

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS



Práctica 7

Organización y Arquitectura de Computadoras

Johann Ramón Gordillo Guzmán - 418046090

José Jhovan Gallardo Valdés - 310192815

Diana Laura Nicolás Pavia - 314183093

Práctica presentada como parte del curso de **Organización y Arquitectura de Computadoras** impartido por el profesor **José de Jesús Galaviz Casas**.

23 de Octubre del 2019

Link al código fuente: <https://github.com/JohannGordillo>

1. Preguntas

1. Investiga y describe con tus propias palabras, ¿cómo resuelve una llamada al sistema el sistema operativo?

El sistema operativo es el encargado de gestionar los recursos de una computadora.

En general, un usuario común hace uso de estos recursos a través de programas que no forman parte del sistema operativo; Firefox, LibreOffice, Emacs, etc. Lo anterior sugiere que existe una forma de comunicación entre dichos programas y el sistema operativo.

Dado que el procesador es el encargado de realizar las tareas que manejan a una computadora, dentro del procesador existen dos modos de ejecución; modo usuario y modo kernel.

Modo usuario:

Dentro de este modo se lleva a cabo la ejecución de una aplicación, a menos que ésta necesite acceder a algún recurso de la computadora, podríamos decir que este modo provee un entorno seguro de interacción ya que en ningún momento se corre el riesgo de modificar o acceder a información que no esté permitida.

Modo kernel:

Dentro de este modo, es posible hacer uso y/o modificar los recursos de la computadora como crear archivos, modificar un documento, escribir en el disco de la computadora, eliminar información, iniciar una aplicación, entre otras cosas.

Cuando una aplicación está ejecutándose usualmente se necesitará acceso a la memoria o al hardware, las llamadas al sistema son aquellas que solicitan el permiso de usar este recurso. Cuando se hace dicha llamada, se le indica al procesador que la ejecución de la aplicación debe detenerse por un instante y pasar a otra subrutina, generalmente a los servicios del modo kernel, a esta situación se le llama *interrupcion*, si la llamada al sistema es válida ésta inicia un proceso en modo kernel que es el que finalmente hará uso del sistema operativo, ejecutará lo necesario y proveerá el recurso que se le ha solicitado, una vez terminada la tarea el control es devuelto a la aplicación en modo usuario y es posible seguir usando la aplicación.

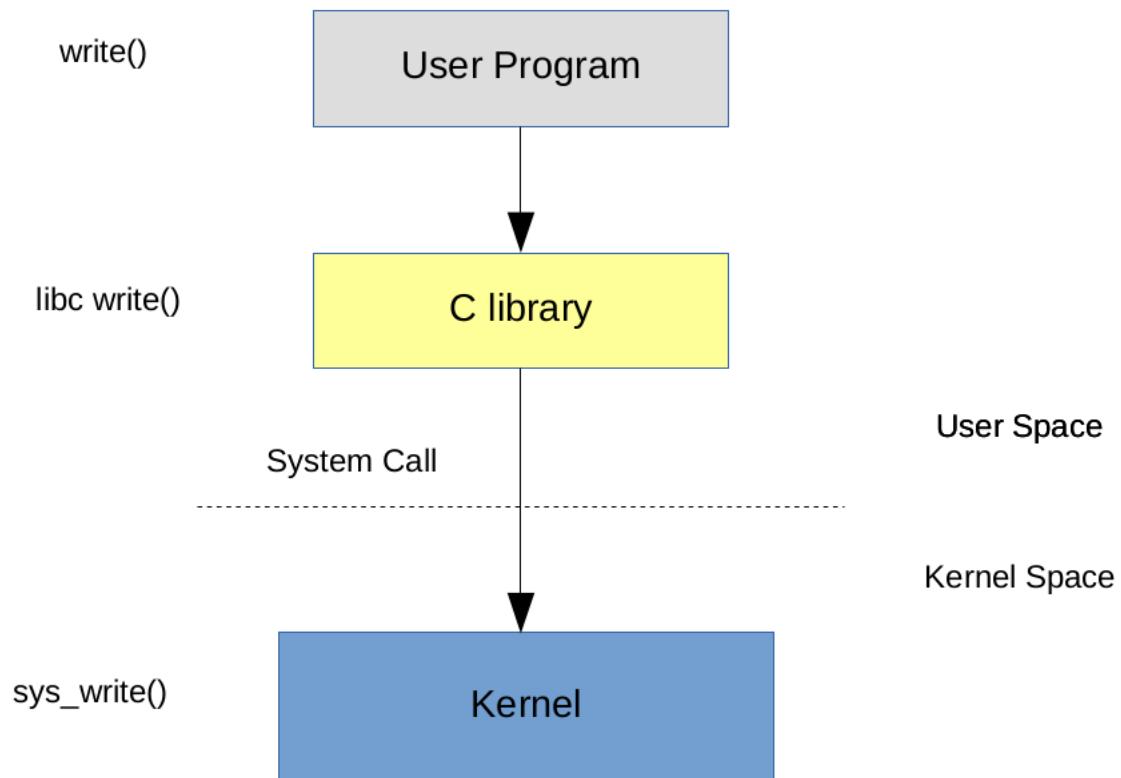
Generalmente, los sistemas proveen una API entre los programas y el sistema operativo de la computadora. En los sistemas basados en Unix suele ser una librería de C, misma que provee subrutinas cuyo propósito es llamar a una llamada al sistema, las cuales generalmente tienen el mismo nombre de la llamada al sistema a la que llaman. Cabe añadir que la llamada a estas funciones de la librería de C no provocan un cambio del espacio de usuario al de kernel por si mismas.

Como dato interesante, Linux usa esta implementación de las llamadas al sistema en la arquitectura x86. En los procesadores con arquitectura RISC (Reduced Instruction Set Computing) es la única implementación.

Un ejemplo de llamadas al sistema se da en la tarea de leer información de un archivo de texto y copiarlo a otro dentro del sistema de archivos del usuario. Primero necesitamos saber el nombre del archivo de entrada, y para ello hacemos una llamada al sistema para obtener ese nombre y también para aceptar esa entrada. Análogamente, necesitamos lo mismo para el archivo de salida.

Para abrir el archivo, necesitamos acceder a la memoria y por ello necesitamos hacer una llamada al sistema. En caso de que el archivo no exista, tenemos que crearlo, y para ello se usa otra llamada al sistema. También las usaremos en el caso de que el archivo de salida ya exista, abortando la ejecución del programa. Todo el ciclo de lectura y escritura entre los archivos se basa en llamadas al sistema, y en el caso de que ocurran errores durante este proceso, se volverá a abortar la ejecución del programa haciendo una llamada al sistema. Así, durante esta simple tarea se realizan muchas llamadas al sistema. Un claro ejemplo de que son sumamente importantes.

2. Imagen para complementar el tema



3. Bibliografía

- Moor, I. (2009). *MIPS - System Calls*. Department of Computing, Imperial College London. Reino Unido. Recuperado el 20 de octubre del 2019 de:
<https://www.doc.ic.ac.uk/lab/secondyear/spim/node8.html>
- Finley, T. (2000). *MIPS examples*. Recuperado el 20 de octubre del 2019 de:
<http://www.tfinley.net/notes/cps104/mips.html>
- Missouri State University. (2011). *SYSCALL functions available in MARS*. Estados Unidos de América. Recuperado el 20 de octubre del 2019 de:
http://courses.missouristate.edu/kenvollmar/mars/Help/Help_4_1/SyscallHelp.html
- Reimann, D. (2012). *File Input, Simple Statistics, and Memory Usage in MIPS*. Albion College. Estados Unidos de América. Recuperado el 20 de octubre del 2019 de:
<http://zeta.albion.edu/dreimann/Spring2012/courses/cs354/projects/stats.php>