Tarea 1

1. ¿Cuáles son las principales funciones de un sistema operativo?

R: Un sistema operativo es un programa que controla la ejecución de los programas de aplicación y que actúa como interfaz entre las aplicaciones del usuario y el hardware de un computador. Entre las funciones principales están:

- Servir de intermediario entre los usuarios y la parte material o hardware del computador liberando al usuario del conocimiento del hardware.
- Gestionar y organizar la utilización de los recursos del computador (procesador, memoria, discos, periféricos) entre los diferentes programas que pueden estar ejecutándose.
- Permanecer ejecutándose mientras se utilice el computador para poder atender la siguiente tarea que le sea encomendada y darle una respuesta y reaccionar ante posibles errores o situaciones anómalas, Controlar las operaciones de E/S, Controlar las interrupciones, Planificar la ejecución de tareas, Entregar recursos a las tareas, Retirar recursos de las tareas, Proteger la memoria contra el acceso indebido de los programas, Soportar el multiacceso.

2. ¿Qué diferencia existe entre un mandato y una llamada al sistema?

R: Diferencias:

- Los mandatos sirven para crear y administrar procesos, manejar la E /S, administrar el almacenamiento secundario, gestionar la memoria principal, acceder al sistema de archivos, proteger el sistema y trabajar con redes.
- Las llamadas al sistema son la interfaz entre un proceso y el sistema operativo. Estas llamadas generalmente están disponibles como instrucciones en lenguaje ensamblador y casi siempre se listan los manuales empleados por quienes programan en ese lenguaje.

3. ¿Cómo se solicita una llamada al sistema operativo?

R: Los sistemas operativos permiten emitir llamadas al sistema directamente desde un programa escrito en un lenguaje de alto nivel, en cuyo caso las llamadas casi siempre semejan llamadas predefinidas a funciones o subrutinas. Éstas podrían generar una llamada a una rutina especial de tiempo de ejecución que emite la llamada al sistema.

4. ¿Cómo indica POSIX en un programa C el tipo de error que se ha producido en una llamada al sistema? ¿Y Win32?

R: El error que se ha producido en una llamada al sistema se indica:

- POSIX: las funciones normalmente devuelve 0 en caso de éxito ó –1 en caso de error.
- Win32: las funciones devuelven en general true en caso de éxito y false en caso de error.

5. ¿Cuál de las siguientes técnicas hardware tiene mayor influencia en la construcción de un sistema operativo? Razone su respuesta

R: Razonamiento de las técnicas hardware con mayor influencia en la construcción de SO:

- Microprogramación del procesador: es la realización de las instrucciones convencionales (aritméticas, booleanas, de miento, de comparación y de bucle) son realizadas paso a paso por un interprete que se ejecuta en el nivel de microprogramación, consta de dos componentes la ruta de datos y la sección de control.

- Caché de memoria Principal: es invisible para el sistema operativo, interactúa con otras partes del hardware de gestión de memoria.
- DMA (Direct Memory Access): su función se puede llevar a cabo por medio de un módulo separado sobre el bus del sistema o puede estar incorporado dentro de un módulo de E/S.
- RISC (Computadora con Reducido Conjunto de Instrucciones): los programas de usuario se compilan en secuencia de microinstrucciones y son ejecutados directamente por el hardware sin ninguna intervención de intérprete.

La microprogramación de los procesadores posiblemente tenga mayor influencia en la construcción de SOS, ya que ofrece la posibilidad de administrar de manera eficiente el sistema completo dominando los aspectos relacionados con la optimización del tiempo de respuesta y la disminución del tiempo libre que se tenga entre tareas.

6. ¿El intérprete de mandatos de UNIX es interno o externo? Razone su respuesta con un ejemplo

R: El intérprete de órdenes o de comandos de UNIX trata el intérprete de órdenes como un programa especial que se esta ejecutando cuando se inicia un trabajo, o cuando un usuario ingresa en el sistema de tiempo compartido. Esto indica que es externo, ejemplo: para que un usuario pueda ejecutar un comando o mandato debe teclear el comando en la pantalla o Terminal impresora y haciendo uso de la tecla “Enter” lo que le indicara al sistema operativo UNIX que la orden esta completa y que ya la puede ejecutar.

7. ¿Dónde es más compleja una llamada al sistema, en un sistema operativo monolítico o en uno por capas?

R: Las llamadas al sistema son más complejas en un sistema monolítico debido a su carencia de estructura, ya que para poder llamar al sistema ocurre lo siguiente: los parámetros deben colocarse en lugares bien definidos, como en los registros o en la pila, se ejecuta la instrucción especial trampa de nombre llamada al núcleo o llamada al supervisor, luego la instrucción cambia la máquina de modo usuario a modo núcleo y transfiere el control al sistema operativo, se examinan los parámetros de la llamada para determinar cual de ellas desea realizar, el SO analiza la entrada k un apuntador que realiza k-enésima llamada al sistema identifica el procedimiento de servicio, la llamada al sistema termina y el control regresa al programa del usuario. Esto no sucede en un sistema por capas.

8. ¿Qué tipo de sistema operativo es más fácil de modificar, uno monolítico o uno por capas? ¿Cuál es más eficiente?

R: El sistema operativo más fácilmente de modificar es el monolítico ya que como carece de estructura alguna, y el sistema operativo se escribe como una colección de procedimientos, cada uno de los cuales puede llamar a los demás cada vez que así lo requiera, simplifica el tener que conocer muy bien su estructura o Arquitectura. Con referencia a la eficiencia el sistema por capas es mucho mejor por su estructuración que permite alcanzar objetivos de seguridad en ciertos trabajos y metodología para la realización de tareas encomendadas por los usuarios.

9. ¿Debe ser un sistema operativo multitarea de tiempo compartido? ¿Y viceversa? Razone su respuesta.

R: Si puede tenerse un sistema operativo multitarea de tiempo compartido debido a que ambos comparten la característica de ejecución de varias tareas en un computador.

10. ¿Qué ventajas considera que tiene escribir un sistema operativo utilizando un lenguaje de alto nivel?

R: El sistema operativo al utilizar un lenguaje de alto nivel como C, Bliss, BCPL, PL/360, y PERL, los cuales permiten hacer llamadas al sistema directamente, esto es una llamada a una rutina especial en tiempo de ejecución que emite una llamada al sistema directamente en línea. Los implementadotes de C y PERL incluyen acceso directo a las llamadas al sistema y ofrecen una serie de rutinas de bibliotecas.

11. Liste cinco servicios que un sistema operativo presta. Explique cómo cada uno ofrece comodidad a los usuarios, y también en que casos sería imposible que programas en el nivel de usuario proporcionen estos servicios.

R: Entre los servicios que el sistema operativo presta están los siguientes:

1. Ejecución de Programas: el sistema podrá cargar un programa en memoria y ejecutarlo.
2. Operaciones de E/S: programa en ejecución podría requerir E/S. Esta podría implicar el uso de un archivo o un dispositivo de E/S. Para el usuario es cómodo porque no tiene que administrar los dispositivos de E/S, el proceso de comunicación con los dispositivos es totalmente transparente para el usuario.
3. Manipulación de archivos: el sistema crea, elimina, lee y escribe archivos específicos identificados con su nombre. El usuario tiene mejor control dentro del sistema de archivos, fácil manejo y acceso al mismo sin tener que especificar rutas de directorios, o archivos.
4. Comunicación: el usuario se evita el uso de búsqueda de la información requerida, las direcciones, rutas de acceso y demás. En el caso de que cada programa tendría que tener algoritmos de búsqueda individuales más todo el sistema de rutas de acceso etc.
5. Detección de errores: el sistema operativo es capaz de detectar errores infórmalos y eliminarlos. Esta siempre pendiente de los posibles errores. Los procesos internos son invisibles al usuario lo que lo imposibilita para la detección de errores antes de que sucedan por lo cual es más fácil que el SO lo haga.