

Sistemas Operativos

Tema 1: Introducción a los OS.

1.2. ¿Qué es un sistema operativo?

Definición histórica: Programa en ejecución que administra el hardware de una computadora.

Definición actual: Aquel programa que se ejecuta continuamente en una computadora (kernel), los demás son programas del sistema y de aplicación.

Proceso: Programa cargado en memoria principal.

Hay 2 tipos de programas:

- Programas de sistema: Controlan las operaciones propias de la computadora
- Programas de aplicación: Resuelven problemas específicos a los usuarios.

El objetivo inicial del OS es proporcionar una interfaz intermediaria entre la máquina desnuda o hardware y el usuario, de forma que se puedan gestionar los recursos de la computadora. El OS se encarga de ocultar la complejidad del hardware. El OS debe disponer de CPU para ejecutarse, de forma que el propio procesador no lo distingue del resto de procesos.

1.1. El OS como gestor/asignador de recursos

Al enfrentarse a numerosas y conflictivas solicitudes de recursos, el OS debe decidir como asignar éstos a programas y usuarios específicos, de modo que la computadora opere de forma eficiente y equitativa. Un OS es un programa de control que se encarga de gestionar la ejecución de los programas de usuario para evitar errores y mejorar el uso de la computadora.

Cualquier asignación de recursos y política de planificación debe tener en cuenta 3 factores:

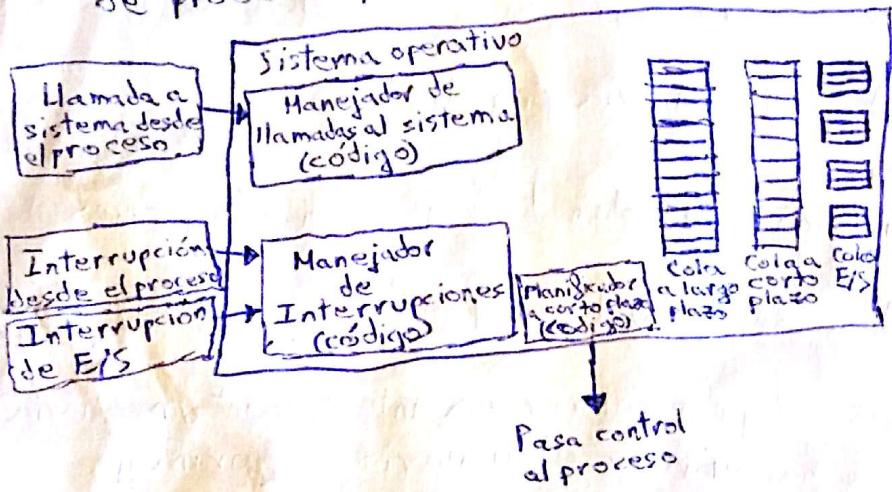
• Equitatividad: Normalmente, se desea que todos los procesos que compiten por un determinado recurso, se les conceda un acceso equitativo a dicho recurso. Esto es especialmente cierto para trabajos de la misma categoría.

• Respuesta diferencial: El OS debe tomar las decisiones de asignación y planificación con el objetivo de satisfacer el conjunto total de requisitos.

• Eficiencia: Debe maximizar la productividad, minimizar el tiempo de respuesta, y, en caso de sistemas de tiempo compartido

acomodar el máximo número de usuarios posible. Medir la actividad del sistema es importante para ser capaz de monitorizar el rendimiento y realizar los ajustes correspondientes.

Principales elementos del OS relacionados con la planificación de procesos y la asignación de recursos.



La cola a corto plazo está compuesta por procesos que se encuentran en memoria principal y están listos para ejecutar.
La cola a largo plazo es una lista de nuevos trabajos esperando a utilizar el procesador. El OS añade trabajos al sistema transfiriendo un trabajo desde la cola a largo plazo hasta la cola a corto plazo. Se debe asignar una porción de memoria principal al proceso entrante.

Hay una cola de E/S por cada dispositivo de E/S. Más de un proceso puede solicitar el uso del mismo dispositivo de E/S. El OS debe determinar a qué proceso asigna un dispositivo de E/S disponible.

1.2. El OS como máquina virtual

Una de las funciones del OS es presentar al usuario el equivalente de una máquina virtual que es más fácil de programar que el hardware subyacente. Una máquina virtual es aquella que presenta una mayor facilidad de uso incluyendo toda su funcionalidad.

El OS tiene que proporcionar servicios para las funciones siguientes:

- Creación de programas.
- Ejecución de programas: Funciones previas tales como cargar el código y los datos en la memoria principal, y preparar los recursos necesarios para la ejecución.
- Operaciones de E/S.

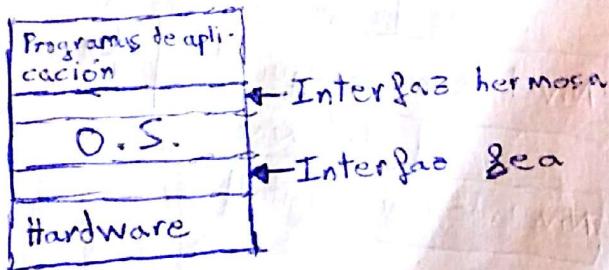
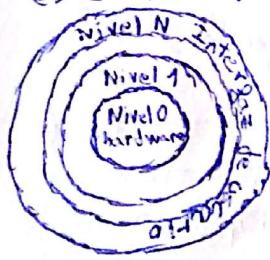
- Manipulación y control del sistema de archivos: Además de comunicarse con el controlador del periférico, y proporcionar los mecanismos adecuados para su control y protección.
- Detección de errores: El OS debe ser capaz de detectarlos y solucionarlos.
- Control de acceso al sistema: En sistemas públicos, el OS debe controlar el acceso al mismo, vigilando quién tiene acceso y a qué recursos.
- Elaboración de informes estadísticos: Resulta muy conveniente conocer el grado de la utilización de los recursos, configuraciones y tiempo de respuesta. Se dispone de información que permite saber con antelación las necesidades futuras y configurar al sistema para dar el mejor rendimiento.

2. Máquina multinivel o máquina virtual en capas.

El OS puede mantener un control mucho mayor sobre la computadora y sobre las aplicaciones que hacen uso de dicha computadora.

Con el método de diseño arriba-abajo, se determinan las características y la funcionalidad globales y se separan en componentes. La ocultación de los detalles a ojos de los niveles superiores deja libres a los programadores para implementar las rutinas de bajo nivel como prefieran.

Un sistema puede hacerse modular mediante una estructura en niveles, en el que el OS se divide en una serie de capas. El nivel inferior es el hardware; el nivel superior es la interfaz de usuario.



Un nivel de un OS típico consta de estructuras de datos y de un conjunto de rutinas que los niveles superiores pueden invocar. El nivel M puede invocar operaciones sobre los niveles inferiores. Llamamos máquina multinivel a una estructuración en capas bajo una serie de abstracciones, donde cada capa se apoya en la que está debajo, y facilita el trabajo con el OS.

Los niveles se seleccionan de modo que cada uno usa funciones y servicios de los niveles inferiores. Simplifica la depuración y la

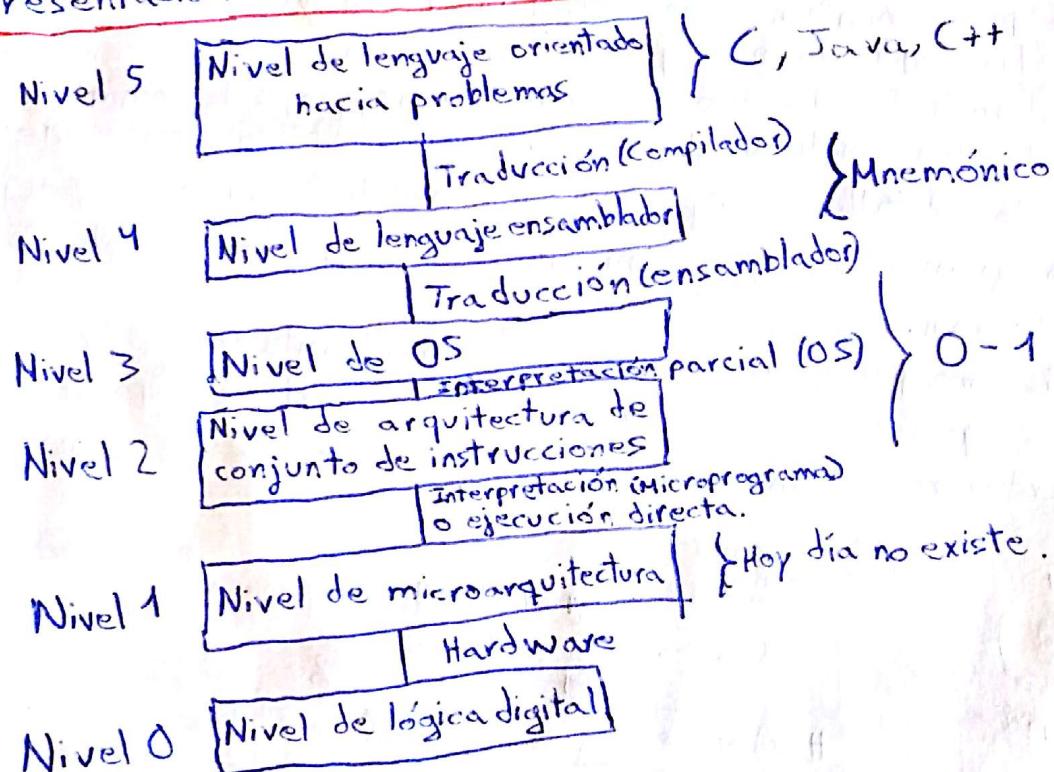
verificación del sistema. Si se encuentra un error en la depuración de un determinado nivel, el error se encuentra en ese nivel.

Cada nivel se implementa usando las operaciones proporcionadas por niveles inferiores. Un nivel solo necesita saber qué hacen esas operaciones.

La principal dificultad es la de definir apropiadamente los diferentes niveles. Las implementaciones por niveles tienden a ser menos eficientes que otros tipos de implementación. En cada nivel se pueden modificar los parámetros, puede ser necesario pasar datos. Cada nivel añade una carga de trabajo adicional a la llamada al sistema.

Las máquinas interpretan más fácilmente las señales on y off. Las instrucciones que ejecutan los computadores son colecciones de bits que pueden ser vistos como números.

Representación de una máquina multinivel



2.1. Lenguaje de alto nivel.

Con el objetivo de expresar los programas en un lenguaje más cercano a la forma de pensamiento de los programadores, surgen los lenguajes de alto nivel (Nivel 5). Para que los computadores puedan procesar estos nuevos tipos de lenguajes se desarrollaron otros programas que traducen a un lenguaje de más bajo nivel llamado ensamblador, hablamos de los compiladores.

2.2. Ensamblador

Es un tipo de programa informático que se encarga de traducir un fichero fuente escrito en ensamblador (Nivel 4) mediante instrucciones mnemónicas a un fichero objeto que contiene código máquina (Niveles 3 y 2). Según la arquitectura del microprocesador y el número de registros de éste, los mnemónicos y la traducción a lenguaje máquina difieren de unas a otras. Hay un ensamblador para cada tipo de procesador.

En los niveles 3 y 2 existen las mismas instrucciones máquina, la diferencia está en que en el nivel 3 el OS debe hacer determinadas reorganizaciones de las instrucciones a nivel de memoria.

El funcionamiento del OS está codificado a nivel de instrucciones en código máquina. Este código codificado en instrucciones máquina se ejecuta en modo núcleo por lo que es inaccesible al usuario, el cual solo puede hacer cambios en el nivel 5 o como mucho en el 4. Esto da lugar a lo que se llama modo núcleo y modo usuario.

2.3. Microprogramación

Dos técnicas conducen a la clasificación de los microprocesadores en dos grupos:

- Microprocesadores "cableados": Tienen una unidad de control específicamente diseñada sobre el silicio.

- Microprocesadores "microprogramados": Tienen una unidad de control genérica o prediseñada defendiendo de un microprograma software.

En las máquinas en las que no existe el nivel de microprogramación se pasa del nivel 2 al nivel 0. Hoy día la microprogramación ha desaparecido prácticamente por completo. Factores a los que se debe la desaparición:

- Ya existen herramientas avanzadas para diseñar complejas unidades de control con millones de transistores litografiados.

- Las unidades de control cableadas tienen un rendimiento significativamente mayor que cualquier unidad microprogramada.

Si un sistema necesita ser microprogramado hay un nivel más entre el 2 y el hardware. El lenguaje máquina se interpreta del nivel 2 al 1 por un programa llamado microprograma. El microprograma transforma el lenguaje máquina en microinstrucciones con un determinado formato de 1 y 0, adaptando instrue-

ciones de un propósito más general a microinstrucciones específicas que dependen de la arquitectura. Constituye un interprete entre el lenguaje máquina del nivel 2 y el hardware del microprocesador (Nivel 0).

3. Elementos básicos y organización de un computador.

Un computador digital electrónico consta de:

- CPU: Contiene una unidad aritmético lógica y registros del procesador. Controla el funcionamiento del computador y realiza sus funciones de procesamiento de datos.

- Memoria principal: Almacena datos e instrucciones, es volátil. También se le denomina memoria real o primaria.

- Módulos o controladoras E/S: Transfieren los datos entre el computador y su entorno externo.

Estos componentes se interconectan de manera que se pueda lograr la función principal del computador. Para su unión se usan los buses del sistema.

Una de las funciones del procesador es el intercambio de datos con la memoria. Para ello se usan registros internos al procesador:

- Registro de dirección de memoria (RDIM): Especifica la dirección de memoria de la siguiente lectura o escritura.

- Registro de datos de memoria: contiene los datos que se van a escribir en la memoria o que recibe los datos leídos de la memoria.

- Registro de datos de E/S (RDAE/S): intercambia datos entre un controlador de entrada/salida y el procesador.

Un módulo de memoria consta de un conjunto de posiciones. Cada posición contiene un patrón de bits que se puede interpretar como una instrucción o como datos.

Un módulo o controladora de E/S contiene buffers. Cada controladora de dispositivo se encarga de un tipo específico de dispositivo. La CPU y las controladoras de dispositivos pueden funcionar de forma concurrente compitiendo por los ciclos de memoria.

3.1. Registros del procesador.

- Registros visibles para el usuario. Permiten al programador en lenguaje máquina o en ensamblador minimizar las referencias a memoria principal optimizando el uso de registros.
- Registros de control y estado (no visibles para el usuario). Son usados por el procesador para controlar su operación y por rutinas privilegiadas del OS para controlar la ejecución de programas. Diferentes máquinas tendrán diferentes organizaciones de registros y utilizarán diferente terminología.

Además de los registros RDIRM, RDAM, RDIE/S y RDAE/S, tenemos:

- Controlador de programa (PC): Dirección de la próxima instrucción que se leerá de la memoria.
- Registro de instrucción: Contiene la última instrucción leída.
- Registro de resultado: Contiene también un registro conocido como la palabra de estado del sistema (PSW). Contiene 1 bit para habilitar/inhabilitar las interrupciones, un bit de modo usuario/supervisor y bits de códigos de condición, bits cuyo valor lo asigna el hardware del procesador teniendo en cuenta el resultado de operaciones. Además de almacenarse un resultado en el registro o en la memoria, se fija también un código de condición en concordancia con el resultado de la ejecución de la instrucción aritmética.

3.2. Estructura de almacenamiento.

Los programas de la computadora deben hallarse en la memoria principal para ser ejecutados. La memoria principal es el único área de almacenamiento de gran tamaño a la que el procesador puede acceder directamente. La interacción se consigue a través de una secuencia de carga o almacenamiento de instrucciones en direcciones específicas de memoria. La CPU carga automáticamente instrucciones desde la (FP) memoria principal para su ejecución.

Idealmente, es deseable que los programas y los datos residan en la memoria de forma permanente. Usualmente, esta situación no es posible por las 2 razones siguientes:

- La memoria principal es demasiado pequeña para almacenar todos los programas y datos permanentemente.
- La memoria principal es un dispositivo de almacenamiento volátil.

La mayor parte de los sistemas informáticos proporciona almacenamiento secundario. El requerimiento de este almacenamiento secundario es que se tienen que poder almacenar grandes cantidades de datos de forma permanente. El más común es el disco magnético que almacena tanto programas como datos.

Las principales diferencias entre los distintos sistemas de almacenamiento están relacionadas con la velocidad, el coste, el tamaño y la volatilidad.

3.3 Estructura de Entrada/Salida

Una computadora de propósito general consta de una o más CPU y de múltiples controladoras de dispositivo que se conectan a través de un bus común. Dependiendo de la controladora puede haber más de un dispositivo conectado. Dos más dispositivos pueden estar conectados a una controladora SATA o una controladora USB. La controladora del dispositivo es responsable de transferir los datos entre los dispositivos periféricos que controla y su búfer local. Un controlador puede estar o no integrado en placa base.

Los OS tienen un controlador de dispositivo software (driver) para cada controladora de dispositivo. Es muy importante no confundir estos conceptos, el primero es un componente software que se integra en el OS para comunicarse con el correspondiente dispositivo, y el segundo es un elemento físico.

Operación de E/S: La controladora inicia la transferencia de datos desde el dispositivo a su búfer local. Una vez completada la transferencia de datos, la controladora hardware informa al control de dispositivo a través de una interrupción. El controlador devuelve entonces el control al OS, devolviendo posiblemente los datos, o un puntero a ellos, si la operación ha sido una lectura.

Esta forma de E/S controlada por interrupción resulta adecuada para transferir cantidades pequeñas de datos, pero

representa un desperdicio de capacidad de procesamiento cuando se usa para movimientos masivos de datos. Para resolver este problema, se usa el acceso directo a memoria. Después de configurar buffers, punteros y contadores para el dispositivo de E/S, la controladora hardware transfiere un bloque (entero) entero de datos en su propio búfer y la memoria sin que intervenga la CPU.

4. Ejecución de instrucciones

Un programa que va a ejecutarse en un procesador consta de 2 pasos: el procesador lee, busca instrucciones de la memoria, y ejecuta cada una de ellas. La ejecución de la instrucción puede involucrar varias operaciones dependiendo de la naturaleza de la misma.

Ciclo de instrucción al procesamiento requerido por una única instrucción. La ejecución del proceso se detiene solo si se apaga la máquina, se produce algún error o se ejecuta una instrucción del proceso que para el procesador.

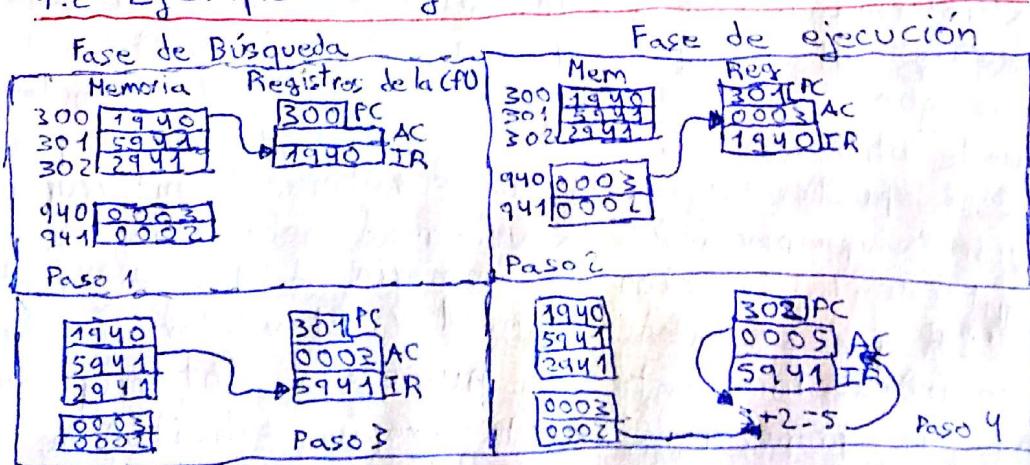
4.1. Búsqueda y ejecución de una instrucción

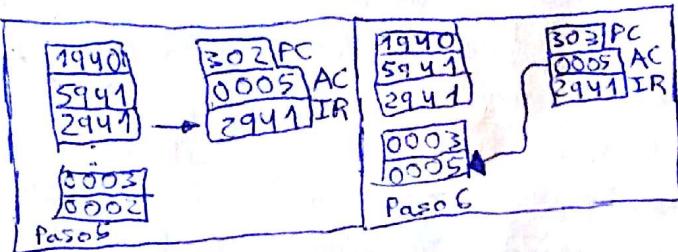
En un procesador típico, el contador del programa almacena la dirección de la siguiente instrucción que se va a leer.

Las acciones que se pueden realizar con una instrucción se dividen en 4 categorías:

- Procesador-memoria: Transferir datos desde el procesador a la memoria.
- Procesador-E/S: Se pueden enviar datos a un dispositivo periférico o recibirlas desde el mismo.
- Control. Una instrucción puede especificar que se va a alterar la secuencia de ejecución.

4.2 Ejemplo de ejecución de instrucciones





5. Interrupciones

Interrupciones hardware:

- Interrupciones de E/S: Generada para señalar la conclusión normal de una operación o para indicar diversas condiciones de error
- Interrupciones por fallo de hardware: Por cortes en el suministro de energía o zonas corruptas de memoria

Interrupciones software:

- Interrupciones de programa: Generadas por alguna condición que se produce como resultado de la ejecución de una instrucción. Se les suele llamar excepciones.
- Interrupciones por temporizador: Se pueden generar por un temporizador del procesador, y permiten al OS realizar ciertas funciones de forma regular.

Cuando se interrumpe a la CPU, ésta deja lo que está haciendo e inmediatamente transfiere la ejecución a una posición de memoria principal fijada y establecida. Dicha posición contiene la dirección de inicio donde se encuentra una rutina de servicio a la interrupción. La ISR trata la interrupción y, cuando ha terminado, la CPU reanuda la operación que estuviera haciendo.

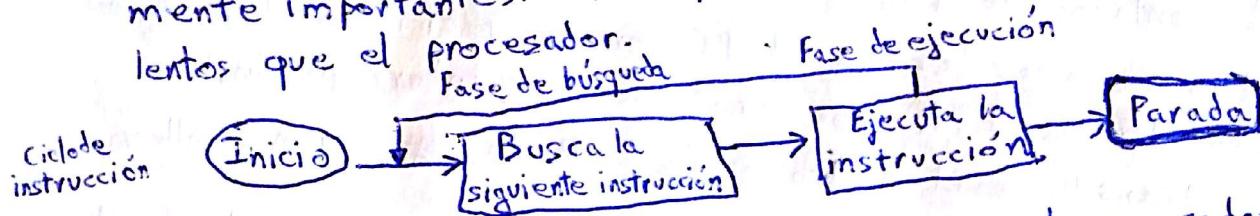
Funciones comunes en el diseño de computadoras: La interrupción debe transferir el control a la ISR apropiada. La transferencia debe consistir en invocar a una rutina ISR genérica para revisar la información de la interrupción; esa rutina genérica debe llamar a la rutina específica de tratamiento de la interrupción, según ^{con} sea el origen que la produce. Es un método algo lento. Puede utilizarse otro sistema, consistente en disponer de una tabla de punteros a las diferentes rutinas de interrupción, con el fin de proporcionar la velocidad necesaria. Cada elemento contiene la dirección de memoria de las diferentes rutinas ISR que se pueden ejecutar. El vector de interrupciones se indexa mediante un número de interrupción único, que se proporciona en la propia solicitud de interrupción para obtener la dirección de la ISR específica para el tratamiento de la interrupción que se ha producido.

Windows y UNIX manejan las interrupciones de este modo.

Cuando estamos hablando de interrupción, el procesador está realizando una tarea concreta y de pronto se ve interrumpido por su propia tarea u otro tipo de suceso. Otra cosa es invocar una operación de E/S, que resulta en una "llamada al sistema", pero eso no es una interrupción. Tienen en común el salvado de contexto del proceso que se está ejecutando, que también se produce en una llamada al sistema.

5.1. A la espera de interrupciones de E/S.

El tratamiento de las interrupciones de E/S son tremenda-mente importantes. Los dispositivos de E/S son mucho más lentos que el procesador.



Después de cada interrupción de escritura el procesador debe parar y permanecer inactivo hasta que la impresora lleve a cabo su labor. La longitud de la pausa puede ser de millones de ciclos de interrupción.

Debido a que la operación de impresión, puede tardar un tiempo, el procesador se queda esperando a que termine, para ello, el programa de usuario se detiene en la llamada a escritura, "desperdiéndose" el uso de la CPU hasta que se haya impreso en el papel.

Cuando termina de imprimir, lo indica mandando una petición de interrupción al OS para que sea atendida por una ISR. Se puede producir una interrupción en cualquier punto de la ejecución de un programa, dicha interrupción se produce en dirección desde el dispositivo externo hacia el procesador. Cuando se completa el tratamiento por la ISR se reanuda la ejecución por donde iba.

El procesador y el OS son responsables de suspender el programa de usuario y reanudarlo en el mismo punto.

En muchos casos, el procesador puede que no siga por el proceso que estaba ejecutando.

Debemos añadir la fase de interrupción al ciclo de instrucción.

En la fase de interrupción, el procesador comprueba si se ha producido cualquier interrupción. Si la hay, suspende la ejecución del proceso actual y ejecuta una ISR que realiza las acciones que se requieran, volviéndose después a reanudar la ejecución del proceso de usuario en el punto de la interrupción.

5.2. Salvado del contexto de un proceso debido a una interrupción

Veamos en detalle qué es un salvado de contexto.

1- Necesita salvar la palabra de estado del programa y la posición de la siguiente instrucción que se va a ejecutar, contenida en el contador de programa (PC).

2- El procesador carga el PC con la posición del punto de entrada de la ISR que tratará esa interrupción.

3- Se transfiere el control a la ISR. La ejecución conlleva

las siguientes operaciones:

- 3.1. El PC y la PSW vinculados con el proceso interrumpido se han almacenado en la pila del sistema. Se necesita salvar el contenido de los registros del procesador, puesto que los podría usar la ISR.

3.2. La ISR puede comenzar a procesar la interrupción. Si es de E/S puede implicar el envío de datos del dispositivo de E/S hacia la CPU y/o memoria principal.

3.3. Cuando se completa el procesamiento de la interrupción, se recuperan los valores de los registros y se restituyen estos registros en procesador con el contexto que se salvó anteriormente.

3.4. La última acción consiste en restituir de la pila los valores de la PSW y del PC.

4- Como resultado, la siguiente instrucción que se va a ejecutar corresponderá al proceso previamente interrumpido. No siempre tiene porque restaurarse el mismo proceso que se interrumpe.

5.3. Interrupciones múltiples

Supóngase que se producen múltiples operaciones, por ejemplo, es posible que se produzca una interrupción de comunicación mientras se está procesando una interrupción de la impresora.

Das alternativas a la hora de tratar con múltiples interrupciones:

- 1) Inhabilitar las interrupciones mientras que se está produciendo una. Es decir, que el procesador ignorará cualquier nueva señal de petición de interrupciones. No tiene en cuenta el grado de urgencia de las interrupciones.
- 2) Una segunda estrategia es definir prioridades para las interrupciones y permitir que una de más prioridad cause que se interrumpe la ejecución de un manejador de una interrupción de menor prioridad.

6. Arquitectura de un sistema informático desde el punto de vista del procesador.

6.1. Sistemas de un solo procesador.

En un sistema de un único procesador, hay una CPU principal capaz de ejecutar un conjunto de instrucciones de propósito general, incluyendo instrucciones de los programas de usuario. Además casi todos los sistemas disponen de otros procesadores ~~esp~~ de propósito especial, los cuales pueden venir en forma de procesadores específicos de un dispositivo que quitan carga al procesador principal realizando tareas concretas por él.

Todos estos procesadores de propósito especial ejecutan un conjunto limitado de instrucciones y no ejecutan programas de usuario. En ocasiones, el SO los gestiona, les envía información sobre su siguiente salida tarea y monitoriza su estado. Este método libera a la CPU principal del trabajo adicional de planificar las tareas de disco.

La presencia de micros de propósito especial resulta bastante común y ~~na~~ convierte a un sistema de un solo procesador en un sistema multiprocesador.

6.2. Sistemas multiprocesador.

1. Mayor rendimiento

Son sistemas con más de un núcleo habitualmente en un solo procesador, o más de un procesador. Ventajas con respecto a los sistemas de un procesador.

1. Mayor rendimiento; la mejora de velocidad con N procesadores es K , sino que es menor que N . Cuando

múltiples procesadores cooperan en una tarea, cierta carga de trabajo se emplea en conseguir que todas las partes funcionen correctamente. Esto junto a la competición por los recursos compartidos, reducen la ganancia esperada por añadir procesadores adicionales.

2. Economía de escala. Los sistemas multiprocesador pueden resultar más baratos ya que pueden compartir periféricos, almacenamiento masivo y fuentes de alimentación. Es más barato almacenar datos en un disco y que todos los procesadores los compartan.

Estos sistemas son simétricos o asimétricos.

■ Simétrico: cada procesador se asigna a una tarea específica. Un procesador maestro controla el sistema y el resto de los procesadores esperan que éste les de instrucciones. Relación maestro-esclavo.

La ventaja más importante de esta configuración es su simplicidad. Presenta 3 desventajas serias:

- Su confiabilidad no es mayor si falla el maestro galla todo el sistema.

- Puede llevar a un uso deficiente.

- Incrementa el número de interrupciones, porque todos los procesadores esclavos deben interrumpir al maestro cada vez que requieren intervención del OS.

■ Los sistemas más comunes usan simétrico (SMP) cada procesador realiza todas las tareas correspondientes al OS. Es un sistema de computación aislado. Características:

- Múltiples procesadores o núcleos contenidos dentro del mismo procesador.

- Éstos comparten las utilidades de MP y E/S interconectadas por un bus.

- Todos pueden realizar las mismas funciones

- Transparente al usuario. El OS se encarga de planificar los hilos o procesos en procesadores individuales y de la sincronización entre los procesadores.

Ventaja fundamental además de Mayor rendimiento y economía de escala:

- Disponibilidad: El fallo de un procesador no para la máquina.

- Más confiable.
- Puede usar los recursos de memoria efectiva.

Hace al sistema operativo más complejo.

6.3. Sistemas en cluster.

Usan múltiples CPU para llevar a cabo el trabajo, y cada CPU puede tener varios núcleos. Están formados por 2 o más sistemas individuales acoplados. Las computadoras en cluster comparten el almacenamiento y se conectan a través de una red de área local. Para el usuario es visto como un único ordenador.

Los clusters son empleados para mejorar el rendimiento y la disponibilidad por encima de la que es provista por un solo computador.

Todos pueden tener la misma configuración de hardware y SO, o tener distintos, lo que hace más fácil su construcción. Para que un cluster funcione como tal es necesario proveer un sistema de software de manejo del cluster que se encargue de interactuar con el software usuario y los procesos.

Diferentes connotaciones para distintos grupos de personas:

• Clusters de alto rendimiento: En ellos se ejecutan tareas que requieren gran capacidad computacional, grandes cantidades de memoria o ambas a la vez. Puede comprometer los recursos del cluster largos períodos de tiempo.

• Clusters de alta disponibilidad: Su objetivo de diseño es proveer disponibilidad y confiabilidad. La confiabilidad se provee con software detector de fallos, mientras en hardware se evita tener un único fallo.

• Clusters de alta eficiencia: Objetivo: Ejecutar la mayor cantidad de tareas en el menor tiempo posible. Existe independencia de datos.

7. Multiprogramación

La multiprogramación incrementa el uso de la CPU de forma que no quede ociosa de trabajos.

La idea es la siguiente:

El OS mantiene en memoria principal varios trabajos y empieza a ejecutar uno de ellos.

Si el sistema no fuera multiprogramado, la CPU se quedaría inactiva hasta que una operación de E/S concluyese o el evento esperado se produjese y el trabajo en ejecución pudiera continuar. En un sistema multiprogramado, el SO simplemente cambia de trabajo y ejecuta otro mientras que se realiza esa operación de E/S. Siempre que la CPU tiene que esperar, se conmuta a otro trabajo que esté listo para ejecución. A qué sistema se cambia es decisión del planificador que el sistema utilice. Mientras haya un trabajo que necesite ejecutarse, además del actual, la CPU nunca estará inactiva y ociosa.

La multitarea solo se puede producir de manera real en un sistema con 2 procesadores.

8. Multiprocesamiento

Es el uso o ejecución de múltiples programas concurrentes en un sistema en lugar de un único proceso en un instante determinado.

En un sistema con multiprocesamiento, también hay multiprogramación.

9. Modo dual (Usuario y Kernel)

Un sistema operativo está esperando siempre a que ocurran sucesos que se indican mediante la ocurrencia de una interrupción o una excepción (interrupción generada por software debida a un error). Para cada tipo de interrupción, diferentes segmentos de código del SO determinan qué acción hay que llevar a cabo.

El código del propio S.O. se encuentra codificado en lenguaje máquina. Necesitamos asegurar que un error que se produzca en un programa de usuario sólo genere problemas en el programa que se estuviera ejecutando, y no en el control del hardware o en otros usuarios y programas.

Para que el sistema informático funcione con ~~seguridad~~ y adecuadamente, hay que impedir que los programas de usuario puedan realizar libremente ciertas operaciones que puedan llevar a conflictos y a un mal uso del sistema como opera-

ciones o invocaciones que tengan que ver con:

- Acceso a zonas de memoria restringidas.
- Acceso directo a los dispositivos de E/S.
- Utilización de la CPU por un proceso todo el tiempo que se requiera.

Modos de ejecución del procesador:

- Modo núcleo, asociado al procesamiento de rutinas e instrucciones del S.O.

- Modo usuario: rutinas e instrucciones de los programas de usuario

El núcleo es la parte del S.O. que engloba las funciones y servicios más importantes del sistema y que deben protegerse. Es una parte relativamente pequeña del sistema, pero la más usada, por ello suele residir en memoria principal de forma continua.

Para saber en qué modo está ejecutando el procesador, existe un bit en la palabra de estado, que indica el modo. Cuando ocurre una interrupción, una excepción o una llamada al sistema, el modo de ejecución se cambia a modo núcleo y, tras la finalización del servicio, el modo se fija de nuevo a modo usuario.

Una vez que se dispone de la protección hardware, el hardware detecta los errores de violación de los modos y el S.O. se encarga de tratarlos. Si un programa de usuario falla, el hardware envía una excepción al S.O. La excepción transfiere el control al S.O. a través del vector de interrupción.

La falta de un modo dual soportado por hardware puede dar lugar a serios defectos en el S.O.

10. Llamadas al sistema

Proporcionan un interfaz entre un programa y el S.O. para poder invocar los servicios que éste ofrece, el usuario no puede acceder o no tiene privilegios directos sobre los recursos que gestiona el S.O.

Las llamadas al sistema pueden agruparse en cinco categorías principales: control de procesos, manipulación de archivos, manipulación de dispositivos, mantenimiento de información y comunicaciones. Determinadas tareas de bajo nivel, pueden necesitar escribirse con instrucciones de lenguaje ensamblador. La ejecución de las llamadas se hace en modo núcleo.

Una vez que se han obtenido los nombres de los 2 archivos, el programa debe abrir el archivo de entrada y crear el de salida.

Normalmente, los desarrolladores de aplicaciones diseñan sus programas utilizando una API, la cual especifica un conjunto de funciones que el programador de aplicaciones puede usar. Tres de las API más usuales disponibles para programadores de aplicaciones son la API Win32 para Windows, la API POSIX para sistemas basados en POSIX y la API Java para diseñar programas que se ejecuten sobre una máquina virtual de Java. En Linux encontraremos muchas de las llamadas al sistema en la librería estándar de C, la cual nos proporciona llamadas que directamente podemos usar desde un programa o desde un terminal.

Las funciones que conforman una API invocan a las llamadas al sistema por cuenta del programador de la aplicación. Una ventaja de programar usando una API cuando quiere poder compilar y ejecutar su programa está relacionada con la portabilidad: un programador de aplicaciones diseña un programa usando una API cuando quiere poder compilar y ejecutar su programa en cualquier sistema que soporte la misma API. Existe una fuerte correlación entre invocar una función de la API y su llamada al sistema asociada disponible en el Kernel.

Quien realiza una llamada solo necesita ajustarse a lo que la API especifica y entender lo que hará el S.O. como resultado de la ejecución de dicha llamada al sistema.

¿Qué ocurre cuando se produce una invocación?

1. Cuando se produce una llamada al sistema los parámetros asociados a la misma se cargarán en la pila.
2. Posteriormente, se ejecuta una instrucción especial del S.O. que se llama trap, y que conlleva un cambio de modo. Al ejecutarse trap se ejecuta un programa que examina la rutina invocada y sus parámetros, y busca en un vector de rutinas si esta existe.

La ejecución de trap lleva consigo que sea necesario salvar la información del contexto del proceso que estaba en ejecución en ese momento.

3. Tras identificar que existe una rutina para la llamada realizada y que los parámetros son correctos, se procede a ejecutarla.

4. Posteriormente el programa trap solicita un código de estado almacenado en un registro que señala si la llamada tuvo éxito o no y ejecuta una instrucción de tipo RETURN FROM TRAP para regresar el control a la rutina de biblioteca. Al pasar de modo núcleo a modo usuario se restaura el contexto del proceso que estaba en ejecución cuando se hizo la llamada.

5. Cuando finalizan esas acciones, la rutina de biblioteca a nivel de usuario descarga la pila y comprueba el resultado de la ejecución de la petición al sistema, se lo devuelve al usuario y se continua por la siguiente línea de código del programa.

11. Interfaz de Usuario del S.O.

Dos métodos para que los usuarios interactúen con el S.O.
Uno consiste en proporcionar una interfaz de línea de comandos que permita al usuario introducir directamente comandos que el S.O. pueda ejecutar. El segundo permite que el usuario interactúe con el S.O. a través de una interfaz gráfica o GUI

11.1. Interpretador de comandos

En los sistemas UNIX y Linux, hay disponibles varias shells diferentes entre las que un usuario puede elegir, incluyendo la shell Bourne, la shell C, la shell Bourne-Again, la shell Korn, etc. La mayoría proporcionan funcionalidades similares, existiendo algunas diferencias menores.

La función principal del interpretador de comandos es obtener y ejecutar el siguiente comando especificado por el usuario. Muchos de los comandos se utilizan para manipular archivos. Estos comandos pueden implementarse de 2 formas generales:

- El propio interpretador de comandos contiene el código que el comando tiene que ejecutar.
- Método alternativo: implementa la mayoría de los comandos a través de una serie de programas del sistema. No "entiende" el comando, sino que lo usa para ~~identificar y cargar~~ ejecutar el archivo que hay que cargar en

memoria y ejecutar.

El comando UNIX rm file.txt para borrar un archivo buscaba un archivo llamado rm, lo cargaría en memoria y lo ejecutaría, pasándole el parámetro file.txt

11.2. Interfaces gráficas de usuario

Una segunda estrategia para interactuar con el S.O. es a través de una interfaz gráfica de usuario suficientemente amigable. Una GUI permite a los usuarios emplear un sistema de ventanas y menús controlables mediante el ratón. Dependiendo de la ubicación del puntero, pulsar el botón del ratón puede invocar un programa, seleccionar un archivo o directorio o desplegar un menú que contenga comandos ejecutables.

La primera GUI se incorporó en la computadora Xerox Alto, en 1973. Las interfaces gráficas solo se popularizaron con la introducción de las computadoras Apple Macintosh en los 80

En los sistemas UNIX han predominado las interfaces de línea de comandos, aunque hay disponibles varias interfaces GUI, incluyendo el entorno de escritorio CDE y los sistemas X-Windows como Solaris o el sistema AIX de IBM.

Los entornos KDE y GNOME, se ejecutan sobre Linux y otros varios sistemas UNIX, y su código fuente es de dominio público. El proyecto GNOME ha derivado en otras interfaces gráficas como Mate y Cinnamon.

Los diversos cambios experimentados por Macintosh proporcionan un interesante caso de estudio. En Mac OS era obligatorio que los usuarios interactuaran con el S.O. a través de la GUI. Sin embargo desde el lanzamiento de Mac OS X el S.O. proporciona tanto GUI como interfaz de línea de comandos.

El diseño de una interfaz de usuario útil y amigable no es una función directa del S.O.

12. Arranque del sistema

Para que una computadora comience a ~~est~~ funcionar es necesario que tenga un programa de inicio que ejecutar, el cuál normalmente se almacena en una ROM o una EEPROM y se conoce como firmware o Bios. Una ROM resulta adecuada ya que no puede ser infectada. Si se posee ROM, para cambiar el sistema de arranque es necesario cambiar el chip en si, en cambio con una EEPROM, puede actualizarse el firmware. El propósito del Bios es identificar y diagnosticar los dispositivos del sistema de forma que el entorno quede preparado para cargar el kernel en la RAM.

Si se pasan las pruebas de diagnóstico satisfactoriamente, el programa puede continuar con la secuencia de arranque. En este segundo paso, el bios hace que se comience a ejecutar un programa llamado gestor de arranque o cargador de arranque de segunda etapa y suele encontrarse en el primer sector del dispositivo de arranque. La Bios busca en ese sector y lo carga en memoria principal, de forma que se comienza a ejecutar el código que hay en dicho bloque de arranque.

El gestor de arranque se trata de un código que cabe en un solo bloque de disco y que únicamente conoce la dirección del disco y la longitud del resto del programa de arranque.

Cuando el gestor de arranque empieza a ejecutarse, comienza a explorar el sistema de archivos para localizar el Kernel del S.O. y cargarlo en memoria principal.

Como ejemplos de cargadores de arranque tenemos Lilo y Grub en UNIX o NT loader y Windows Boot Manager en Windows.

Un problema general con el firmware es que ejecutar código en él es más lento que en RAM además de ser relativamente caro.

13. Implementación

El primer sistema que no fue escrito en ensamblador fue probablemente el MCP para las computadoras Burroughs; MCP fue escrito en una variante de ALGOL. Los S.O. Linux y Windows XP están escritos en su mayor parte en C, aunque hay algunas pequeñas secciones de código ensamblador.

Las ventajas al usar un lenguaje de alto nivel para implementar S.O. son: el código puede escribirse más rápido, es más compacto y más fácil de entender y depurar, además de ser más fácil de portar.

Las únicas desventajas se reducen a los requisitos de velocidad y de espacio de almacenamiento. Si lo que queremos es desarrollar programas grandes, un compilador moderno puede realizar análisis complejos y aplicar optimizaciones avanzadas que produzcan un código excelente.

Las principales mejoras de rendimiento en los S.O. son el resultado de utilizar mejores estructuras de datos y mejores algoritmos, más que de usar código optimizado en ensamblador. Después de escribir el sistema y de que éste esté funcionando correctamente, pueden identificarse las rutinas que constituyan un cuello de botella y reemplazarse por equivalentes en ensamblador.

Para identificar los cuellos de botella y las ineficiencias, debemos poder monitorizar el rendimiento del sistema. Debe añadirse código para calcular y visualizar medidas del comportamiento del sistema. Todos los registros interesantes se registran, junto con la hora y los parámetros importantes, y se escriben en un archivo. Después, un programa de análisis puede procesar el archivo de registro para determinar el rendimiento del sistema e identificar los cuellos de botella y las ineficiencias. Estas mismas trazas pueden proporcionarse como entrada para una simulación que trate de verificar si resulta adecuado introducir determinadas mejoras.