bits que especifica qué LEDs se encenderán/apagarán:
  - bit 2 → encender/apagar *Num Lock*
  - bit 1 → encender/apagar *Caps Lock*
  - bit 0 → encender/apagar *Scroll Lock*

Parámetro de ledctl()	Num Lock	Caps Lock	Scroll Lock
0x4	ON	OFF	OFF
0x7	ON	ON	ON
0x3	OFF	ON	ON
0x0	OFF	OFF	OFF
0x2	OFF	ON	OFF

Programa `ledctl_invoke`

- Para llevar a cabo la depuración de la llamada al sistema se desarrollará un programa de usuario para invocarla desde terminal
  - En caso de que `ledctl()` devuelva un error, el programa mostrará el error correspondiente con `perror()`
- Modo de uso:

  ```
  $ ledctl_invoke <comando_ledctl>
  ```

  - Ejemplo: $ ./ledctl_invoke 0x4
    - invocará `ledctl(0x4);`
  - `ledct_invoke.c` se puede compilar fácilmente como sigue:

    ```
    $ gcc -Wall -g ledctl_invoke.c -o ledct_invoke
    ```

Implementación Parte B (I)

- La implementación de la llamada al sistema requiere modificar el kernel
  - Por cada fallo detectado:
    1. Modificar código del kernel
    2. Compilar y reinstalar kernel
    3. Reiniciar la máquina
- Se aconseja usar un módulo del kernel auxiliar para depurar el código de la llamada al sistema antes de introducirla en el kernel
  - Por ejemplo, el módulo de depuración podría exportar una entrada /proc de sólo escritura que permita modificar el estado de los leds al escribir en ella
  - `sudo echo 0x6 > /proc/ledctl`

Implementación Parte B (II)

Pasos a seguir

1. Realizar modificaciones pertinentes en el código del kernel
2. Compilar el kernel modificado
3. Instalar paquetes (*image* y *headers*) en la máquina virtual y reiniciar
4. Probar código usando programa `ledctl_invoke` (a desarrollar)
5. Si fallo, ir a 1. En otro caso, hemos acabado :-)

Implementación Parte B (III)

- Al definir la llamada al sistema dentro del kernel, se debe utilizar la macro `SYSCALL_DEFINE1()`

```
#include <linux/syscalls.h> /* For SYSCALL_DEFINEi() */
#include <linux/kernel.h>

SYSCALL_DEFINE1(ledctl,unsigned int,leds)
{
    /* Cuerpo de la función */

    return 0;
}
```

Parte B: Ejemplo de ejecución

- Arrancar la MV con el kernel modificado con `ledctl()` y abrir una ventana de terminal...

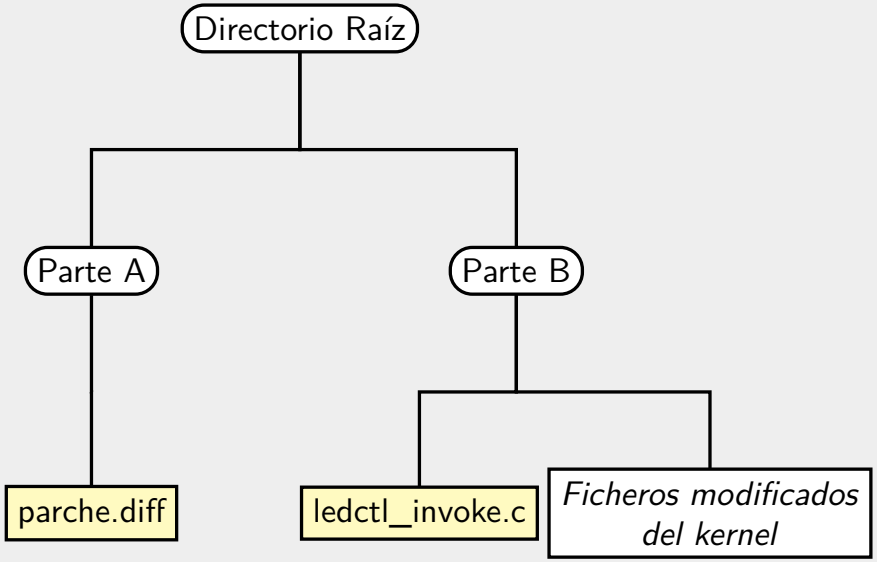
terminal

```
kernel@debian:p2$ gcc -g -Wall ledctl_invoke.c -o ledctl_invoke
kernel@debian:p2$ ./ledctl_invoke
Usage: ./ledctl_invoke <ledmask>
kernel@debian:p2$ sudo ./ledctl_invoke 0x6
<< Se deberían encender los dos LEDs de más a la izquierda>>
kernel@debian:p2$ sudo ./ledctl_invoke 0x1
<< Se debería encender solamente el LED de la derecha >>
kernel@debian:p2$
```

Entrega de la práctica

- A través del Campus Virtual
  - Hasta el 26 de octubre
- Obligatorio mostrar el funcionamiento de la práctica en clase

Estructura entrega (en un fichero comprimido .tar.gz o .zip)



Licencia

LIN - Práctica 2: Llamadas al sistema
Versión 1.1

Este documento (o uno muy similar) está disponible en
`https://cv4.ucm.es/moodle/user/index.php?id=105108`