El Sistema Operativo, Tipos, Funcionamiento y Componentes

Prof. Aris Castillo de Valencia
2010

Objetivos:

- Discutir las funciones principales de los sistemas operativos
- Diferenciar los tipos de acuerdo con su operación y funciones
- Evaluar los componentes del sistema operativo
- Diferenciar las estructura de los sistemas operativos
- Características de los sistemas operativos modernos
- Describir el proceso de arranque del Sistema Operativo

Tabla de Contenidos

Definición

Un sistema operativo es un programa que controla la ejecución de programas de aplicación y que actúa como interfaz entre las aplicaciones del usuario y el hardware del computador.

Principales funciones del sistema operativo

Por áreas de servicio

- Desarrollo de programas – ej. editores, debuggers.
- Ejecución de programas – incluye acciones como cargar a memoria, acceso a archivos.
- Acceso a dispositivos de E/S – ejecutar instrucciones y señales de control propias de cada dispositivo.
- Acceso controlado a archivos – compresión, estructura de datos en los archivos, medio de almacenamiento.
- Acceso al sistema – provee protección a recursos y datos.
- Detección y respuesta a errores – errores internos y externos de hardware, acceso a memoria, desbordamiento, etc.
- Contabilidad - estadísticas, monitoreo de rendimiento del sistema.

Por recurso controlado

- La memoria es controlada por el sistema operativo y por el hardware de manejo de memoria en el procesador.
- El OS controla el acceso y utilización de dispositivos de E/S, acceso y utilización de archivos, así como el tiempo de uso del procesador.
- La porción del sistema operativo en memoria incluye el kernel (que contiene las funciones más frecuentemente usadas) y en ciertos momentos, incluye otras partes del OS que estén en uso.

Tipos de Sistemas Operativos

Según el tipo de computadora que controlan y las aplicaciones que soportan, los sistemas operativos se pueden clasificar en cuatro tipos:

Sistema Operativo de tiempo real (RTOS): es un sistema operativo que ha sido desarrollado para aplicaciones de tiempo real y utilizado para controlar instrumentos científicos y sistemas industriales. El objetivo no es un rendimiento alto, sino garantizar el cumplimiento de una tarea. Puede haber dos tipos de sistemas operativos de tiempo real:

- Tiempo real duro (Hard-real time) en que los plazos se alcanzan determinísticamente.
- Tiempo real suave (Soft-real time) en que el objetivo alcanzar un plazo.

Algunas de las filosofías de diseño de estos sistemas operativos incluyen:

- El sistema operativo funciona por eventos, lo que significa que se realiza intercambio de tareas sólo cuando un evento de mayor prioridad requiere servicio, llamado prioridad preferente (preemptive).
- El diseño es de tiempo compartido (time-sharing) lo que significa que las tareas se van intercambiando de acuerdo con interrupciones regulares de reloj o por eventos, generalmente a través de algoritmos Round Robin.

Inicialmente, los diseños de CPUs requerían muchos ciclos para el intercambio tareas, lo que significaba que el CPU no estaba realizando algo útil. Por ejemplo, un procesador 68000 de 20 MHz el intercambio de tareas se realizaba cada 20 microsegundos; mientras que con un procesador de 100 MHz se realiza cada 3 milisegundos. Por esta razón, los diseños iniciales trataban de reducir el intercambio de tareas innecesario.

Algunos ejemplos son:

a. **QNX.** Es un sistema operativo comercial tipo Unix, principalmente en el mercado de los sistemas embebidos. Tiene licencia propietaria. Es un sistema microkernel donde la mayor parte del OS corre como un conjunto de tareas pequeñas conocidas como servidores. Esto difiere de los sistemas monolíticos en que el OS es un gran programa compuesto de un gran número de partes con habilidades especiales. En el caso de QNX, los usuarios pueden desactivar funciones que no están necesiten sin tener que

cambiar el OS mismo. Dichos servidores simplemente no corren. Corre en plataformas PowePC, familia x86, MIPS, SH-4, ARM, StrongARM y XScale. [3]

b. **RTLinux.** Es un sistema operativo tiempo real duro y microkernel que corre Linux por completo como un proceso preferente. Es basado en una máquina virtual ligera (light weight virtual machine) donde el huésped Linux tiene un controlador de interrupción y un temporizador virtualizados, mientras que todos los demás accesos al hardware son directos. Es de licencia abierta (open source) y programado en lenguaje C. [5]
c. **WindowsCE.** Su nombre oficial es Windows Embedded Compact. Es desarrollado por Microsoft usando lenguaje C y su licencia es propietaria. Corre en procesadores Intel x86, MIPS, ARM e Hitachi SuperH. Es de 32 bits y permite integrar capacidades de tiempo real con tecnologías Windows. [6]

Monousuario – Monotarea**:** está diseñado para administrar la computadora de manera que un solo usuario puede ejecutar una tarea a la vez. Ejemplo: Palm OS para computadoras Palm portátiles, DOS de Microsoft. En este caso el sistema operativo no es capaz de realizar más de una tarea en un tiempo determinado. Esto significa que no hay intercambio de tareas en el CPU y que la memoria no requiere ser compartida. En este caso la memoria estaría dividida en dos bloques – uno para el sistema operativo y otro para el proceso activo.

Monousuario – Multitarea**:** es ampliamente utilizado ya que permite a un usuario ejecutar varias tareas al mismo tiempo. Sin embargo, no tiene la capacidad de más de una sesión de usuario a la vez. Ejemplo: las plataformas Microsoft Windows y sistemas operativos MAC de Apple.

Este tipo de OS es más complejo que el monousuario-monotarea ya que requiere capacidad para administrar diversos procesos simultáneamente, lo que implica asignación de identificadores, espacios de memoria privados y compartidos, y planificación de recursos tales como el CPU y los dispositivos de E/S.

Multiusuario**:** permite a diferentes usuarios tomar ventajas de los recursos de la computadora de forma simultánea. Para que este sistema operativo funcione de manera correcta se debe asegurar que los requerimientos de los usuarios estén

balanceados de manera que si se presenta algún inconveniente con un usuario, no afecte a los demás usuarios del sistema. El hecho de ser multiusuario implica necesariamente que sea multitarea. Ejemplos: Sistema Operativo Unix, Mainframes, Linux y de Windows Vista en adelante.

Sistema Operativo **Monoprogramado y Multiprogramado vs Monoprocesador y Multiprocesador.**

Desde el punto de vista de las tareas que el sistema operativo es capaz de realizar, se habla de:

- Monoprogramación: modo de operación que solo permite la ejecución de una tarea o programa en un único procesador.
- Multiprogramación: modo de operación permite la ejecución intercalada de dos o más tareas o programas en un único procesador.

Utilización de los procesadores.

Desde el punto de vista, del número de procesadores reconocidos por el sistema operativo, se habla de:

- Sistemas operativos monoprocesador (un solo CPU)
- Sistemas operativos multiprocesador (dos o más CPUs).

Tipos de sistema operativo multiprocesador

Un sistema operativo multiprocesador debe dividir el trabajo de carga entre los CPUs para balancear las demandas de los procesos requeridos con los ciclos disponibles en los diferentes CPUs. Entre los diferentes tipos tenemos:

- Sistema Operativo Asimétrico (AMP)**:** utilizan un CPU para sus propias necesidades (las del OS) y dividen lo procesos de aplicación entre los CPUs restantes.
- Sistema Operativo Simétrico (SMP)**:** las tareas del OS y los procesos de usuario se dividen entre los distintos CPUs, balanceando la demanda contra la disponibilidad del CPU.

Componentes de los Sistemas Operativos

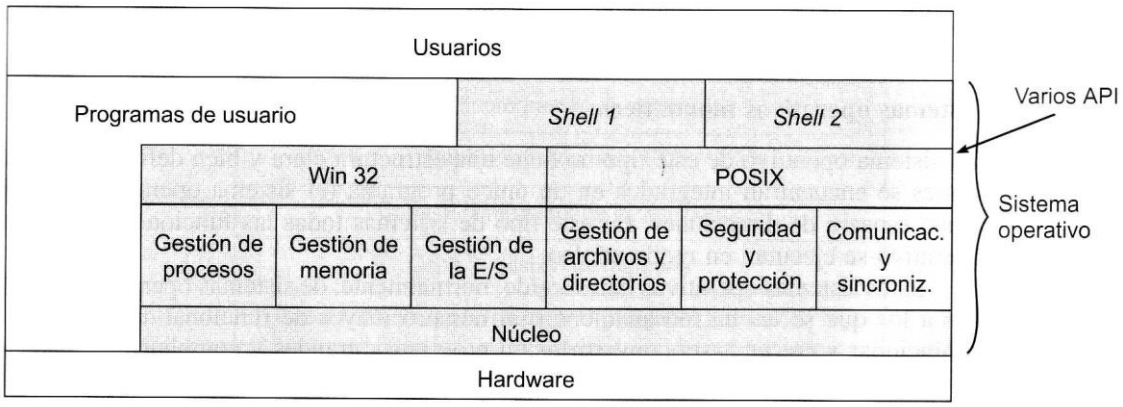


Figura 2.5. *Componentes del sistema operativo.*

Fuente: Carretero, Pag. 41
Veamos cada uno de estos componentes.

Shell

Es una pieza de software que provee una interfaz para los usuarios del sistema operativo, de manera que puedan tener acceso a los servicios del kernel. Estas piezas de software interpretan comandos. Generalmente hay dos categorías:

- Línea de comandos**:** proveen una interfaz de línea de comandos (CLI, command line interface) al sistema operativo. Algunos ejemplos son los shells de Unix tales como Bourne Shell (sh), Bourne-again Shell (bash), Korn Shell (ksh), C shell (csh), entre otros. Shells no Unix incluyen cmd.exe que es el principal shell de OS/2, Windows CE y Windows NT; command.com es el Shell para varios DOS de Windows; Google Shell que es el utilizado en Google Search.
- Gráfico**:** proveen una interfaz gráfica al usuario (GUI, Graphical User Interface). En el caso de sistemas X Windows, existen manejadores X Windows independientes y ambientes desktop completos que dependen de un manejador de ventanas. Ejemplos del primer caso son Blackbox y Fluxbox. En el segundo caso están CDE, GNOME, KDE, Xfce y LXDE. Para otras plataformas, existen otros Shells. [10]

  Es importante anotar que también existen shells para Lenguajes de Programación.

- API – Application Program Interface**.**

Es una abstracción que describe una interfaz para la interacción entre un conjunto de funciones utilizadas por los componentes de un sistema de software. En otras palabras, las API permiten a los programadores de aplicaciones usar funciones de la computadora y del sistema operativo sin necesidad de seguir los detalles de operación del CPU. El software que provee las funciones descritas por un API se dice que es la implementación del API.

WIN 32 y POSIX son implementaciones de API. El estándar POSIX define un gran conjunto de funciones computacionales a ser escritas de forma tal que operen en diferentes sistemas. MAC OS X y varias distribuciones BSD implementan esta interfaz, sin embargo, se requiere recompilar el sistema para usarlas. Las API de Microsoft, Windows API o Win32 corren en nuevas versiones a través de Compatiblity Mode. Las API compatibles permiten que no se requieran cambios en el sistema, para lo cual se requieren bibliotecas (Dynamic Link Library, DLL) que permiten las llamadas de las aplicaciones de usuario (user applications) al sistema [11].

- Componente Gestión de Procesos**:**

  Implica todos los aspectos administrativos que debe realizar el OS para asegurar que varios procesos puedan ser ejecutados en el sistema.

  - Se debe asegurar que cada proceso y aplicación reciba la cantidad suficiente de tiempo del procesador para funcionar correctamente.
  - Se debe permitir la utilización de muchos ciclos de procesos para trabajos reales.

Componente de Gestión de memoria y Almacenamiento

- Cada proceso debe tener suficiente memoria para ejecutar, y tampoco puede correr dentro del espacio de otro proceso.
- Los diferentes tipos de Memoria en el sistema deben ser utilizados adecuadamente.

Incluye aislamiento de procesos (configurar los límites de memoria), que los programas sean alojados dinámicamente en la jerarquía de memoria, soporte de programación

modular (programadores pueden crear, destruir, y cambiar el tamaño de módulos dinámicamente), protección y control de acceso, almacenamiento a largo plazo.

El sistema de archivo implementa un almacenamiento a largo plazo, con información almacenada en archivos en memoria secundaria. La memoria virtual permite a los programas direccionar memoria desde un punto de vista lógico, con esto se permite tener más procesos activos concurrentemente residentes entre la memoria principal y la secundaria porque los procesos no requieren tener todas sus páginas en memoria principal. Generalmente, usando un sistema de paginación (los procesos están compuestos de un número de bloques de tamaño fijo llamados páginas.

Para que un programa de usuario corra algunas de sus páginas deben estar en memoria principal. Para el intercambio de datos se usan direcciones virtuales y reales, donde el mapper (MMU- Memory Management Unit) hace la traducción. Stallings, p. 74

Según su velocidad, las memorias se clasifican en memoria caché, memoria principal (RAM), memoria secundaria (almacenamiento por rotación magnética). El sistema operativo debe mantener un balance entre las necesidades de los diversos procesos con la disponibilidad de los diferentes tipos de memoria, moviendo los datos en bloques (llamados páginas) entre la memoria disponible [9].

- Componente de Administración de Dispositivos de E/S**:**

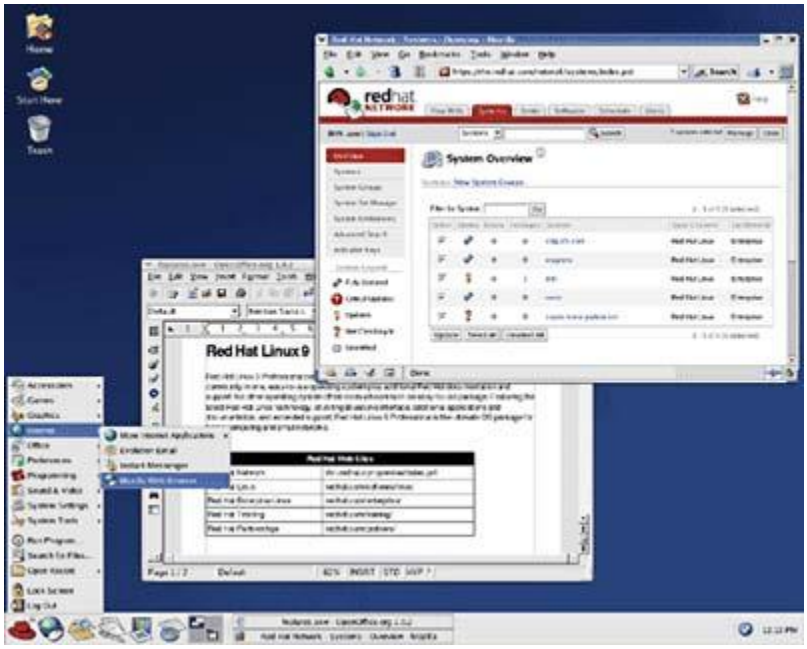
  La ruta entre el sistema operativo y virtualmente todos los componentes de hardware en la tarjeta madre va a través de programas especiales denominados manejadores (drivers). La función principal de un driver es servir de traductor entre las señales eléctricas del hardware y el lenguaje de programación del sistema operativo y los programas de aplicación. Un driver provee una forma en que las aplicaciones puedan hacer uso de subsistemas de hardware sin necesidad de conocer los detalles de operación del mismo.

  La razón por la que los driver son separados del sistema operativo es porque nuevas funciones pueden ser añadidas al mismo de este modo a los subsistemas del hardware sin requerir la modificación del sistema operativo.

La administración de todos los recursos del computador es una gran parte de la función del sistema operativo, y en algunos casos de sistemas operativos en tiempo real [9].

Interfaz de usuario (UI):

Brinda la estructura para la interacción entre un usuario y la computadora. En las últimas décadas casi todo el desarrollo de interfaces de usuario se ha enfocado en el área de interfaz de usuario grafico (GUI), destacándose los modelos Apple´s Macintosh y Microsoft Windows creciendo más atención y ganancia, pero el más popular es el sistema operativo Linux de código abierto que también soporta la interfaz de usuario gráfico.



La base de la función de sistema operativo y la administración del sistema de la computadora está en el kernel del sistema operativo. El administrador de imágenes es separado pensando que podía estar ligado debajo del kernel. Este enlace entre el kernel del sistema operativo y la interfaz de usuario, utilidades entre otros software definen muchas de las diferencias de los sistemas operativos de hoy.

Proceso de Arranque del Sistema Operativo:

Al encender nuestra computadora, el primer programa que corre es un conjunto de instrucciones mantenidos en la Memoria de Solo Lectura (ROM) de la computadora. Este

código examina el hardware del sistema se asegura de que esté funcionando correctamente.

Este POST (power-on self test) revisa el CPU, la memoria y el sistema básico de entrada y salida (BIOS) en busca de errores y almacena el resultado en una localización de memoria especial. Una vez que el POST se completa de manera exitosa, el software ROM (BIOS) empezará a activar los controladores de disco de la computadora. Una vez esto sucede, se encuentra la primera pieza del sistema operativo en disco: el bootstrap loader. El bootstrap loader es un pequeño programa que tiene la función de cargar el sistema operativo a la memoria y le permite iniciar operación. El boostrap configura los drivers y los subsistemas de la computadora; establece las divisiones de memoria para el OS, información de usuario y aplicaciones; establece las estructuras de datos para las señales, banderas y semáforos que se usan para la comunicación entre subsistemas y aplicaciones. Luego de esto, se pasa el control de la computadora al sistema operativo.

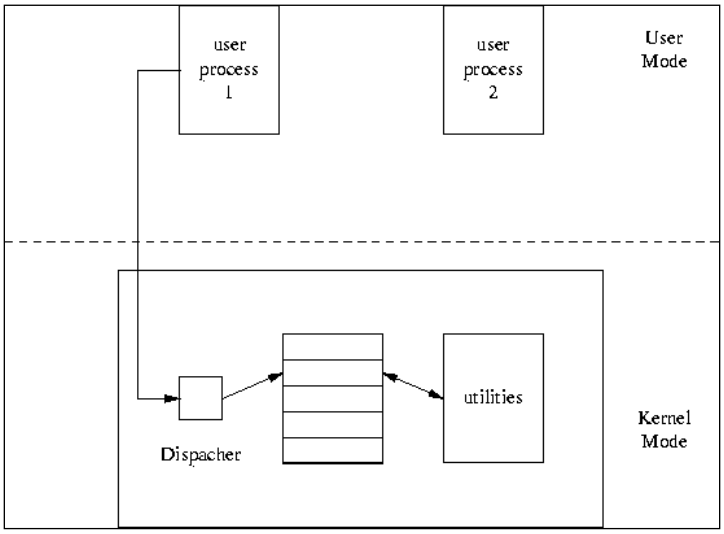
Estructura de los sistemas operativos:

Monolítico

Es la arquitectura más antigua. Un núcleo **monolítico** significa un único proceso muy grande con todos sus elementos integrados en un mismo espacio de memoria; todas las funciones del sistema operativo se ejecutan en modo núcleo.

Características:

- Un programa enorme
- No hay información oculta
- Llamadas supervisor
- Modo usuario vs modo núcleo (kernel)



Fuente: [12] Arquitectura monolítica

Cómo funciona?

1. Un proceso realiza una llamada al sistema – modo usuario a modo kernel
2. Revisión de parámetros
3. Llamada a la rutina de servicio
4. Rutina de servicio llama a la función de utilidad
5. Reprograma y regresa al proceso de usuario

Ejemplo: El núcleo de Unix es el maestro que controla todos los demás programas, les asigna tiempo para acceder al hardware, y maneja el sistema de archivos y el modelo de seguridad. El núcleo se ejecuta como el proceso raíz con privilegios especiales. Todo lo demás corre en cuentas de usuario restringidas. Todo lo que esté fuera del kernel se denomina “espacio de usuario”.

Estructuras de modelos de sistemas operativos monolíticos:

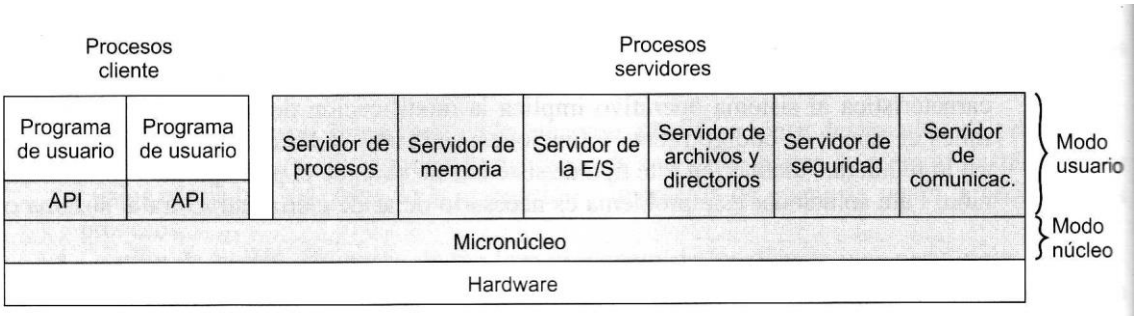


Figura 2.7. *Estructura cliente-servidor en un sistema operativo.*

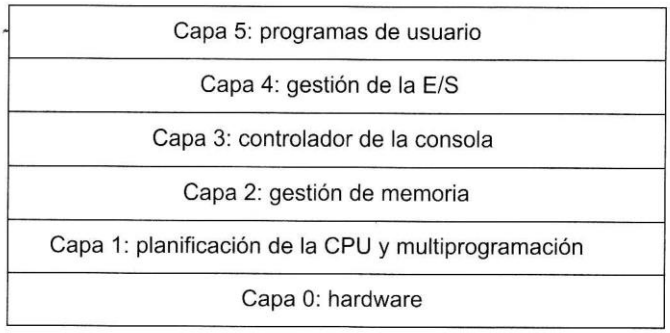


Figura 2.6. *Estructura por capas del sistema operativo THE.*

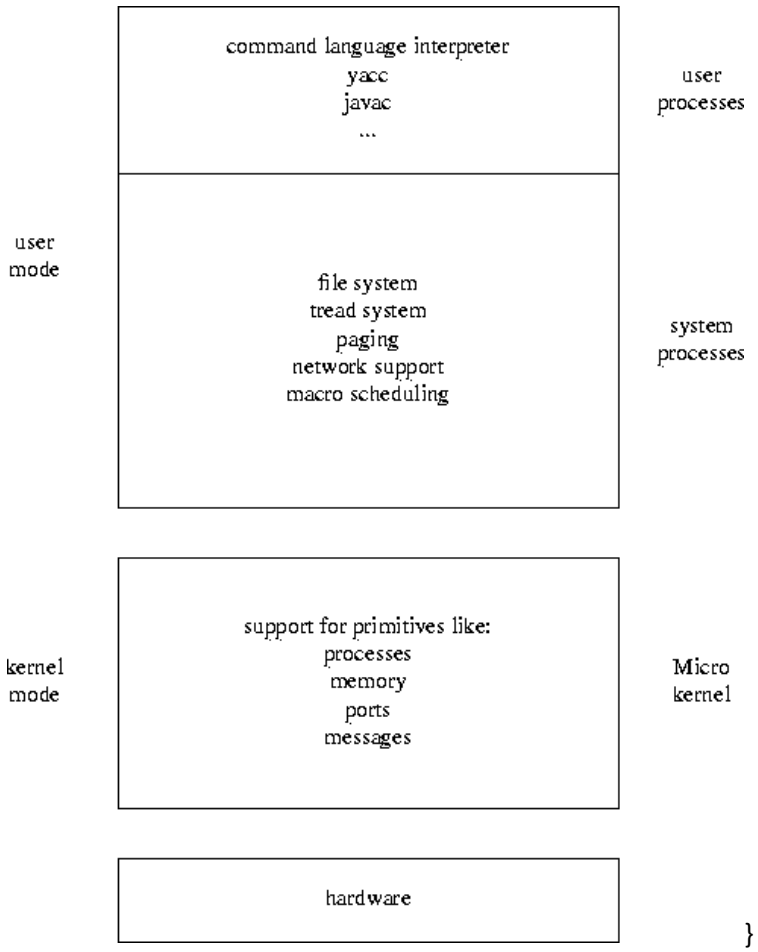
Nota: *Cada capa se comunica sólo con la capa inmediatamente superior y utiliza sólo los servicios de la capa inferior.*

Micronúcleo

La mayor parte del sistema operativo se ejecuta en procesos separados, fuera del núcleo. Se comunican por paso de mensajes. La tarea del núcleo es manejar el paso de mensajes, las interrupciones, administración de procesos de bajo nivel, y tal vez las entradas/salidas. Ejemplos de sistemas operativos con este diseño son: RC4000, Ameba, Chorus, Mach y Minix.

El micronúcleo asigna solamente algunas funciones esenciales al núcleo (espacio de direcciones, comunicación entre procesos, y planificación básica). Otros servicios son provistos por procesos llamados servidores que corren en modo usuario y son tratados como otras aplicaciones por el micronúcleo.

Arquitectura:

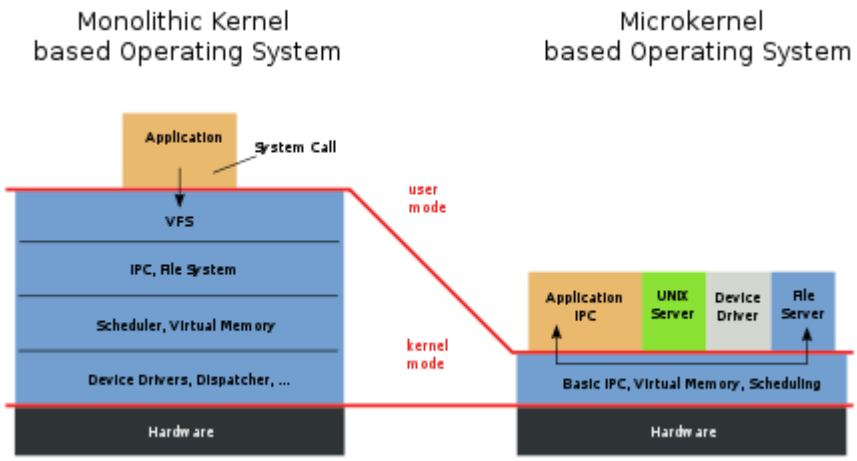


Fuente [12] Arquitectura Microkernel

Sistemas Monolitico vs Microkernel:

- Los sistemas operativos monolíticos son un solo archivo a.out que corre en modo kernel. Dicho archivo binario contiene todo: administración de procesos, de memoria, sistema de archivos, etc. Ejemplos de estos sistemas son UNIX, MS-DOS, VMS, MVS, OS/360, MULTICS y Linux.
- MINIX es un sistema basado en microkernel. EL sistema de archivo y la administración de memoria son procesos separados corriendo fuera del kernel. Los drivers de entrada y salida son también procesos separados.

Arquitecturas:



Fuente: Wikipedia, Microkernel [8]

Híbridos:

Es una arquitectura de sistemas operativos que combina aspectos de monolítico y micronúcleo (también se incluyen los casos de nanokernels y exokernels). El caso más popular de arquitectura híbrida es Windows NT dado que los subsistemas de emulación corren en procesos servidores en modo usuario en lugar de modo kernel como en la arquitectura monolítica. Adicionalmente, la mayor parte de los componentes del sistema corren en el mismo espacio de dirección de memoria que el kernel como en la arquitectura monolítica, mientras que en la arquitectura micronúcleo se usan espacios separados. [13]

Evolución de los sistemas operativos:

La evolución de los sistemas operativos se debe a actualizaciones de hardware, paginación, transferencia de páginas entre la memoria principal y secundaria, nuevos servicios, y las nuevas aplicaciones; correcciones, entre otras.

- **Procesamiento serial.** Los usuarios tienen acceso directo al CPU en serie. Problemas: planeamiento, tiempo de configuración (montar/desmontar cintas, iniciar todo el proceso cuando hay errores, etc.). Desperdicia tiempo del CPU. Es como si no hubiera OS.
-

- **Sistema de lotes simple** (Batch): usa un software residente en memoria (monitor) de manera que no haya acceso directo del usuario a la máquina.

    - Un operador recoge los trabajos secuencialmente en un dispositivo de entrada, para el uso del monitor.
    - Muestra como el monitor controla la secuencia de eventos: Se leen los trabajos desde un dispositivo de entrada, los colocan en el área de programas de usuario, y pasa el control a dicho trabajo.
    - Cuando éste ha terminado, el control es regresado al monitor, y los resultados son enviados a un dispositivo de salida.

    Hay menos tiempo vicioso, se mejora el tiempo de configuración a través de un lenguaje de control de trabajos. Este lenguaje provee instrucciones al monitor para manejar errores, control, etc. Problemas: no tiene protección de memoria – ningún programa de usuario debe alterar el área de memoria que contiene el monitor; temporizador (timer) – cada trabajo debe tener un tiempo limitado; instrucciones privilegiadas – algunas instrucciones deben ser ejecutadas solamente por el monitor; interrupciones. Es monoprogramado.

    CPU: P0 executes then E/S (CPU idle, waiting) P0…E/S…End…P1 can start execution

- **Sistema de lotes multiprogramado –** (ej. Mainframe systems) se busca mantener tanto el procesador como los dispositivos de E/S ocupados la mayor parte del tiempo. Debido a que los dispositivos de E/S son lentos comparados con el procesador, este tipo de sistema permite que el procesador sea intercambiado a otro trabajo mientras el programa de usuario en ejecución espera por la respuesta de un dispositivo E/S. Provee manejo de memoria y planificación.

    El hardware soporta interrupciones E/S y DMA (Direct Access Memory), de manera que el CPU pueda enviar un comando de E/S para un trabajo, proceder con la ejecución de otro mientras el comando de E/S es enviado al controlador de dispositivos. Cuando la operación de E/S es completada, el procesador es interrumpido y el control se pasa al programa manejador de interrupciones en el sistema operativo. Ej. WinXP

P0ex…E/S……..P0ex… Done.
P1 P1ex….E/S… P1ex
P2 P2ex……E/S

* Recordar los términos multiprogramación y monoprogramación; monoprocesador y multiprocesador.

- **Sistemas de tiempo compartido –** el tiempo del procesador es compartido entre múltiples usuarios. Múltiples usuarios accesan simultáneamente el sistema a través de terminales, con el sistema operativo intercalando la ejecución de cada programa de usuario en una pequeña computación. Los usuarios pueden estar desarrollando programas, ejecutando y usando distintas aplicaciones. (Pag. 66 Stallings). Ej. Windows 2003 Server, Red Hat.

- **Sistemas de transacciones de tiempo real –** (ej. Algunas líneas aéreas) parecido al concepto anterior, pero aquí es sólo una aplicación que puede ser consultada y actualizada por los usuarios, lo que puede ocasionar acceso al mismo registro simultáneamente.

Características de los sistemas operativos modernos:

Stallings [1] enumera ciertas características comunes en los sistemas operativos modernos.

→ Arquitectura **micronúcleo**.

→ **Multihilo.** Técnica que divide los procesos ejecutando una aplicación en hilos que pueden correr concurrentemente. Los procesos son colecciones de hilos y recursos del sistema asociados. Incrementa la velocidad de procesamiento en un monoprocesador.

→ **Multiprocesamiento simétrico (SMP).** Hay múltiples procesadores compartiendo el mismo espacio de memoria y E/S, interconectados por un bus común; todos los procesadores pueden realizar las mismas funciones. Es un sistema de un solo usuario con varios procesadores. SMP supera a los monoprocesadores en desempeño – ejecución en paralelo; hay mayor disponibilidad – si un procesador falla, el otro toma su lugar; crecimiento incremental – se pueden añadir más procesadores; escalabilidad – variedad de precios y características de acuerdo con

el # de procesadores. Se puede aplicar en máquinas de un solo hilo (un proceso = un hilo y varios procesos corren concurrentemente en distintos procesadores) o de varios hilos.

→ **Sistema distribuido.** Es un sistema en clusters de computadoras separadas cada una con su propia memoria principal, secundaria, y otros módulos de E/S. Provee la ilusión de un único espacio de memoria y una única memoria secundaria, y acceso a otras facilidades. Provee mecanismos unificados de acceso como por ejemplo un sistema de archivos distribuido. SMP y un sistema distribuido son características diferentes, la primera es monousuario y la segunda multiusuario. Ej. Sistema de archivos distribuidos, BDD.

→ **Diseño orientado a objetos**. Permite a los programadores personalizar un OS sin alterar la integridad del sistema cuando se agregan módulos.

Situaciones de Aprendizaje:

- Hacer un cuadro sinóptico de las características de Windows 2000 y de Unix. Qué incluye cada nivel?
- Leer y discutir a nivel técnico las estructuras Monolíticas, de micronúcleo e híbridas:

  http://en.wikipedia.org/wiki/Monolithic_kernel

  http://en.wikipedia.org/wiki/Microkernel

  http://linuxfinances.info/info/microkernel.html

  http://www.linuxjournal.com/article/6105

- Discutir el cuestionario:

1. Qué es un OS?
2. Cuál es el propósito del OS?
3. Qué hace el OS?
4. Describa los distintos tipos de OS.
5. Describa la secuencia de arranque normal de un computador
6. Qué es un proceso?
7. Qué son interrupciones?
8. Cómo se pueden clasificar las interrupciones?
9. Cuáles son los pasos del multiprocesamiento?
10. Qué significa thrashing?
11. Qué es un thread (hilo)?
12. Cuál es la diferencia entre OS Simétrico y Asimétrico?
13. Cuál es la función del administrador de memoria?
14. Qué es GUI?
15. Cuál es la diferencia entre API y interfaz de usuario?

Referencias

1. Stallings, William. Sistemas Operativos. V Edición. Prentice Hall. 2000.
2. Carretero, Sistemas Operativos: Una visión aplicada. Mc-Graw Hill. Segunda edición.
3. QNX, Wikipedia. http://en.wikipedia.org/wiki/QNX
4. RTLinux, Wikipedia: http://en.wikipedia.org/wiki/RTLinux
5. What is RTLinux: http://www.rtlinuxfree.com/
6. Windows Embedded CE 6.0 Documentation: http://msdn.microsoft.com/en-us/library/bb159115.aspx
7. Open Sources: Voices from the Open Source Revolution: http://oreilly.com/catalog/opensources/book/appa.html
8. Wikipedia, Microkernel: http://en.wikipedia.org/wiki/Microkernel
9. How operating system work: http://computer.howstuffworks.com/operating-system.htm/printable
10. Wikipedia, Shell (computing): http://en.wikipedia.org/wiki/Shell_(computing)
11. Wikipedia, Application Programming Interface: http://en.wikipedia.org/wiki/Application_programming_interface
12. Monolithic Architecture: http://www.cs.rit.edu/~hpb/Lectures/SIA/OS1/all-inOne-3.html
13. Hybrid Kernel: http://en.wikipedia.org/wiki/Hybrid_kernel