Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingeniería

División de Tecnologías para la Integración Ciber-humana

Materia: Sistemas Operativos

Profesor: Violeta Del Rocío Becerra Velázquez

Alumno: Denice Estefania Rico Morones

Código: 219421171

Carrera: Ingeniería en Computación

Sección: D04

Título: Estructura y tipos de Sistemas Operativos

Fecha: 30/08/24

Contenido

Modelos de los sistemas operativos:

- **Sistema monolítico:**

  Este modelo no cuenta con una estructura definida, estos mantienen la funcionalidad mínima en un núcleo de alta coherencia, por lo cual sí tienden a integrar una buen parte de la funcionalidad en un sistema kernel se le puede llamar como monolítico; una característica también es que cualquier proceso podía llamar a cualquier otro gracias a esto es por qué no tiene una estructura definida, esto ocasiona que fuera insostenible con el paso del tiempo. Las partes del kernel de este se compilan por capas, como ejemplos de sistemas operativos monolíticos tenemos a Unix, Linux y FreeBSD, Linux es un buen ejemplo ya que cuenta con un kernel muy potente. Sí hay alguna falla en el núcleo de este todo el sistema puede caer.

- **Sistema cliente-servidor:**

  Este tipo de sistema operativo es base de los SO distribuidos, consta del modo privilegiado que dependen de la seguridad gracias a esto hay ciertas acciones que no se pueden realizar (al estar en modo usuario), un proceso se puede convertir en uno u otro. Lo que hace el núcleo de este es controlar la comunicación mediante mensajes entre el cliente y el servidor; si se quiere abrir una aplicación en un archivo dentro de una aplicación se manda un mensaje o petición al servidor del sistema de archivos, cada servidor tiene la capacidad de mandar otros mensajes ah demás servidores, todo esto siempre manejándose entre una comunicación como su nombre lo indica cliente/servidor donde toda acción va a requerir de su “autorización”. Si la relacionamos con un poco con los sistemas distribuidos cada mensaje que se envíe desde el cliente al servidor necesita y debe tener un identificador, las relaciones muchos-a-uno son una característica más de esta clasificación, este permite múltiples emisores y receptores.

- **Máquina virtual:**

  Emulador de SO, presentó una interfaz que emula a una máquina real; dentro de estos tenemos un elemento llamado monitor el cual puede ejecutar el hardware, además de realizar la programación y proporcionar varios tipos de máquinas virtuales, tienen una gran ventaja la cual es poder ejecutar cualquier sistema operativo de forma directa en ella, cuentan con la característica llamada Conversational Monitor System (CMS) el cual funciona como una serie de instrucciones para la lectura del disco virtual.

- **Capas:**

  Estos son niveles con una comunicación más estructurada, cada capa es un servicio o gestor, las capas son organizadas jerárquicamente y sólo interactúan las capas adyacentes esto ejecutándose en modo núcleo, las capas pueden llegar a tener demasiados procesos o demasiada funcionalidad que puede tener grandes efectos los cuales pueden llegar a ser un problema ya que puede ser difícil construir la seguridad gracias a las demasiadas interacciones entre las capas adyacentes.

  Estos sistemas constan de 6 capas:

  - Capa 0: Hardware.
  - Capa 1: Administración de memoria y discos
  - Capa 2: Administración de procesos.
  - Capa 3: Dispositivos de entrada y salida.
  - Capa 4: Programas de usuario.
  - Capa 5: Interfaz de usuario.

- **Hibrido:**

  Estos funcionan mediante la paginación para lograr implementar interfaces convenientes de programación. Estos son las bases de muchos sistemas de administración de memoria, además gestionadas los dispositivos de entrada y salida esto para mejorar el rendimiento del sistema; el paso de mensajes y la ejecución de componentes del sistema en espacio de usuario es otra característica de estos.

Linux

- **Historia:**

  Todo esto comienza en el año 1991 gracias a Linus Torvalds, él, comenzó a desarrollar un sistema de operativo inspirado en UNIX, estoy con un proyecto pequeño, pero al momento de liberar el código fuerte bajo la licencia GNU, permitió que otros desarrolladores pudieran contribuir. Linux comenzó a ganar popularidad entre la comunidad de desarrollo esto al ser un sistema libre y abierto, muchos decidieron contribuir mejorando el sistema y añadiendo nuevas funcionalidades. Linux es considerada como una alternativa viable a otros sistemas operativos. Con el tiempo distintas distribuciones de Linux comenzaron a surgir como por ejemplo Slackware, Debian y Red Hat. Una característica que yo considero importante es que Linux ha llegado a ser la base del sistema operativo Android ya que es una distribución de uso específico del kernel de Linux.

- **Servicios que presta:**

Linux está relacionado con la disponibilidad de computadoras personales, permite dar funcionalidad y soporte a la paginación gracias a esto realizar la implementación de su administración de memoria (híbrida); permite generar segmentos de memoria en páginas grandes lo que ayuda a que el sistema operativo requiera memoria contigua sin preocuparse de los límites o del tamaño de la página, Linux protege los segmentos de memoria de aplicaciones del sistema operativo para que no sean modificados por las aplicaciones del usuario. Los entornos de las ventanas en Linux pueden funcionar sobre un programa de gestión de ventanas, el sistema primero carga este gestor antes de mostrar el entorno. Linux cuenta con su línea de comandos o Shell con el cual el usuario puede interactuar directamente con el sistema operativo, es un método muy común de interacción entre usuario-sistema.

Linux es multitarea y multiusuario ya que permite que múltiples usuarios trabajen simultáneamente en un mismo sistema ejecutando múltiples procesos a la vez al igual que tiene añadido un planificador de tareas para gestionar la asignación de estas mismas; soporta una gran variedad de archivos y ofrece buen sistema de permisos y control de acceso, tiene la capacidad de actuar como un servidor de red, de firewall y soporta múltiples protocolos de red. La virtualización y el servicio en la nube también son parte de este sistema operativo.

- **Objetivos:**

Linux tiene como objetivo proporcionar una libertad de uso gracias a su código fuente disponible para todos haciendo posible modificarlo y distribuirlo libremente, esto además puede proporcionar confiabilidad mediante el acceso a su código fuente; la colaboración abierta y global promueve una innovación constante incorporando mejoras al sistema operativo.

Seguridad y protección es un objetivo más de este sistema operativo ya que cuenta con diversas capas de seguridad como por ejemplo la gestión de permisos, además vale la pena añadir que es un sistema operativo estable. Linux además se adapta a diferentes entornos, busca que pueda escalar desde dispositivos pequeños hasta supercomputadoras, además de poder personalizarlo según nuestras necesidades.

- **Funciones:**

- **Multitarea:**

  Linux es multitarea por lo cual tiene la funcionalidad y capacidad de ejecutar muchos programas al mismo tiempo sí de tener o afectar la ejecución de las aplicaciones, lindo se logra esto mediante la administración de los recursos garantizando que cada proceso y parte de ellos tenga su tiempo de procesador.

- **Multiusuario:**

  Permite que varias personas puedan acceder al sistema compartiendo recursos que el administrador decide y asigna a cada uno, esto también añadiendo privacidad y protección.

- **Soporta consolas virtuales:**

  Ofrece soporte para la creación y gestión de máquinas virtuales.

- **Sistema operativo de red:**

  Se puede tener más de una sesión abierta en la consola y conmutar entre ellas, además de acudir funcionar como un sistema de red completo ofreciendo servicios como DNS, DHCP, HTTP/HTTPS, FTP, SSH, etcétera.

- **Direccionamiento:**

  Tiene un direccionamiento de 32 bits en PC y 64 bits en sistemas Alpha.

- **Kernel:**

  Tiene la capacidad de aprovechar las características del modo protegido.

- **Soporte de hardware:**

Linux es compatible con hardware multimedia como también módems, impresoras, tarjetas de vídeo, monitores, teclados, etcétera.

- **Entorno gráfico:**

Cuenta con un potente y versátil entorno gráfico con numerosos sistemas de ventana como FWVM, GNOME, KDE, CDE, Enlightenment, Afterstep, NextLevel, @rM, entre otros.

Estructura

Su arquitectura/estructura cuenta con un núcleo o kernel y sobre éste una capa de interfaz de usuario, cómo dato el kernel es la parte del sistema operativo más cercana al hardware de la computadora.

- Gestión de procesos.
- Gestión de archivos.
- Gestión de memoria.
- Gestión de entrada-salida.
- Interfaz de llamadas al sistema.

Su interfaz de llamadas recibe las solicitudes de los programas o bibliotecas, en la versión del sistema operativo 5.0 si algún proceso es fatal el sistema operativo decide sobre qué acción va a tomar; la parte del subsistema se encarga de gestionar las interrupciones, la planificación de procesos, la comunicación entre procesos en la gestión de la memoria. Otra parte de su arquitectura es el subsistema de archivos mediante el cual se maneja el intercambio de los datos entre la memoria y los dispositivos externos, Encontró del hardware es mediante la interacción entre el kernel y el mismo hardware no todas las funciones deben de estar integradas en el kernel. Argumento de querer configurar el kernel podemos determinar qué funciones necesitamos incorporar de manera fija y de manera de módulos.

Los módulos del kernel siempre se guardan en la misma ruta la cual es: lib/module/<versión>; Es una web práctica utilizar módulos ya que en el arranque del kernel no siempre se necesitan las funciones por esto mismo se deben de tratar como módulos, Esto ayuda a que el BIOS no tenga problemas al cargar el kernel.

¿Qué significa JCL?

Las siglas JCL significan **lenguaje de control de trabajos/monitores** (Job Control Language), el cual es un lenguaje de programación que es utilizado para poder dotar de instrucciones a un monitor, el monitor realiza una función de planificación esto es un lote de trabajos el cual se trata de ejecutar lo más rápido posible sin ningún tiempo ni una interrupción de por medio; los monitores pueden mejorar gracias a la configuración de los trabajos los cuales consisten en conjuntos de instrucciones en formatos JCL. Esto se puede explicar enviando un programa escrito en algún lenguaje JCL el cual contendrá instrucciones para el control de trabajo expresadas mediante el símbolo '$'. Al ejecutar este trabajo es necesario que el monitor lea la línea $FTN para cargar al compilador apropiado, se traduce este código como código objeto para almacenarla en memoria del sistema de almacenamiento lo cual se denomina como "compilar, cargar y ejecutar", si se quiere almacenar una cinta tenemos que utilizar el comando $LOAD, al ser leída esta instrucción por el monitor se recupera el control después de la compilación, en monitor tiene que invocar al cargador para poder utilizar el código objeto en el lugar del compilador y darle todo el control a este.

Diferencia entre el procesamiento por lotes y el de procesamiento por lotes con multiprogramación.

Una de las principales diferencias y de las más notorias es la utilización eficiente de los recursos y mediante esto obtenemos un tiempo de respuesta más corto esto ya que se permite ejecutar múltiples procesos y con la capacidad de poder cambiar entre ellos, así que podemos ejecutar múltiples programas con los sistemas de multiprogramación así utilizamos y aprovechamos todos los recursos del CPU y la memoria. Como hemos visto en el procesamiento por lotes los programas o procesos son ejecutados uno tras otro sin ninguna interacción del usuario; con ayuda de la multiprogramación el procesamiento por lotes se vuelve más eficiente. En el procesamiento por lotes el usuario no interactúa con la computado además de que no puede acceder a un modo "administrador".

Utilidades de la interrupción int86 en C:

Esta función se utiliza para interrumpir la ejecución de un software (llama a interrupciones del BIOS).

Declaración: *int86(int intnum, union REGS *in, union REGS *out)*

Intnum representa el número de la interrupción, **in** es la unión entre los manejadores de la interrupción los cuales nos va a permitir pasar la información a estos, **out** este guardará los valores devueltos por la interrupción. Además, suele usarse con números en hexadecimal, además de interrumpir también podemos solicitar a un programa que realice una acción o tarea específica.

En cuanto las tareas o utilidades en las que se puede aplicar esta función son: al imprimir la pantalla, operaciones de entrada y salida de vídeo, entrada y salida de disco, entradas o salidas del puerto serie, del teclado, de la impresora, operaciones de hora y fecha hoy interrupciones del control del ratón.

Si queremos utilizarlo como por ejemplo en el manejo del ratón invocamos primero al int 33h (para manejo del ratón) el cual contendrá un ciclo que mientras no se oprima ninguna tecla va a monitorear constantemente el estatus del ratón, esto inicialmente con un estatus AL=3 El cual puede devolver en BX = 1 sí se oprimió el botón izquierdo o BX = 2 si se oprime el botón derecho.

¿Para qué sirve la función Kbhit?

Está función inspecciona el buffer del teclado, retorna un cero sin hay teclas pulsadas almacenadas en el buffer, y en caso contrario si hay una tecla almacenada (devuelve un valor distinto a cero y diferente de uno); es muy común combinar esta función con el uso de getch() si se quiere realizar una acción sólo si sea presionada una tecla sin la necesidad de que el programa se detenga o tenga que esperar por la entrada del usuario (se le puede llamar como un tiempo de interrupción), es muy eficiente utilizarlo a unos ya que el programa sí ejecutándose hasta que nosotros queramos y gracias a estos podemos implementar controles específicos cómo salir del programa, pausar, etcétera.

Equivalente de Kbhit (utilizado en c) en otros lenguajes de programación

- **Python:**

msvcrt.kbhit()

La cual retorna True si hay una tecla que está esperando por ser leída.

- **Javascript:**

Readline en Node.js

Este hacer un lenguaje que se ejecuta en entornos web, se puede utilizar esta función en consola mediante un enfoque basado en eventos, en este caso, eventos de teclas presionadas.

Conclusión:

Como conclusión puedo decir que es importante saber qué tipos de sistemas operativos existen y poder identificarlos para así saber con cuál nos conviene trabajar y cuáles son los más utilizados actualmente, es importante expandir nuestros conocimientos y no sólo enfocarnos en un solo sistema operativo por lo cual yo hablé de un sistema operativo que nunca he usado ya que me parece importante también conocerlo y cuáles son sus ventajas y en qué se diferencia con el sistema operativo que yo normalmente uso ya que puede tener usos totalmente diferentes y que tal vez este SO que yo tengo en mi computadora no sea capaz de realizar; Hay nuevos términos que conocer los cuales unos son muy complejos ya que unos tienen que ver con otros aún no son la base de otros entonces mediante las conexiones estos se crea un concepto que globaliza toda una parte este tema que estamos viendo como lo fue el tema de JCL con los monitores y el procesamiento por lotes y multiprogramación. Finalmente puedo decir que en los lenguajes de programación hay funciones muy importantes que nos pueden ayudar a facilitar el trabajo y que son muy importantes al momento de querer simular algún proceso, no funciona en específico que se vio en esta investigación fue kbhin() la cual nos ayuda a obtener teclas previamente almacenadas yo decidí agregar también la función getch() ya que puede trabajar de la mano con la función anterior para poder mediante teclas realizar cualquier acción dentro de este programa sin la necesidad de que este termine.

Bibliografía:

Cura, N. J. (2020). *Fundamentos de Sistemas Operativos.* Córdoba: Universitas.

Gelpi Fleta, D., & Sierra González, J. M. (s.f.). *Sistemas Operativos Monopuesto.* Madrid: MACMILLAN.

Llaven, D. S. (2015). *Sistemas operativos. Panomarama para ingeniería en computación e informática.* México: Patria.

O'Reilly Media, Inc. (s.f.). *O'REILLY*. Obtenido de O'REILLY: https://www.oreilly.com/library/view/python-standard-library/0596000960/ch12s11.html

pranjal_srivastava. (4 de Abril de 2023). *Geeks for Geeks*. Obtenido de Geeks for Geeks: https://www.geeksforgeeks.org/node-js-readline-module/

Python Software Foundation. (s.f.). *Python*. Obtenido de Python: https://docs.python.org/es/3/library/msvcrt.html#msvcrt.kbhit

Raya Cabrera, J. L., & Raya González, L. (2014). *Sistemas Operativos Monopuesto 2ª Edición.* Madrid: Ra-Ma.

Red Hat, Inc. (31 de Julio de 2023). *Red Hat*. Obtenido de Red Hat: https://www.redhat.com/es/topics/linux

Serna, M., & Allende, S. (2020). *Sistemas Operativos Linux.* Argentina: UNIVERSITAS.

Stallings, W. (2005). *Sistemas operativos. Aspectos internos y principios de diseño.* Madrid: Pearson Education.