INGENIERÍA UNIVERSIDAD DE LOS ANDES MÉRIDA VENEZUELA | DEPARTAMENTO DE COMPUTACIÓN ESCUELA DE SISTEMAS

Laboratorio de Sistemas Operativos
Semestre A-2017
Práctica Uno
Laboratorio

Prof. Rodolfo Sumoza
Prep. Alvaro Araujo

1. Llamadas al Sistema

Las llamadas al sistema son interfaces que el sistema operativo proporciona para poder utilizar los servicios y utilidades que el mismo proporciona.

Desde del punto de vista de la programación las llamadas al sistema poseen un papel importante, cualquier programa puede usar intensamente servicios del sistema operativo. Normalmente cuando se desarrolla un programa se utiliza una API (*Application Programming Interface*, interfaz de programación de aplicaciones), y en la mayoría de los casos no se presta atención a cómo las funciones de las API interactúan con el sistema operativo. Conociendo cómo funcionan las llamadas al sistema se podría programar de una manera más eficiente, cabe destacar que los sistemas ejecutan miles de llamadas al sistema cada segundo.

En este laboratorio, para hacer un estudio cuantitativo de las llamadas al sistema se utilizará la herramienta ***strace***, que entre otras cosas permite monitorizar las llamadas al sistema realizadas por un programa durante su ejecución. Esta herramienta para su funcionamiento utiliza a su vez la utilidad ***ptrace*** (*process trace*) contenida en los kernel de las distribuciones Linux.

Para ilustrar el uso de la herramienta y consolidar los conceptos, se desarrollarán algunos ejercicios.

1.1. Ejercicio

Desarrolle dos programas en C, que van a cumplir el mismo objetivo, copiar el contenido de un archivo a un nuevo archivo, uno de ellos debe realizarlo copiando línea por línea y el otro debe realizarlo copiando carácter por carácter.

Para ejecutar un programa utilizando ***strace***:

Se puede añadir la bandera `-l` para tener un resumen de las llamadas realizadas.

Ahora se deben ejecutar ambos programas y analizar (en cuanto al tiempo de ejecución y el número de las llamadas al sistema) las salidas que se obtuvieron, ¿qué pasa si en vez de copiar el contenido de un archivo a otro, esté se muestra por pantalla? ¿se obtendrán mayor número de llamadas al sistema?

1.2. Ejercicio

Desarrolle dos programas en C, uno de ellos que reserve memoria al comenzar la ejecución para un arreglo de `100000000` posiciones, el valor de la posición `n` será `n+500`, luego modifique cada una de las posiciones y coloque como nuevo valor `vector[i]+500` y por último muestre en pantalla el valor de la última posición del arreglo. El segundo programa debe cumplir la misma función que el primero, con la diferencia que la memoria que se reserva al comenzar es sólo para una posición y luego a medida que se inserta el valor de los `100000000` elementos se va re-dimensionando.

Ahora se deben ejecutar ambos programas y analizar las salidas que se obtuvieron, ¿los resultados obtenidos fueron los esperados? Modifique el primer programa que realizó para que en el bucle donde se modifica el arreglo también se debe mostrar cada valor por pantalla, ¿hubo un cambio significativo en el tiempo de ejecución (mayor al 10 %)?

2. Módulos del Kernel Linux

En esta parte de la práctica se estudiará cómo crear y montar un módulo del kernel. Analicemos el siguiente código que pertenece a un módulo sencillo:

```
1  #include <linux/init.h>
2  #include <linux/module.h>
3  #include <linux/kernel.h>
4
5  /* Cargar el modulo */
6  int simple_init(void)
7  {
8    printk(KERN_INFO "Cargando el Modulo :)\n");
9    return 0;
10 }
11
12 /* Remover el modulo */
13 void simple_exit(void)
14 {
15   printk(KERN_INFO "Removiendo el Modulo :(\n");
16 }
17
18 /* Punto de entrada y salida */
19 module_init( simple_init );
20 module_exit( simple_exit );
21 /* Standard info */
22 MODULE_LICENSE("GPL");
23 MODULE_DESCRIPTION("Primer Modulo SO ULA A-2016");
24 MODULE_AUTHOR("ULA SO");
```

El módulo del kernel se carga con el comando:

```
root@pc:/# insmod primermodulo.ko
```

Para comprobar si el módulo se ha cargado, introduzca el comando:

```
root@pc:/# lsmod
```

Para ver el contenido del buffer de los mensajes del kernel, se ejecuta:

```
root@pc:/# dmesg
```

Para remover el módulo, se ejecuta:

```
root@pc:/# rmmod primermodulo
```

Para limpiar el buffer de los mensajes del kernel se ejecuta:

```
root@pc:/# dmesg -c
```