UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

Organización y Arquitectura de Computadoras
Practica 8

PRESENTA

Carlos Emilio Castañon Maldonado
Pablo César Navarro Santana

PROFESOR

José de Jesús Galaviz Casas

AYUDANTES

Ricardo Enrique Pérez Villanueva
Sara Doris Montes Incin
María Ximena Lezama Hérnandez

Organización y Arquitectura de Computadoras

Practica 8

Preguntas

1 **¿Que es Shell?**
Da un pequeño resumen sobre que es Shell, para que se usa y como funciona.
La palabra Shell se refiere a el intérprete de comandos que actúa como "interfaz" entre el usuario y el sistema operativo, el Shell proporciona una línea de comandos en la que los usuarios pueden ingresar comandos para realizar tareas específicas en el sistema, como crear archivos, copiar directorios o ejecutar programas.

La forma en que funciona un Shell es que el usuario escribe un comando en la línea de comandos para después presionar Enter, el Shell interpreta el comando y lo ejecuta, para después mostrar la salida del comando en la pantalla.
Shell también proporciona acceso a variables de entorno, utilidades de red, herramientas de programación y una amplia variedad de otros recursos del sistema.

2 **¿Cual es el propósito de un interprete de comandos para cualquier sistema?**
El propósito principal de un intérprete de comandos para cualquier sistema operativo es proporcionar una interfaz de línea de comandos para que los usuarios interactúen con el sistema operativo y realicen diversas tareas, ya que es importante que los usuarios pueden ingresar comandos para que a su vez el intérprete de comandos los interprete, ejecute y muestre en el sistema.

3 **Si tuviéramos que implementar el interprete de comandos en una sistema operativo. ¿Porque es importante usar subrutinas?**
Cuando se implementa un intérprete de comandos en un sistema operativo, es importante usar subrutinas porque permiten reutilizar código y evitar la repetición de código.

En el contexto de un intérprete de comandos, las subrutinas pueden ser utilizadas para realizar tareas comunes, como la lectura de archivos de configuración, la manipulación de archivos, la ejecución de comandos del sistema y otras tareas similares. Al separar estas tareas en subrutinas, se puede reducir la cantidad de código repetitivo y hacer que el código sea más fácil de mantener y actualizar.

Además, el uso de subrutinas también puede hacer que el intérprete de comandos sea más modular y escalable. Si se necesita agregar nuevas funciones o características al intérprete de comandos en el futuro, se puede hacer mediante la creación de nuevas subrutinas que se integren en el sistema existente.

4 **Considerando los componentes básicos de una computadora moderna, además de las llamadas al sistema descritas en la practica, ¿que otras llamadas al sistema resultarían necesarias?** Además de las llamadas al sistema descritas en la práctica, se pueden necesitar otras llamadas al sistema dependiendo de los componentes específicos de la computadora moderna y los requisitos del programa que se está ejecutando. Algunas posibles llamadas al sistema que podrían ser necesarias incluyen:

- Llamadas al sistema para acceder a dispositivos de entrada y salida, como teclados, pantallas, unidades de disco y dispositivos USB.
- Llamadas al sistema para administrar la memoria y el almacenamiento, como asignar y liberar memoria, y acceder a archivos en el sistema de archivos.
- Llamadas al sistema para administrar procesos y programación, como crear y finalizar procesos, administrar subprocesos y señales de interrupción.

5 **De forma general, ¿Que comandos son básicos para un interprete de comandos de un sistema operativo?**
Teniendo en cuenta que el intérprete de comandos de un sistema operativo proporciona una interfaz de línea de comandos para que los usuarios puedan interactuar con el sistema operativo y realizar diversas tareas, los comandos que son básicos en un intérprete de comandos, en general son:

- `cd`: Cambiar de directorio.
- `ls o dir`: Listar los archivos y carpetas en un directorio.
- `mkdir`: Crear un nuevo directorio.
- `rmdir`: Eliminar un directorio.
- `rm o del`: Eliminar un archivo.
- `cp`: Copiar un archivo o directorio.
- `mv`: Mover un archivo o directorio.
- `cat`: Mostrar el contenido de un archivo.
- `echo`: Imprimir texto en la pantalla o en un archivo.
- `chmod`: Cambiar los permisos de un archivo o directorio.

6 **Investiga y describe con tus propias palabras ¿como resuelve una llamada al sistema el sistema operativo?**
Recordemos que una llamada al sistema, también conocida como "system call", es una solicitud realizada por un programa de usuario al sistema operativo para que este último realice alguna tarea en nombre del programa de usuario, la tarea solicitada puede incluir operaciones de entrada/salida, administración de procesos, gestión de memoria, acceso a recursos del sistema, etc.

Cuando un programa de usuario realiza una llamada al sistema, el procesador cambia de modo de usuario a modo kernel, lo que permite que el sistema operativo tenga acceso a los recursos del sistema, a lo que el procesador salta a una ubicación específica de memoria que contiene una tabla de llamadas al sistema ("system call table") que es mantenida por el kernel del sistema operativo.

La tabla de llamadas al sistema contiene una lista de funciones o rutinas que el kernel del sistema operativo puede ejecutar en respuesta a las llamadas al sistema, cada función en la tabla está asociada con un número de identificación único, conocido como "número de llamada al sistema" o "system call number".

Cuando se realiza una llamada al sistema, el programa de usuario especifica el número de llamada al sistema correspondiente a la tarea que desea realizar, el kernel del sistema operativo usa este número para buscar la función correspondiente en la tabla de llamadas al sistema y ejecutarla.
Una vez que la función se completa, el kernel del sistema operativo devuelve el control al programa de usuario, que continúa su ejecución desde el punto en que se realizó la llamada al sistema.