

Sistemas operativos

TEMA 1

INTRODUCCIÓN A LOS SISTEMAS OPERATIVOS

- Sergio García Mondaray -

Índice del tema

PARTE 1 – INTRODUCCIÓN

1. ¿Qué hace un sistema operativo?	01
1.1. Punto de vista del usuario	01
1.2. Punto de vista del sistema	01
1.3. Definición de sistema operativo	01
2. Organización de una computadora	01
2.1. Funcionamiento de una computadora	01
2.2. Estructura de almacenamiento	02
2.3. Estructura de E/S	02
3. Arquitectura de un sistema informático	02
3.1. Sistemas de un solo procesador	02
3.2. Sistemas multiprocesador	02
3.3. Sistemas en cluster	03
4. Estructura de un sistema operativo	03
5. Operaciones del sistema operativo	04
5.1. Operación en modo dual	04
5.2. Temporizador	04
6. Gestión de procesos	05
7. Gestión de memoria	05
8. Gestión de almacenamiento	05
8.1. Gestión del sistema de archivos	05
8.2. Gestión de almacenamiento masivo	05
8.3. Almacenamiento en caché	06
8.4. Sistemas de E/S	06
9. Protección y seguridad	06
10. Sistemas distribuidos	07
11. Sistemas de propósito general	07
11.1. Sistemas embebidos en tiempo real	07
11.2. Sistemas multimedia	07
11.3. Sistemas de mano	07
12. Entornos informáticos	08
12.1. Sistema informático tradicional	08
12.2. Sistema cliente-servidor	08
12.3. Sistema entre iguales	08
12.4. Sistemas basados en web	08

PARTE 2 – ESTRUCTURA DE SOs

1. Servicios del sistema operativo	09
2. Interfaz de usuario del sistema operativo	09
2.1. Intérprete de comandos	09
2.2. Interfaces gráficas de usuario	10
3. Llamadas al sistema	10
4. Tipos de llamadas al sistema	10
4.1. Control de procesos	10
4.2. Administración de archivos	10
4.3. Administración de dispositivos	10
4.4. Mantenimiento de información	11
4.5. Comunicaciones	11
5. Programas del sistema	11
6. Diseño e implementación del SO	12
6.1. Objetivos del diseño	12
6.2. Mecanismos y políticas	12
6.3. Implementación	12
7. Estructura del sistema operativo	12
7.1. Estructura simple	12
7.2. Estructura en niveles	12
7.3. Microkernels	13
7.4. En módulos	13
8. Máquinas virtuales	13
8.1. Implementación	14
8.2. Beneficios	14
8.3. Ejemplos	14
9. Generación de sistemas operativos	14
10. Arranque del sistema	15

PARTE 1 - INTRODUCCIÓN

Un sistema operativo es un programa que administra el hardware de una computadora y proporciona las bases para la ejecución de programas de aplicación. Actúa como intermediario entre el usuario y el hardware de la computadora.

1. ¿QUÉ HACE UN SISTEMA OPERATIVO?

Podemos ver un sistema informático como hardware, software y datos. El sistema operativo proporciona los medios para el adecuado uso de estos recursos. El SO no realiza ninguna función útil por sí mismo, sino que proporciona un entorno para que otros programas puedan trabajar.

1.1. Punto de vista del usuario

La visión del usuario varía según la interfaz que utilice. En la mayoría de casos el SO se diseña para un fácil uso, prestando cierta atención al rendimiento y ninguna a la utilización de recursos. En otros casos, cuando varios usuarios acceden a través de sus terminales a una mainframe, el SO se diseña para maximizar la utilización de recursos y asegurar que se usen eficientemente. En otros casos los usuarios usan estaciones de trabajo conectadas a redes de otras estaciones y servidores; estos usuarios tienen recursos dedicados, pero también compartidos, por lo que el SO se diseña para llegar a un compromiso entre usabilidad individual y utilización de recursos. Algunos casos especiales son los de SOs con poca o ninguna interacción con el usuario (lavadoras, automóviles...).

1.2. Vista del sistema

Un sistema informático tiene muchos recursos: tiempo de CPU, espacio en memoria, espacio de almacenamiento de archivos, dispositivos de E/S, etc. El SO actúa como el administrador de estos recursos, decidiendo a qué programa/usuario asignárselos frente a numerosas (y posiblemente conflictivas) solicitudes. Un SO es un programa de control: gestiona la ejecución de los programas de usuario, controlando especialmente la E/S.

1.3. Definición de sistema operativo

Los SO existen para hacer posible la ejecución de programas de usuario y poder resolver los problemas de un computador, puesto que esa es la finalidad de un computador. El hardware no es fácil de utilizar por sí sólo, por lo que se desarrollan programas de aplicación; estos necesitan de ciertas operaciones comunes como control de la E/S o asignación de recursos, estas necesidades se incorporan en el sistema operativo.

2. ORGANIZACIÓN DE UNA COMPUTADORA

2.1. Funcionamiento de una computadora

Una computadora de propósito general consta de una o más CPU y de una serie de controladoras de dispositivo conectadas, a través de un bus común, a la memoria. Al encender una computadora, se carga un programa de arranque (firmware), éste debe saber cómo cargar e iniciar la ejecución del SO.

La ocurrencia de algún suceso se indica mediante una interrupción, hardware o software. Las interrupciones por software son las excepciones, producidas por un error o una llamada al sistema. Cuando se interrumpe a la CPU, deja lo que está haciendo, ejecuta la rutina de servicio de la interrupción y, cuando ha terminado, reanuda la operación que estuviera haciendo. El

método más simple para que la interrupción transfiera el control a la rutina de servicio apropiada consiste en invocar una rutina genérica que examine la información de la interrupción y llame, a su vez, a la rutina específica para dicha interrupción. Sin embargo este método es lento, por lo que puede utilizarse otro sistema: disponer una tabla de rutinas de interrupción (vector de interrupciones) para llamar de forma indirecta a la rutina de interrupción concreta.

2.2. Estructura de almacenamiento

La memoria RAM es el único área de almacenamiento de gran tamaño a la que el procesador puede acceder directamente. En una arquitectura Von Neumann, un ciclo típico instrucción-ejecución extrae una instrucción de memoria y la almacena en el registro de instrucciones, la decodifica para extraer los operandos y almacenarlos en registros, y después ejecuta la instrucción, almacenando el resultado de nuevo en memoria. Los programas y datos están almacenados en almacenamiento secundario, ya que la memoria principal es pequeña y volátil.

2.3. Estructura de E/S

Normalmente los sistemas operativos disponen de un controlador (*driver*) de dispositivo para cada controladora hardware. Al iniciar una operación de E/S el controlador carga los registros apropiados de la controladora; ésta examina el contenido de estos registros para determinar qué acción realizar. La controladora inicia la transferencia de datos desde el dispositivo a su búfer local, al terminar produce una interrupción y el controlador devuelve el control al SO (con los datos, o un puntero a los mismos, si la operación ha sido de lectura; o información de estado si la operación ha sido de escritura). El problema de este sistema es el desperdicio de capacidad de proceso para datos grandes. El acceso directo a memoria (DMA) soluciona este problema: tras configurar el dispositivo E/S, la controladora transfiere un bloque entero de datos entre su búfer y la memoria sin que intervenga el CPU. De esta manera la CPU está disponible mientras tanto.

3. ARQUITECTURA DE UN SISTEMA INFORMÁTICO

3.1. Sistemas de un sólo procesador

En un sistema monoprocesador hay una CPU principal que ejecuta instrucciones de propósito general, incluyendo instrucciones de procesos de usuario. Pero casi todos los sistemas disponen de otros procesadores de propósito especial (como procesadores gráficos); o, en *mainframes*, de más procesadores de propósito general, como procesadores de E/S. Los procesadores de propósito especial ejecutan un conjunto limitado de instrucciones, nunca procesos de usuario. En ocasiones el SO los gestiona (enviando información sobre su siguiente tarea y monitorizando su estado). Este método libera a la CPU principal de trabajo adicional. Por ejemplo, un microprocesador incluido en una controladora de disco, que recibe una secuencia de solicitudes procedentes de la CPU, libera a ésta de tener que preocuparse de planificar las tareas de disco.

3.2. Sistemas multiprocesador

Estos sistemas disponen de dos o más procesadores que se comunican entre sí, compartiendo el bus de la computadora y, en ocasiones, el reloj, la memoria y los dispositivos periféricos. Presentan tres ventajas:

- a) Mayor rendimiento: se realiza más trabajo en menos tiempo (aunque la mejora en velocidad con N procesadores es menor que N).
- b) Economía de escala: los sistemas multiprocesador son más baratos que múltiples sistemas de un sólo procesador, ya que comparten periféricos, alimentación y almacenamiento.
- c) Mayor fiabilidad: el fallo de un procesador no hará que el sistema deje de funcionar. Si tenemos 10 procesadores y uno falla, el sistema trabajará un 10% más despacio, en lugar de dejar de funcionar.

La capacidad de seguir proporcionando servicio proporcionalmente al nivel de hardware superviviente se llama degradación suave. Los sistemas multiprocesador actuales son de dos tipos: multiprocesamiento asimétrico (en el que cada procesador se asigna a una tarea específica, un procesador maestro controla el sistema y el resto esperan instrucciones de éste) y multiprocesamiento simétrico (SMP) (en el que todos los procesadores son iguales y realizan las tareas correspondientes al SO. La ventaja es que se pueden ejecutar simultáneamente muchos procesos; la desventaja es que, al estar las CPUs separadas, una puede estar inactiva y la otra sobrecargada, dando lugar a ineficiencias).

Una tendencia actual es incluir múltiples núcleos de cálculo en un mismo chip. Dejando de lado las consideraciones sobre la arquitectura (caché, memoria y contienda del bus) el sistema operativo ve estas CPU con N núcleos como N procesadores estándar.

3.3. Sistemas en cluster

Estos utilizan múltiples CPU, pero se diferencian de los sistemas multiprocesador en que un sistema en cluster está formado por dos o más sistemas individuales acoplados, compartiendo almacenamiento a través de una red LAN. Se suelen usar para proporcionar un servicio de alta disponibilidad, es decir, que funcione incluso si uno o más sistemas fallan. Un cluster se puede estructurar de varias formas: cluster asimétrico (una máquina está en modo de espera mientras otra ejecuta las aplicaciones, monitorizándola, si dicho servidor falla, el *host* en espera pasa a ser el servidor activo) y cluster simétrico (dos o más *hosts* ejecutan aplicaciones y se monitorizan entre sí, obviamente es más eficiente), cluster en paralelo (permite que varios *hosts* accedan a los mismos datos) y clusters conectados a una red de área extensa. Otros son los clusters de base de datos (donde docenas de *hosts* pueden compartir la misma base de datos, incrementando el rendimiento y la fiabilidad).

4. ESTRUCTURA DE UN SISTEMA OPERATIVO

Uno de los aspectos más importantes de los SOs es la capacidad de multiprogramar. La multiprogramación incrementa el uso de la CPU organizando los trabajos de modo que ésta siempre tenga uno que ejecutar. Es importante porque en general, un sólo usuario no puede mantener la CPU o los dispositivos E/S ocupados continuamente. El tiempo compartido (o multitarea) es una extensión lógica de la multiprogramación: la CPU ejecuta múltiples trabajos conmutando entre ellos, pero tan frecuentemente que los usuarios pueden interactuar con cada programa mientras está en ejecución. Un SO de tiempo compartido permite que muchos usuarios compartan simultáneamente la computadora, puesto que el sistema cambia rápidamente de un usuario al siguiente, cada uno tiene la impresión de que el sistema completo está dedicado a él.

El tiempo compartido y la multiprogramación requieren mantener simultáneamente en memoria varios trabajos. Dado que la memoria principal suele ser pequeña para todos esos trabajos, éstos se suelen mantener inicialmente en el disco, en una cola de trabajos, esperando la asignación de

memoria. Si hay varios trabajos y no hay espacio para todos, el sistema debe hacer una selección, es lo que se llama planificación de trabajos. Además, si hay varios trabajos preparados para ejecutarse al tiempo, el sistema debe elegir, es lo que se denomina planificación de la CPU.

En un sistema de tiempo compartido, el sistema operativo debe asegurar un tiempo de respuesta razonable, lo que en ocasiones se hace a través de un mecanismo de intercambio, donde los procesos se intercambian entrando y saliendo de la memoria al disco. Un método más habitual es utilizar una memoria virtual, que permite ejecutar un proceso que no está completamente en memoria. La ventaja de éste último método es que permite ejecutar programas más grandes que la propia memoria física.

5. OPERACIONES DEL SISTEMA OPERATIVO

Los SOs modernos están controlados por interrupciones, si no hay ningún proceso que ejecutar, ningún dispositivo E/S al que dar servicio y ningún usuario al que responder, el SO permanece inactivo, esperando que algo ocurra. Estos sucesos se indican mediante la ocurrencia de una interrupción o una excepción (que es una interrupción generada por software, debido a un error o a una llamada al sistema). Dado que el SO y los usuarios comparten los recursos hardware y software, es muy importante asegurar que un error en un programa de usuario sólo genere problemas en dicho programa.

5.1. Operación en modo dual

Para asegurar la correcta ejecución del SO, tenemos que poder distinguir entre la ejecución de código del SO y del código definido por el usuario. Por ello, como mínimo necesitamos dos modos de operación: el modo usuario y el modo kernel. Un bit de modo se añade al hardware para diferenciar entre una tarea que se ejecute en nombre del SO y otra que se ejecute en nombre del usuario. El modo dual nos proporciona los medios para proteger el sistema operativo de los usuarios que puedan causar errores, y para proteger a unos usuarios de los errores de otros. Esta protección se consigue designando algunas instrucciones como instrucciones privilegiadas.

Las llamadas al sistema proporcionan los medios para que un programa de usuario pida al sistema operativo que realice tareas reservadas. Cuando se ejecuta una llamada al sistema, el hardware la trata como una interrupción software. El control pasa a través del vector de interrupción a una rutina de servicio del sistema operativo, y el bit de modo se establece en modo kernel.

5.2. Temporizador

Para asegurar que el SO mantenga el control sobre la CPU debemos, por ejemplo, impedir que un programa de usuario entre en un bucle infinito, o que no llame a los servicios del sistema y nunca devuelva el control al SO. Para ello generalmente se implementa un temporizador variable mediante un reloj de frecuencia fina y un contador. El SO configura el contador: cada vez que avanza el reloj, el contador se decrementa. Cuando el contador llega a 0, se produce una interrupción y el control pasa automáticamente al SO, que puede tratar la interrupción como un error o conceder más tiempo al programa.

6. GESTIÓN DE PROCESOS

Un proceso es un trabajo en ejecución en un sistema de tiempo compartido. Necesita ciertos recursos: tiempo de CPU, memoria, archivos y dispositivos E/S. Estos recursos se le proporcionan en el momento de crear el proceso o se le asignan mientras se está ejecutando. Cuando el proceso termina, el SO reclama todos los recursos utilizables. Cada sistema consta de una colección de procesos, algunos del SO y el resto de los usuarios. El SO es responsable de:

- a) Crear y borrar procesos
- b) Suspender y reanudar procesos
- c) Proporcionar mecanismos de sincronización de procesos
- d) Proporcionar mecanismos para la comunicación entre procesos
- e) Proporcionar mecanismos para el tratamiento de los interbloqueos

7. GESTIÓN DE MEMORIA

La memoria principal es fundamental, y es compartida por la CPU y los dispositivos de E/S. Para que la CPU procese unos datos de disco, dichos datos deben transferirse primero a la memoria principal. Para mejorar tanto la utilización de la CPU como la velocidad de respuesta de la computadora, ésta puede mantener varios programas en memoria, lo que crea la necesidad de mecanismos de gestión de la misma. El SO es responsable de:

- a) Controlar las partes de la memoria en uso y por parte de quién
- b) Decidir qué datos y procesos añadir o extraer de memoria
- c) Asignar y liberar espacio en memoria según sea necesario

8. GESTIÓN DE ALMACENAMIENTO

8.1. Gestión del sistema de archivos

Un archivo es una colección de información relacionada definida por su creador. Comúnmente, los archivos representan programas (tanto en formato fuente como objeto) y datos. El SO implementa el abstracto concepto de archivo gestionando los medios de almacenamiento masivos. Asimismo, los archivos normalmente se organizan en directorios para hacer más fácil su uso. El SO es responsable de:

- a) Creación y borrado de archivos
- b) Creación y borrado de directorios para organizar los archivos
- c) Soporte de primitivas para manipular archivos y directorios
- d) Asignación de archivos a los dispositivos de almacenamiento secundario.
- e) Copia de seguridad de los archivos en medios de almacenamiento no volátiles

8.2. Gestión de almacenamiento masivo

Como la memoria principal es demasiado pequeña para almacenar los datos y programas, y además de volátil, la mayoría de los sistemas informáticos modernos usan discos como principal medio de almacenamiento. Por tanto, la apropiada gestión del almacenamiento en disco tiene una importancia crucial. El SO es responsable de:

- a) Gestión del espacio libre
- b) Asignación del espacio de almacenamiento
- c) Planificación del disco

Las unidades de cinta magnética y de CD/DVD son dispositivos de almacenamiento terciario. Este almacenamiento no es crucial, pero también necesita ser gestionado (montar y desmontar medios, asignar y liberar los dispositivos, y migrar datos de almacenamiento secundario al terciario). Algunos SO realizan esta tarea, mientras que otros lo dejan en manos de programas de aplicación.

8.3. Almacenamiento en caché

La caché es un sistema de almacenamiento muy rápido, y de mucho menor tamaño que la RAM. Cuando necesitamos alguna información particular, primero comprobamos si está almacenada en caché, si lo está usamos dicha información, en caso contrario utilizamos la información original, colocando una copia en la caché bajo suposición de que pronto la necesitaremos. La mayoría de los sistemas disponen de una caché de instrucciones para almacenar las siguientes instrucciones en espera de ser ejecutadas. Sin una caché, la CPU tendría que esperar varios ciclos mientras las instrucciones son extraídas de la RAM. También la mayoría de sistemas disponen, por la misma razón, de una caché de datos o más. La transferencia de datos del disco duro a la memoria principal es una función controlada por el SO, por el contrario, el paso de datos de la caché a los registros de la CPU normalmente no.

En un entorno donde sólo se ejecuta un proceso al tiempo, un acceso a un dato (por ejemplo un número entero A) siempre se realiza a la copia situada en el nivel más alto de la jerarquía de memoria (registros). Sin embargo, en un entorno multitarea, en el que la CPU conmuta entre varios procesos, hay que tener cuidado para asegurar que, si varios procesos quieren acceder a A, cada uno obtenga el valor más reciente. La situación se complica en un entorno multiprocesador, donde A puede encontrarse en varias cachés al tiempo. En ese caso debemos asegurarnos de que una actualización de A en una caché se vea reflejada en el resto. Esto se denomina coherencia de caché. El caso más complejo es en un entorno distribuido, donde varias copias del mismo archivo pueden estar en diferentes computadoras.

8.4. Sistemas de E/S

Uno de los propósitos de un SO es ocultar al usuario las peculiaridades de los dispositivos hardware. El subsistema de E/S consta de varios componentes:

- a) Un componente de gestión de memoria que incluye almacenamiento en búfer, gestión de caché y gestión de colas.
- b) Una interfaz general para controladores de dispositivo.
- c) Controladores para dispositivos hardware específicos.

9. PROTECCIÓN Y SEGURIDAD

La protección es cualquier mecanismo que controle el acceso de procesos y usuarios a los recursos definidos por un sistema informático. Aunque un sistema tenga la protección adecuada puede estar expuesto a fallos y permitir accesos inapropiados, por ejemplo en el caso de que a un usuario le hayan robado su clave de acceso. Es responsabilidad de los mecanismos de seguridad defender al sistema frente a ataques como virus, gusanos, ataques de denegación de servicio, robo de identidad y robo de servicio. La prevención de algunos de estos ataques se considera una función del SO en algunos sistemas, mientras que e otros se deja a algún software adicional.

La protección y seguridad requieren que el sistema pueda distinguir a todos sus usuarios. La mayoría de SO mantienen una lista con los nombres y sus identificadores asociados. Cuando un usuario inicia sesión, la fase de autenticación determina el ID correspondiente, ese ID estará asociado con todos los procesos y hebras del usuario. En algunos casos es deseable diferenciar entre grupos de usuarios, esta funcionalidad de grupo se puede implementar manteniendo en el sistema una lista de nombres de grupo e identificadores. Un usuario puede pertenecer a uno o más grupos. En ocasiones el ID de usuario e ID de grupo no son suficientes, y un usuario necesita escalar sus privilegios para obtener permisos adicionales para una actividad.

10. SISTEMAS DISTRIBUIDOS

Un sistema distribuido es una colección de computadoras físicamente separadas que están conectadas en red para proporcionar a los usuarios acceso a los diversos recursos que el sistema mantiene. Acceder a un recurso compartido incrementa la velocidad de cálculo, la funcionalidad, la disponibilidad de los datos y la fiabilidad. Una red es una vía de comunicación entre dos o más sistemas (las redes se diferencian según el protocolo que usen, o la distancia entre sus nodos – LAN, WAN...–). La funcionalidad de los sistemas distribuidos depende de la red.

11. SISTEMAS DE PROPÓSITO GENERAL

11.1. Sistemas embebidos en tiempo real

Son las predominantes hoy en día. Se encuentran por todas partes, desde motores de automóviles y robots para fabricación, hasta magnetoscopios y hornos microondas. Estos sistemas tienen tareas muy específicas, y disponen de una interfaz de usuario muy limitada o nula. Casi siempre ejecutan SOs en tiempo real. Un sistema operativo en tiempo real se utiliza cuando se han establecido rígidos requisitos de tiempo en la operación de un procesador o del flujo de datos. SOs de tiempo real son los que controlan experimentos científicos, los de imágenes médicas, los de control industrial y ciertos sistemas de visualización.

11.2. Sistemas multimedia

La incorporación de datos multimedia (abarcando audio, video, una combinación de ambos, y archivos convencionales) en los sistemas es una tendencia reciente. Estos datos deben suministrarse cumpliendo ciertas restricciones de tiempo.

11.3. Sistemas de mano

Los sistemas de mano incluyen los PDA y teléfonos móviles. Los desarrolladores de aplicaciones y sistemas de mano se enfrentan a muchos retos, la mayoría debidos al tamaño limitado de dichos dispositivos. Debido a su tamaño, la mayoría de los dispositivos de mano tienen muy poca memoria, procesadores lentos (para ahorrar energía) y pantallas de visualización pequeñas. Debido a la baja velocidad de procesador, las aplicaciones y el SO tienen que diseñarse para no imponer una excesiva carga al procesador. El último problema es la E/S. La falta de espacio físico limita los métodos de entrada a pequeños teclados, sistemas de reconocimiento de escritura manual o pequeños teclados basados en pantalla. Las pequeñas pantallas de visualización limitan asimismo las opciones de salida.

12. ENTORNOS INFORMÁTICOS

12.1. Sistema informático tradicional

El típico entorno de oficina, de hace unos pocos años, constaba simplemente de varios PCs conectados mediante una red, con servidores que proporcionan servicios de archivos e impresión. Las empresas establecen portales, que proporcionan acceso web a sus servidores internos. Actualmente las computadoras de mano y PDA pueden sincronizarse con los PC y conectarse a redes inalámbricas para usar el portal web de la empresa.

Durante bastante tiempo, los sistemas estuvieron separados en dos categorías: sistemas de procesamiento por lotes (que procesaban trabajos masivos, con una entrada predeterminada) y sistemas interactivos (que esperaban la introducción de datos por parte del usuario). Para optimizar el uso de recursos, varios usuarios compartían el tiempo en estos sistemas, los sistemas de tiempo compartido empleaban un temporizador y algoritmos de planificación para ejecutar rápidamente una serie de procesos por turnos en la CPU.

12.2. Sistema cliente-servidor

Muchos sistemas actuales actúan como sistemas servidor para satisfacer las solicitudes generadas por los sistemas cliente. Esta forma de sistema distribuido especializado, se denomina sistema cliente-servidor. Los sistemas servidor, por su parte, puede clasificarse en:

- a) Servidor de cálculo: proporciona una interfaz a la que un cliente puede enviar una solicitud para realizar una acción, como leer datos; en respuesta, el servidor ejecuta la acción y devuelve el resultado al cliente.
- b) Servidor de archivos: proporciona una interfaz de sistema de archivos mediante la que los clientes pueden crear, actualizar, leer y eliminar archivos.

12.3. Sistema entre iguales

Es un modelo de sistema distribuido en el que los clientes y servidores no están diferenciados. Todos los nodos del sistema se consideran iguales, y pueden actuar como cliente o como servidor. Ofrecen una ventaja sobre la estructura cliente-servidor tradicional: en un sistema cliente-servidor, el servidor es un cuello de botella, pero en un sistema entre iguales, varios nodos distribuidos a través de la red pueden proporcionar los servicios.

12.4. Sistema basado en web

Los PC todavía son los dispositivos de acceso predominante, aunque las estaciones de trabajo, os PDA e incluso los teléfonos móviles se emplean ahora también para proporcionar acceso a Internet. La implementación de sistemas basados en la Web ha hecho surgir nuevas categorías de dispositivos, tales como los mecanismos de equilibrado de carga, que distribuyen las conexiones de red entre una batería de servidores similares. La Web ha incrementado la complejidad de muchos dispositivos, ya que los usuarios de los mismos exigen poder conectarlos a la Web.

PARTE 2 – ESTRUCTURAS DE SISTEMAS OPERATIVOS

1. SERVICIOS DEL SISTEMA OPERATIVO

El sistema presta ciertos servicios a los programas y usuarios, aunque estos servicios difieren de un SO a otro. Los más comunes son los siguientes:

- a) Interfaz de usuario: puede ser de varios tipos, por ejemplo por línea de comandos, de proceso por lotes, etc. Aunque la más habitual es la interfaz gráfica, compuesta por un sistema de ventanas, con un dispositivo señalador para elegir opciones en los menús.
- b) Ejecución de programas: el sistema tiene que poder cargar un programa en memoria y ejecutarlo. Todo programa debe poder terminar su ejecución, normal o anormalmente.
- c) Operaciones de E/S: el SO debe proporcionar medios para realizar tareas de E/S, ya que por protección y eficiencia los usuarios no pueden controlar directamente los dispositivos de E/S.
- d) Manipulación del sistema de archivos: los programas necesitan leer y escribir en archivos y directorios, crearlos y borrarlos, realizar búsquedas o presentar la información contenida en un archivo. Por último, algunos programas incluyen mecanismos de gestión de permisos para conceder o denegar el acceso a los archivos.
- e) Comunicaciones: hay muchas circunstancias en las que un proceso necesita intercambiar información con otro. Dicha comunicación puede tener lugar entre procesos que se ejecutan en la misma computadora o entre procesos que se ejecuten en computadoras diferentes conectadas a través de una red.

Hay disponible otro conjunto de funciones pensadas para garantizar la eficiencia del propio sistema:

- a) Asignación de recursos: cuando hay varios usuarios o trabajos ejecutándose al tiempo, deben asignarse a cada uno los recursos necesarios. El SO gestiona muchos tipos de recursos. Por ejemplo, para poder determinar cuál es el mejor modo de usar la CPU, el SO dispone de rutinas de planificación de la CPU, que tienen en cuenta la velocidad del procesador, los trabajos que tienen que ejecutarse, el número de registros disponibles...
- b) Responsabilidad: normalmente conviene hacer un seguimiento de qué usuarios emplean qué clase de recursos de la computadora y en qué cantidad. Este registro puede ser una herramienta valiosa para aquellos investigadores que deseen reconfigurar el sistema con el fin de mejorar sus servicios informáticos.

2. INTERFAZ DE USUARIO DEL SISTEMA OPERATIVO

2.1. Intérprete de comandos

Algunos sistemas operativos incluyen el intérprete de comandos en el kernel. Otros tratan al intérprete de comandos como un programa especial. En los SO que disponen de varios intérpretes de comandos entre los que elegir, los intérpretes se conocen como shells (algunos son shell Bourne, shell C, shell Bourne-Again, shell Korn...). La función del intérprete de comandos es obtener y ejecutar el siguiente comando especificado por el usuario. Los comandos pueden implementarse de dos maneras generales:

- a) El propio intérprete contiene el código que el comando tiene que ejecutar. En este caso, el número de comandos que puede proporcionarse determina el tamaño del intérprete de comandos.

- b) La mayoría de comandos están implementados a través de una serie de programas del sistema. En este caso, el intérprete no entiende el comando, sino que simplemente lo usa para identificar el archivo que hay que cargar en memoria y ejecutar. De esta forma los programadores pueden añadir comandos al sistema fácilmente, creando nuevos archivos con los nombres apropiados.

2.2. Interfaces gráficas de usuario

Una GUI (interfaz gráfica de usuario) permite a los usuarios emplear un sistema de ventanas y menús controlable mediante el ratón. Dependiendo de la ubicación del puntero, pulsar el botón puede invocar un programa, seleccionar un archivo o directorio o desplegar un menú que contenga comandos ejecutables.

3. LLAMADAS AL SISTEMA

Las llamadas al sistema generalmente están disponibles como rutinas escritas en C y C++, aunque determinadas tareas de bajo nivel pueden necesitar escribirse con ensamblador. La mayoría de programadores de aplicaciones diseñan sus programas utilizando una API (application programming interface). La API especifica un conjunto de funciones que el programador de aplicaciones puede usar, indicándose los parámetros que hay que pasar a cada función y los valores de retorno que el programador debe esperar. Una ventaja de programar usando una API es la portabilidad (un programa diseñado usando una API puede ser compilado y ejecutado en cualquier sistema que soporte la misma API).

Habitualmente, cada llamada al sistema tiene asociado un número y la interfaz de llamadas al sistema mantiene una tabla indexada según dichos números. Usando esa tabla, la interfaz de llamadas al sistema invoca la llamada necesaria del kernel del sistema operativo y devuelve el estado de la ejecución de la llamada al sistema y los posibles valores de retorno.

4. TIPOS DE LLAMADAS AL SISTEMA

4.1. Control de procesos

Un programa en ejecución necesita poder interrumpir dicha ejecución, bien de forma normal o bien de forma anormal. En cualquier caso el SO debe transferir el control al intérprete de comandos que realizó la invocación para que lea el siguiente comando. En un sistema GUI, una ventana alertará al usuario del error.

Algunas llamadas al sistema, en cuanto al control de procesos se refiere, son: terminar/abortar; cargar/ejecutar; crear/terminar procesos; obtener/definir atributos del proceso; asignar/liberar memoria; esperar/señalizar suceso...

4.2. Administración de archivos

Necesitamos crear y borrar archivos; abrir y cerrar archivos; leer, escribir y reposicionar; obtener y definir atributos de un archivo, etc. Lo mismo ocurre con directorios.

4.3. Administración de dispositivos

Un proceso puede necesitar varios recursos para ejecutarse: memoria principal, unidades de disco, acceso a archivos, etc. Si los recursos están disponibles, pueden ser concedidos, en caso contrario el proceso tendrá que esperar. Puede pensarse en los distintos recursos controlados por el SO como dispositivos, algunos físicos (como el espacio en memoria) y otros abstractos (como un archivo).

En cuanto a la administración de dispositivos, las principales llamadas al sistema son: solicitar o liberar dispositivos; leer, escribir y reposicionar; obtener y definir atributos de dispositivos; conectar y desconectar dispositivos lógicamente.

4.4. Mantenimiento de información

Muchas llamadas al sistema existen simplemente con el propósito de transferir información entre el programa de usuario y el sistema operativo. Por ejemplo la llamada al sistema para conocer la hora y la fecha, el número actual de usuarios, la versión del sistema operativo, cantidad de memoria libre, etc. Además, el SO mantiene información sobre todos sus procesos.

4.5. Comunicaciones

Existen dos modelos comunes de comunicación entre procesos:

- a) Modelo de paso de mensajes: los procesos se comunican enviándose mensajes con información, directa o indirectamente a través de un buzón de correo común. La mayoría de los procesos que reciben conexiones son de propósito especial y se denominan demonios. Los demonios son programas que se mantienen a la espera y despiertan cuando se establece una conexión. El origen de la comunicación, denominado *cliente*, y el receptor, *servidor*, intercambian mensajes usando las llamadas al sistema para leer y escribir.
- b) Modelo de memoria compartida: los procesos usan las llamadas al sistema para crear y obtener acceso a regiones de la memoria que son propiedad de otros procesos. Normalmente el sistema operativo intentaría evitar que un proceso acceda a la memoria de otro, por ello la memoria compartida entre dos o más procesos requiere que estos acuerden eliminar esa restricción. La forma de los datos y su ubicación son determinadas por parte de los procesos y no por el SO.

5. PROGRAMAS DEL SISTEMA

Los programas del sistema proporcionan un cómodo entorno para desarrollar y ejecutar programas. Algunos son simplemente interfaces de usuario para las llamadas al sistema; otros son más complejos. Pueden dividirse en:

- a) Administración de archivos: manipulan archivos y directorios.
- b) Información de estado: algunos solicitan la hora, fecha memoria o espacio en disco disponible... Otros proporcionan información detallada sobre rendimiento, inicios de sesión y mecanismos de depuración.
- c) Modificación de archivos: editores de texto para crear y modificar archivos almacenados en disco, programas para buscar un dato en el contenido de los archivos...
- d) Soporte de lenguajes de programación: compiladores, ensambladores, depuradores e intérpretes para los lenguajes de programación más habituales.
- e) Carga y ejecución de programas: cargadores absolutos o reubicables, editores de montaje, cargadores de sustitución y sistemas de depuración.
- f) Comunicaciones: mecanismos para crear conexiones virtuales entre procesos, usuarios y computadoras. Permiten enviar mensajes a las pantallas de otros, explorar páginas web, enviar mensajes de correo electrónico, iniciar una sesión remota o transferir archivos.

Además de estos programas de sistema, la mayoría de SO se suministran con programas de utilidad (llamados programas de aplicación o utilidades del sistema) para resolver problemas comunes o realizar operaciones frecuentes: exploradores web, procesadores y editores de texto, hojas de cálculo, compiladores, juegos...

6. DISEÑO E IMPLEMENTACIÓN DEL SISTEMA OPERATIVO

6.1. Objetivos del diseño

En el nivel más alto, el diseño de un SO se verá afectado por la elección del hardware y tipo de sistema (procesamiento por lotes, de tiempo compartido, monousuario o multiusuario, distribuido...). Más allá de este nivel, los requisitos se pueden dividir en dos grupos básicos:

- a) Requisitos del usuario: los usuarios buscan un sistema cómodo y fácil de utilizar y aprender, fiable y seguro, además de rápido.
- b) Requisitos del sistema: aquellas personas que tienen que diseñar, crear, mantener y operar el sistema necesitan flexibilidad, fiabilidad y eficiencia.

6.2. Mecanismos y políticas

Los mecanismos determinan cómo hacer algo, mientras que las políticas determinan qué hacer. Las políticas cambian de un sitio a otro o con el paso del tiempo, por lo que es deseable un mecanismo insensible a los cambios de política. Un cambio de política significaría entonces la redefinición de sólo unos determinados parámetros del sistema. Las decisiones sobre políticas son importantes para la asignación de recursos.

Los SO basados en microkernel llevan al extremo la separación entre mecanismos y políticas, implementando un conjunto básico de bloques independientes de las políticas concretas, y permitiendo que se añadan políticas y mecanismos más avanzados a través de módulos del kernel.

6.3. Implementación

Actualmente los SO están escritos en su mayor parte en lenguajes de alto nivel (sobre todo C, por ser el lenguaje con compiladores más eficientes, y por permitir acercarse más al hardware que otros lenguajes), aunque determinadas partes son escritas en ensamblador (las referentes a los controladores de dispositivos de E/S).

La ventaja de usar lenguajes de alto nivel es que el código puede escribirse más rápido, es más compacto y fácil de entender y depurar. Además cada día mejoran los compiladores, permitiendo mejorar el código simplemente mediante una recompilación. Las únicas desventajas se reducen a los requisitos de velocidad y de espacio de almacenamiento (aunque en los sistemas de hoy en día no son importantes).

7. ESTRUCTURA DEL SISTEMA OPERATIVO

7.1. Estructura simple

Existente en sistemas operativos que comenzaron siendo sistemas pequeños, simples y limitados, y crecieron más allá de su ámbito original. MS-DOS es un ejemplo, en el que las interfaces y niveles de funcionalidad no están separados. Por ejemplo, como el 8088 de Intel para el que fue escrito no proporcionaba modo dual ni protección, los programas de aplicación pueden acceder a las rutinas de E/S, lo que hace que sea un sistema vulnerable.

7.2. Estructura en niveles

Con el soporte hardware apropiado, los sistemas operativos pueden dividirse en partes más pequeñas. Los implementadores tienen más libertad para cambiar el funcionamiento interno del sistema y crear sistemas operativos modulares. En una estructura de niveles, el SO se divide en capas. El nivel inferior es el hardware, y el nivel superior es la interfaz de usuario. Un nivel

intermedio consta de estructuras de datos y de un conjunto de rutinas que los niveles superiores pueden invocar. A su vez, dicho nivel intermedio puede invocar operaciones sobre los niveles inferiores.

La principal ventaja es la simplicidad de construcción y depuración: el primer nivel puede depurarse sin afectar al resto, dado que sólo usa el hardware básico para implementar sus funciones; una vez depurado su funcionamiento se supone correcto y se puede pasar a depurar el siguiente nivel. Si se encuentra un error durante la depuración de un nivel, el error tendrá que estar en dicho nivel, pues los niveles inferiores ya se depuraron. La principal dificultad es la de definir apropiadamente los niveles. Otro problema es que la implementación por niveles es menos eficiente, ya que cuando un programa de usuario realiza una llamada al sistema, el nivel correspondiente la captura (el nivel E/S si la llamada es para una operación E/S) y llama a su vez al nivel de gestión de memoria, que a su vez llama al de planificación de CPU... añadiendo cada nivel una carga de trabajo adicional a la llamada al sistema. En los diseños más recientes, para suplir ese defecto, se utiliza un menor número de niveles con más funcionalidad.

7.3. Microkernels

Este método estructura el SO eliminando todos los componentes no esenciales del kernel e implementándolos como programas del sistema y de usuario. El resultado es un kernel más pequeño. La funcionalidad principal del microkernel es proporcionar un mecanismo de comunicaciones (con el sistema de paso de mensajes) entre el programa cliente y los distintos servicios que se ejecutan en el espacio de usuario. Por ejemplo, si el programa cliente desea acceder a un archivo, debe interactuar con el servidor de archivos; pues bien, el programa cliente y el servicio se comunican de forma indirecta intercambiando mensajes a través del microkernel.

La ventaja principal es la facilidad para ampliar el sistema operativo. Todos los servicios nuevos se añaden al espacio de usuario, por lo que no requieren modificación del kernel. El microkernel proporciona mayor seguridad y fiabilidad, dado que la mayor parte de los servicios se ejecutan como procesos de usuario en lugar de como procesos del kernel (si un servicio falla, el resto del SO no se ve afectado). El inconveniente es que este sistema ofrece un peor rendimiento, debido a la carga de procesamiento adicional impuesta por las funciones del sistema.

7.4. En módulos

Quizá la mejor metodología actual para diseñar sistemas operativos sea la programación orientada a objetos para crear un kernel modular. En este caso, el kernel dispone de un conjunto de componentes fundamentales y enlaza dinámicamente los servicios adicionales. Tal estrategia utiliza módulos que se cargan dinámicamente y resulta habitual en las implementaciones modernas de UNIX. El resultado global es similar a un sistema de niveles, en el sentido de que cada sección del kernel tiene interfaces bien definidas y protegidas, pero es más flexible que un sistema de niveles, porque cualquier módulo puede llamar a cualquier otro. Además, el método es similar a la utilización de un microkernel, ya que el módulo principal sólo dispone de las funciones esenciales. Sin embargo, es más eficiente que un microkernel, ya que los módulos no necesitan invocar un mecanismo de paso de mensajes para comunicarse.

8. MÁQUINAS VIRTUALES

La idea fundamental que subyace a una máquina virtual es la de abstraer el hardware de la computadora, formando varios entornos de ejecución diferentes, y creando así la ilusión de que cada entorno está operando en su propia computadora privada. Con los mecanismos de planificación de la CPU y las técnicas de memoria virtual un SO puede crear la ilusión de que un proceso tiene su propio procesador con su propia memoria. Normalmente un proceso utiliza características adicionales, como llamadas al sistema y un sistema de archivos, que el hardware

básico no proporciona. El método de máquina virtual proporciona una interfaz que es idéntica al hardware básico subyacente. Cada proceso dispone de una copia (virtual) de la computadora subyacente. Una de las principales dificultades del método de máquina virtual son los sistemas de disco.

8.1. Implementación

Hay que tener un modo usuario y un modo kernel virtuales, ejecutándose ambos en modo usuario físico. Las acciones que dan lugar a la transferencia del modo usuario al modo kernel en una máquina real también tienen que hacer que se pase del modo usuario virtual al modo kernel virtual en una máquina virtual. Cuando el monitor de la máquina virtual obtiene el control, puede cambiar el contenido de los registros y del contador de programa para que la máquina virtual simule el efecto de la llamada al sistema. La principal diferencia es el tiempo: la E/S virtual puede tardar más (porque es interpretada) o menos tiempo (puesto que se pone en cola) que la E/S real.

8.2. Beneficios

Cada máquina virtual está completamente aislada de las demás, por lo que no existen problemas de protección. Sin embargo no es posible la compartición directa de recursos, por lo que se han implementado dos métodos para permitir esta compartición: compartir un minidisco y definir una red de máquinas virtuales (que puedan compartir almacenamiento). Un sistema de máquina virtual es perfecto para la investigación y desarrollo de SO. Modificar un sistema operativo es una tarea complicada, dado que se ejecuta en modo kernel. Por tanto es necesario probar todos los cambios con sumo cuidado, y manteniendo el equipo ocupado con dicha tarea (tiempo de desarrollo del sistema). Una máquina virtual elimina gran parte de ese problema, puesto que los programadores pueden desarrollar el sistema dentro de la máquina virtual en lugar de en la máquina física.

8.3. Ejemplos

- a) VMware: se ejecuta como una aplicación sobre un sistema operativo host y permite al sistema ejecutar de forma concurrente varias máquinas virtuales independientes.
- b) Máquina virtual Java: es una especificación de una computadora abstracta. Consta de un cargador de clases y de un intérprete Java, que ejecuta el código intermedio de este lenguaje. La JVM puede implementarse sobre un sistema operativo, como parte de un explorador web, o bien por hardware (con un chip específicamente diseñado).

9. GENERACIÓN DE SISTEMAS OPERATIVOS

Lo habitual es que los SO se diseñen para ejecutarse en cualquier clase de máquina. El sistema debe entonces configurarse para cada computadora. Este proceso se denomina generación del sistema. Para generar un sistema hay que determinar cierta información: tipo de CPU que se usará, cantidad de memoria disponible, dispositivos instalados y opciones del sistema operativo que se desean. Una vez determinada esta información, puede utilizarse de varias formas:

- a) Un administrador de sistemas puede usarla para modificar una copia del código fuente del SO y, a continuación, compilar el sistema operativo completo.
- b) Con la información obtenida se crean una serie de tablas y se seleccionan los módulos apropiados de una biblioteca precompilada. Estos módulos se montan para formar el sistema operativo final.
- c) Es posible construir un SO controlado completamente por tablas, realizándose la selección en tiempo de ejecución en lugar de en tiempo de compilación o montaje.

10. ARRANQUE DEL SISTEMA

Para que un sistema informático empiece a funcionar, la CPU debe inicializarse e iniciar la ejecución del programa de arranque implementado en firmware. El programa de arranque puede ejecutar directamente el SO si éste se encuentra también en el firmware, o puede completar una secuencia en la que progresivamente se cargan programas más inteligentes desde el firmware y el disco, hasta que el propio sistema operativo se carga en memoria y se ejecuta.