GABRIEL ALEJANDRO MORAN VALDES
ID: UB3844SIT8938

USAGE AND ADMINISTRATION OF OPERATING SYSTEMS

INFORMATION TECHNOLOGY SCIENCE

MEXICO,DF

ATLANTIC INTERNATIONAL UNIVERSITY

JAN 2008

BIBLIOGRAFIA

Deitel, Harvey M., 1945-, Operating systems / Harvey M. Deitel, Paul J. Deitel, David R. Choffnes., 3rd ed., Upper Saddle River, NJ : Pearson/Prentice Hall

www.google.com

Introducción a los sistemas operativos.

¿Qué es un sistema operativo?

Es un programa que controla la ejecución de los programas de aplicación. Actúa como interfaz entre el usuario y el hardware. Proporciona al usuario un entorno cómodo y eficiente para ejecutar sus programas.

Objetivos:

Facilitar el uso del sistema informático.
Uso eficiente del hardware del ordenador.

El sistema operativo como máquina virtual.

Sería muy difícil para el usuario y su aplicación dominar todos los elementos a nivel de hardware y direcciones máquina.

Un sistema operativo transforma un cierto hardware en una máquina más fácil de usar.

El sistema operativo como administrador (manejador) de recursos.

Un sistema informático tiene recursos como la memoria central, CPU, discos, impresoras y tiene usuarios que quieren usar esos recursos. El sistema operativo debe optimizar el uso de sus recursos entre los distintos usuarios para conseguir un máximo rendimiento y debe lograr que se compartan los recursos del sistema y los datos entre varios usuarios que trabajen simultáneamente.

Evolución histórica de los sistemas operativos.

Los primeros sistemas.

No existía un sistema operativo propiamente dicho. Se utilizaba un esquema de reserva por solicitud. El modo de operación era proceso en serie. Se operaba directamente con el ordenador. Era un modo de trabajo interactivo. El programa tenía que ser cargado directamente en memoria.

Aparecen los primeros ensambladores, enlazadores, cargadores de programas y compiladores de Fortran y Cóbol. Aparecen las bibliotecas de funciones comunes y controladores de dispositivos.

Sistemas por lote.

Se contratan operadores especiales que interactuaban directamente con el ordenador, no el programador. Se organizan los trabajos por lotes: Datos, código ejecutable, instrucciones de control. Por ejemplo: varios trabajos todos en el mismo lenguaje.

Esto da lugar al primer sistema operativo: secuencionamiento automático de trabajos. Un programa monitor residente se encarga de cargar en memoria el programa y ejecutarlo. Luego se vuelve a llamar al monitor que carga otro programa y lo ejecuta y así sucesivamente. Estas órdenes se le daban a través de tarjetas de órdenes.

Partes del monitor residente:

- Intérprete de tarjeta de control.

Cargador.

Manejadores de dispositivos.

Tratamiento de interrupciones.

Mientras se realizan operaciones de E/S, la CPU está aprovechando ese tiempo:

Operaciones fuera de línea: la E/S se realiza en otro aparato, a través de cintas magnéticas.

Procesamiento satélite: un ordenador satélite sólo se dedica a operaciones de E/S. Esto da lugar a los dispositivos lógicos de E/S.

Aparecen el buffering y el spooling para que todas las operaciones se realicen en la misma máquina.

Buffering: Método de solapar operaciones de E/S con operaciones de CPU del mismo proceso. Para sincronizar el trabajo entre la CPU y los dispositivos de E/S se emplean las interrupciones. En un sistema guiado por interrupciones, la CPU es interrumpida por cada carácter que realiza una operación de E/S. Para liberar a la CPU de todo el trabajo de E/S se data al sistema de DMA. En un sistema con DMA, la CPU es interrumpida por cada bloque de datos que realiza una operación de E/S.

Spooling: Aparece cuando aparecen los discos magnéticos. Método para simultanear las operaciones de E/S de otros procesos con las de computación del trabajo actual. Las operaciones de E/S de los procesos anteriores y posteriores, son realizadas por el spool, concurrentemente con el proceso actual. Utiliza el disco como un gran buffer de E/S. La CPU no va a tener que esperar al dispositivo de entrada porque ya tiene la información disponible en un dispositivo rápido. El dispositivo de salida tomará la información del buffer a su máxima velocidad. Los sistemas que utilizan el spool crean las colas de trabajo, teniendo que seleccionar el que va a realizar primero.

Sistemas por lote multiprogramados.

Características:

Existen varios procesos simultáneamente residentes en memoria.

Los tiempos de E/S de un proceso se solapan con tiempos de CPU de otros.

Los procesos se alternan el uso del procesador.

Existe una compartición de recursos.

Grado de multiprogramación: es el número de procesos que están activamente ejecutándose, es decir, residentes en memoria en un instante dado.

Los sistemas multiprogramados son más complejos por que requieren llevar el control sobre:

Planificación CPU.

Planificación de los dispositivos.

Control de la concurrencia.

Control de la memoria.

Protección.

Sistemas de tiempo compartido.

Son sistemas multiprogramados, multiusuarios e interactivos. Los usuarios pueden interactuar con su programa mientras se ejecuta y obtener una respuesta rápida. Los recursos del sistema son compartidos por todos los usuarios, que tienen la “ilusión” de tener todos los recursos a su disposición.

Utilizan planificación de reparto de tiempo. Proporcionan un buen tiempo de respuesta. Aparecen los teclados y las pantallas.

Sistemas para ordenadores personales.

Son sistemas monoprogramados donde todos los recursos los utiliza un solo usuario. Suponen un abaratamiento del hardware.

Sistemas multiprocesadores.

Son sistemas que tienen más de un procesador en donde se están ejecutando más de un proceso realmente de modo simultaneo, cada uno en un procesador diferente. Con estos sistemas se consigue un aumento en la fiabilidad y en el rendimiento.

Tipos:

Fuertemente acoplados: los procesadores comparten memoria y reloj. La comunicación se realiza por medio de la memoria compartida. Pueden ser:

Multiproceso simétrico: cada CPU posee su copia del sistema operativo.

Multiproceso asimétrico: hay una CPU maestra que controla a las demás.

Débilmente acoplados: (sistemas distribuidos) Los procesadores no comparten memoria ni reloj. Cada procesador tiene su memoria local. Las comunicaciones entre procesos se realizan por medio de líneas de comunicación. Los procesadores pueden ser desde pequeños microprocesadores hasta grandes sistemas. Sistema distribuido es aquél en el que la inteligencia del sistema informático está repartida por distintos procesadores. Ventajas: Compartición de recursos, aumentar la velocidad de computación, fiabilidad y comunicación.

Sistemas de tiempo real.

Su prioridad es el tiempo de respuesta corto. Deben responder inmediatamente a los eventos externos. El usuario y la utilización de recursos son una preocupación secundaria. Cada proceso tiene asignado una prioridad (planificación apropiativa por prioridades). Atienden dos tipos de tareas:

Tareas duras: necesitan una atención inmediata. Posee un plazo fijo breve.

Tareas blandas: necesitan una respuesta rápida. Posee un plazo algo más largo que las duras, pero sigue siendo breve.

Una tarea blanda puede convertirse en dura si pasa su plazo.

Dos tipos de sistemas en tiempo real:

S.T.R. duros: aceptan tareas duras.

S.T.R. blandos: aceptan tareas duras y blandas y además es flexible. Debe saber cuando pasar una tarea blanda a dura.

Estructura y funciones de los sistemas operativos.

Servicios de un sistema operativo.

Servicios ligados a facilitar el uso del sistema informático:

Creación de programas.

Ejecución de programas.

Operaciones de E/S.

Gestión de información.

Comunicaciones entre procesos.

Detección de errores.

Servicios ligados a un uso eficiente del hardware:

Asignación de recursos.

Contabilidad.

Protección y seguridad.

Métodos para solicitar servicios.

Llamadas al sistema.

Las llamadas al sistema proveen una interfaz entre los procesos y los servicios del sistema operativo. Mediante las llamadas al sistema, el usuario solicita los servicios que desea del sistema operativo.

El tratamiento de una llamada al sistema es igual al de una interrupción. El paso de parámetros se puede realizar:

Directamente en la llamada.

Cargando en un registro, la dirección de la tabla donde están todos los parámetros.

Mensajes.

Estructura de un sistema operativo.

Un sistema operativo es una compleja y enorme colección de rutinas de software, por lo que se suelen dividir en pequeños módulos. Los principales módulos de un sistema operativo son:

El administrador de procesos.

*

El administrador o gestor de memoria principal.

*
El administrador del almacenamiento secundario.
*
El administrador del sistema de E/S.
*
El administrador de ficheros.
*
El sistema de protección.
*
El sistema de comunicaciones o trabajo en red.
*
El intérprete de comandos.

Existen diversas estructuras de sistemas operativos:

Estructura simple o monolítica.

Es un macroprograma con miles de líneas de código. No existe una estructura interna definida entre los módulos del sistema operativo. Utilizan técnicas de programación modular. Son difíciles de mantener y actualizar. Todos los componentes están muy relacionados. No existe ningún tipo de ocultación de información. Se comporta como un único proceso. Tienen buen rendimiento hardware.

Ejemplos:

Estructura modular.

Se divide el núcleo en módulos con sus funciones bien definidas. Dependiendo de la metodología tenemos:

* Núcleo estructurado: usando metodología estructurada.
* Núcleo orientado a objetos: usando metodología orientada a objetos.

Estructura por capas o estratos.

Consiste en organizar el sistema operativo como una jerarquía de estratos o capas. Cada capa se implementa usando sólo aquellas operaciones que le proporciona el nivel inmediatamente inferior. Se emplean técnicas modulares y de diseño top-down. La depuración y verificación del sistema es mucho más sencilla. La mayor dificultad está en definir las capas. Las operaciones al tener que pasar por varias capas, son menos eficientes.

Máquinas virtuales.

El concepto de máquina virtual es crear sobre una máquina real varias máquinas virtuales en las que puedan correr sistemas operativos distintos simultáneamente.

Un ejemplo es el sistema operativo VM que separa los aspectos de multiprogramación y máquina extendida mediante monitor de máquina virtual, que proporciona multiprogramación, y por otro lado están los núcleos de los sistemas operativos que proporcionan la característica de máquina extendida.

Estructura de micronúcleo.

Núcleo mínimo que hace las funciones mínimas que debe hacer un núcleo. El resto de funciones se crean como procesos del sistema. Estos procesos del sistema se ejecutan en modo usuario. Cuando un usuario solicita un servicio al núcleo este le pasa el control al proceso del sistema correspondiente. Cuando termina, el núcleo le pasa la respuesta al usuario. El núcleo realiza un paso de mensajes.

Modelo cliente/servidor.

Consiste en diseñar un núcleo del sistema operativo mínimo (micronúcleo), dejando la mayoría de las funciones a procesos de usuarios. El proceso que solicita un servicio es el proceso cliente. El proceso que ofrece el servicio es el proceso servidor.

Diseño, implementación y configuración de un sistema operativo.

*

Diseño.

Se tendrán en cuenta las técnicas actuales de ingeniería del software, con dos objetivos principales: Objetivos de usuario y objetivos del sistema.

Lo que nos va a condicionar es el hardware y el tipo de sistema operativo que vamos a tener.

Los requisitos se dividen en:

*

Del usuario: fácil de usar, fiable, seguro, rápido.

*

Del sistema: fácil de diseñar, implementar, mantener, fiable, libre de errores, transportable.

*

Implementación.

Se puede implementar:

*

En lenguaje de bajo nivel: es dependiente de la máquina y de difícil migración.

*

En lenguaje de alto nivel: el código es más fácil de entender, de depurar, de implementar, de transportar. Inconvenientes: la velocidad disminuye y una mayor capacidad de almacenamiento.

*

Configuración del sistema operativo.

El sistema debe de generarse o adaptarse a cada instalación. Las características que podemos configurar son:

*

Memoria disponible.

*

Número máximo de usuarios posibles.

*

Dispositivos.

*

Número máximo de procesos.
*

Tipo de CPU, si se ejecuta en varias diferentes.

El núcleo que se genera puede ser:

*

Estático: configurado para unas características. Si cambia el hardware hay que volver a compilar el núcleo.
*

Dinámico: Tiene partes que se pueden cargar y descargar de memoria.

Mediante un programa de arranque, `bootstrap program', que sabe donde está el núcleo, lo carga y lo ejecuta.

El sistema operativo Linux.

Es un sistema operativo tipo UNIX. Tiene núcleo monolítico. El núcleo tiene los siguientes componentes:

*

Administrador de procesos.
*

Administrador de memoria.
*

Administrador de dispositivos de red.
*

Administrador de sistemas de ficheros.

*

Administrador de dispositivos de E/S.

Para acceder los usuarios a los componentes del núcleo lo hacen mediante llamadas al sistema.

Linux reconoce diferentes sistemas de ficheros, gracias al administrados del sistema de ficheros virtual. Nos proporciona un conjunto de operaciones genérico. Este administrador posee una serie de administradores específicos para cada sistema de ficheros.

Linux genera núcleos dinámicos mediante los módulos.

Descripción y control de procesos.

¿Qué es un proceso?

La definición más aceptada: Programa que se está ejecutando. Un proceso es la unidad de trabajo de un sistema. Mientras que un programa es una entidad estática, un proceso es una entidad dinámica.

Vida de un proceso.

Un proceso está formado por:

*

Un programa ejecutable.

*

Unos datos.

*

Un contexto de ejecución:

*

Registros de la CPU.

*

Otra información.

El sistema operativo ve al proceso como una unidad que se va ejecutando y evolucionando entre distintos estados. El estado global del sistema está constituido por el estado en el que se encuentran todos los procesos y recursos del sistema.

Creación y terminación de procesos.

La creación de un proceso consta de dos etapas:

*

Creación de estructuras de control.

*

Asignación del espacio de memoria.

Generación de procesos: Cuando un proceso (padre) crea a otro proceso. (hijo)

La terminación de un proceso consta de dos etapas:

*

Liberación del espacio de memoria.

*

Destrucción de la estructura.

Causas para la terminación de un proceso.

*

Por que finaliza su ejecución

*

Debido a errores:

*

Por que se lo pide el proceso padre.

*

Intento de acceso a una zona de memoria donde no tiene permiso.

*

Intento de acceder a un recurso que no tiene derecho o de forma indebida.

*

Intento de realizar una operación aritmética no permitida.

Estados de un proceso y transiciones entre estados.

Modelo de dos estados.

Es el más simple. Posee dos estados: Ejecución y No Ejecución.

Modelo de cinco estados.

*

Estado de ejecución: el proceso tiene el dominio de la CPU en ese momento y se está ejecutando.

*

Estado de listo: el proceso está preparado y en disposición de usar la CPU, si hubiese una libre.

*

Estado de bloqueado: el proceso está a la espera de que ocurra algún evento.

*

Estado de nuevo: el proceso acaba de ser creado, pero no se le ha asignado memoria todavía. Todavía no ha sido admitido por el sistema.

*

Estado de terminado: el proceso ha terminado, ya se ha liberado su memoria, pero no se han destruido sus estructuras de datos.

Un proceso cambia de estados a lo largo de su vida. Transiciones:

*

Despacho: Listo Ejecución

*

Tiempo excedido: Ejecución Listo

*

Bloqueo: Ejecución Bloqueado

*

Despertar: Bloqueado Listo

La única transición iniciada por el propio proceso es la de bloqueo. Este modelo puede ser completado con un estado más: Suspendido.
#

Descripción de un proceso.
#

Estructuras de control del sistema operativo.

El sistema operativo necesita una serie de estructuras de datos que le permitan controlar el estado de los procesos y de los recursos. Ejemplo:

* Tablas de memoria.
* Tablas de E/S.
* Tablas de ficheros.
* Tablas de procesos.

#

Estructura de control de un proceso.

Físicamente un proceso se manifiesta por tener en memoria:

* Un código.
* Unos datos.

*

Una pila.

*

Unos atributos.

A estos 4 elementos se les denomina imagen de un proceso. Y formalmente se les llama:

*

Programa de usuario: Programa que va a ser ejecutado.

*

Datos de usuario: Parte modificable del espacio de usuario. Incluye:

*

Datos del programa.

*

Pila del usuario.

*

Parte del programa que puede ser modificada

*

Pila del sistema: utilizada para las llamadas a procesos y llamadas al sistema.

*

Bloques de control de procesos: formados por toda la información que necesita el sistema operativo para controlar a los procesos.

Bloque de control de procesos.

La estructura de datos que contiene la información a cerca de un proceso se denomina bloque de control de procesos (BCP)

La manifestación de un proceso en un sistema operativo es un BCP. Un BCP es la representación de un proceso durante su ejecución. Un BCP se crea cuando se crea su proceso asociado. El conjunto de BCPs representa el estado del sistema. Los BCP son accedidos y mantenidos por rutinas del sistema operativo.

Elementos del BCP:
#

Identificación de procesos.

* Identificación única del proceso.
* Identificación del proceso padre.
* Identificación del usuario.

#

Información del estado del proceso.

* Registros de control y estado: PC, PSW, códigos de condición.
* Registros visibles al usuario.
*

Punteros de pila.

#

Información de control de procesos.

*

Información sobre planificación y estado.

*

Información útil en la planificación de la CPU: prioridad del proceso, punteros a colas, cantidad de tiempo en espera, en ejecución, etc.

*

Eventos pendientes.

*

Gestión de memoria.

*

Límite superior e inferior de la memoria asignada.

*

Lista de los bloques asignados.

*

Apuntador a la tabla de segmento o de página.

*

Información de contabilidad: Tiempo de CPU y tiempo real.

*

Información del estado de los dispositivos y operaciones de E/S: Lista de dispositivos asignados, ficheros abiertos.

*

Apuntadores para asignar recursos.

*

Comunicación entre procesos: Tubos, señales.

El BCP permite al sistema operativo localizar toda la información clave sobre procesos. El conjunto de BCP forman las tablas de procesos. El id del proceso se usa como índice en las tablas de los procesos.

#

Gestión de procesos.

#

Creación de un proceso.

Cuando se crea un proceso, el sistema operativo realiza los siguientes pasos:

#

Asigna un identificador único al proceso. Se crea el BCP, pero sólo se rellena el campo del identificador.

#

Asigna memoria al proceso.

#

Inicializa el BCP.

#

Insertar el proceso en la lista de procesos que corresponda. Por ejemplo en la lista de procesos listos.

#

Otras operaciones, como actualizar todas las estructuras de control que mantenga el sistema operativo.

Un proceso puede ser creado por:

* Un usuario que lo solicita.
* El sistema operativo.
* Otro proceso.

#

Cambio de proceso.

Mecanismos para interrumpir la ejecución de un proceso:

Mecanismo

Causa

Uso

Interrupción

Externa a la ejecución de la instrucción actual.

Reacción a un evento externo asíncrono.

Trampa.

Ecepciones.

Asociado con la ejecución de la instrucción actual.

Manejo de errores o de condiciones excepcionales.

Petición de un servicio.

(Llamada al sistema)

Petición explícita.

Llamada a una función del

sistema operativo

*

Tratamiento de una interrupción. (Cambio de contexto)

No siempre se cambia el estado de un proceso. Cuando se produce una interrupción, el procesador:

*

Salva el contenido de los registros del procesador, del CP y de los punteros de pila. (Cambio de contexto)

*

Establece el contador de programa.

Debe guardar toda aquella información que puede ser alterada por el tratamiento de la interrupción, en el BCP.

* Cambio de proceso. (Por interrupción de reloj, fallo de memoria, servicio de E/S, etc.)

Pasos:

* Cambio de contexto.
* Actualizar el BCP del proceso que está actualmente en ejecución. Para cambiar el estado.
* Mover el BCP a la cola adecuada.
* Seleccionar otro proceso para ejecutar.
* Actualizar el BCP del proceso seleccionado. Para cambiar el estado.
* Actualizar las estructuras de datos para el manejo de la memoria.
* Restaurar el contexto del procesador.

Se producen dos cambios de contexto.
#

Terminación de un proceso.

Para destruir un proceso:

* Se devuelven sus recursos al sistema.
* Se borra de todas las listas y tablas del sistema.
* Se borra su BCP.

Un proceso puede ser destruido por:

* Completar la ejecución del proceso.
* Exceder el límite de tiempo asignado al proceso.
* No haber la memoria disponible que el proceso solicita.
* Violación de protección de memoria.
* Error de protección al intentar acceder a un recurso no autorizado.
* Error aritmético.
* Instrucción inválida.

Fallo de E/S.

*

Intervención del usuario o del sistema operativo.

#

Otras operaciones.

*

Cambiar la prioridad de un proceso.

*

Bloquear un proceso.

*

Despertar un proceso.

*

Suspender/reanudar un proceso.

Concepto: Suspender un proceso consiste en detener un proceso, esté en el estado en que esté. El proceso puede salir o no de memoria.

Características de un proceso suspendido:

*

El proceso que está suspendido, no está inmediatamente disponible para su ejecución.

*

La condición de bloqueado es independiente de la de suspendido.

*

El proceso puede ser situado en estado de suspendido por él mismo, el proceso padre, el sistema operativo o el usuario con el fin de evitar que pueda ser ejecutado.

*

Un proceso sólo puede salir del estado de suspendido si se le aplica una orden externa de reanudar. El propio proceso no puede reanudarse.

Justificación. Motivos por los que un proceso puede ser suspendido:

*

Por el sistema operativo:

*

Ante un fallo del proceso.

*

Carga alta del sistema. (No hay memoria suficiente)

*

Procesos que sólo se ejecutan cada ciertos intervalos de tiempo. (Demonios,...)

*

Por decisión del propietario del proceso.

*

Por el proceso padre.

Cuando un proceso es suspendido, puede que esté a la espera de un evento (bloqueado). Si ocurre el evento, el proceso ya no está bloqueado, simplemente está suspendido.

Debemos de subdividir el estado de suspendido en dos nuevos estados:

*

Suspendido bloqueado.

*

Suspendido listo.

#

Hilos de ejecución. (threads)

Partes de un proceso:

*

Unidad propietaria de recursos.

*

Unidad de ejecución o despacho.

En los procesos tradicionales existe un solo espacio de direcciones y un solo hilo de ejecución. Procesos pesados.

Hay situaciones en que es deseable tener más de un hilo de ejecución, que compartan un único espacio de direcciones, pero que se puedan ejecutar de modo más o menos simultaneo. A estos flujos de control se les denomina hebras o hileras de ejecución. Procesos ligeros, es la unidad propietaria de recursos y el hilo es la unidad despachable.

Las hebras son como mini procesos.

Ventajas:

*

Cuando una hebra se bloquea, puede ejecutarse otra hebra del mismo proceso.

*

Es menos costoso crear/terminar una hebra que un proceso.

*

Es menos costoso intercambiar dos hebras que dos procesos.

*

La comunicación entre hebras dentro de una misma tarea no requiere intervención del núcleo del sistema operativo.

*

Cada hebra se puede estar ejecutando en un procesador distinto simultáneamente.

#

Procesos en Linux.
#

Imagen en Linux.

Está compuesta por:

*

Segmento de datos, divididos en:

*

Inicializados: no varían a lo largo de la vida del proceso.

*

No inicializados.

*

Segmento de texto o código: puede ser compartido por varios procesos.

*

Segmento de pila. Lo crea el núcleo al arrancar el proceso. Compuesto por marcos de pila. (Parámetros de la función, variables locales) Dos tipos:

*

Usuario: para las funciones que se ejecuten en modo usuario.

*

Núcleo: para las funciones que se ejecuten en modo núcleo.

*

BCP.

El tamaño de la tabla de proceso es fijo. Reserva 4 procesos para el superusuario. Tiene 6 estados: Nuevo, listo, bloqueado, terminado y dos estados de ejecución. (Modo usuario y modo supervisor)
#

Hilos en Linux.

Un proceso puede tener muchos hilos.

*

Proceso:

*

Unidad propietaria de recursos.

*

Tiene asignado un espacio de direcciones.

*

Tiene recursos asignados.

*

Hilo:

*

Posee un estado.

*

El contexto del procesador.

*

Una pila de ejecución.

*

Almacenamiento para variables locales.

*

Tiene acceso a la memoria y recursos asignados al proceso.

Llamada a Fork.

Crea un proceso. Pasos que realiza:

*

Busca una entrada en la tabla de procesos.

*

Se copia toda la información del proceso padre en las estructuras del proceso hijo.

*

Le asigna un identificador.

*

Comparte con el proceso padre todos los segmentos, utilizando una técnica “copia en escritura”, si se intenta modificar algún segmento se hace una copia y se modifica.

#

Planificación de la CPU.
#

Planificador de procesos.

La multiprogramación produce un aumento de la utilización de la CPU y una mayor productividad del sistema.

La planificación es un asunto de manejo de colas para minimizar el tiempo de retraso en las colas y optimizar el rendimiento del sistema. La planificación de la CPU es necesaria en los sistemas multiprogramados debido a que el número de procesos que se quiere ejecutar supera el número de procesadores existentes en el sistema.
#

Niveles de planificación del procesador.
#

Planificador a largo plazo.

Cuando termina/comienza un proceso. Determina que trabajo se admite en el sistema para su procesamiento. Decide cuántos trabajos acepta y qué trabajos.

Con la primera decisión controla el grado de multiprogramación del sistema. La segunda decisión puede basarse en distintos criterios:

* Prioridad.
* Puede elegir entre los orientados a R/S y los orientados a CPU.
* FIFO.

Se ejecuta con una frecuencia menor y puede tardar más tiempo en decidir que trabajo seleccionar.
#

Planificador a medio plazo.

Para intercambiar procesos. Descarga y carga procesos de memoria activa, reduciendo el grado de multiprogramación. Suspende y reanuda procesos para que el rendimiento del sistema se mantenga dentro de unos niveles.
#

Planificador a corto plazo.

El próximo en ejecutarse. Selecciona entre los trabajos que están en la cola de listos a cual asignarle la CPU. Se ejecuta cada vez que ocurre un evento que puede causar un cambio de proceso:

* Interrupción de reloj.
* Interrupción de E/S.

*

Llamada al sistema.
*

Señales.

Posee un módulo que se llama despachador:

*

Cambia de procesos.
*

Cambia al modo usuario.
*

Salta a la instrucción que le corresponda ejecutarse en ese instante.

Al tiempo que tarda en hacer estas operaciones se le denomina latencia de despacho.
#

Características del planificador a corto plazo.
#

Criterios del planificador a corto plazo.

*

Criterios relacionados con el rendimiento, orientados al usuario.

*

Tiempo de respuesta: tiempo que transcurre desde que se envía un trabajo al sistema hasta que se empieza a atender. (Sistemas interactivos)

O tiempo que transcurre desde que el proceso entra en la cola de listos hasta obtener la primera atención de la CPU.

*

Tiempo de retorno: tiempo total que emplea el proceso desde que entra en el sistema hasta que termina su ejecución. (Sistemas batch)

*

Plazos. (STR)

*

Otros criterios orientados al usuario.

Previsibilidad: Un trabajo debería de tardar en ejecutarse el mismo tiempo independientemente...

*

Criterios relacionados con el rendimiento, orientados al sistema.

*

Rendimiento: El número de trabajos que se ejecuta por unidad de tiempo (Throutput), sea el máximo posible.

*

Utilización del procesador: el porcentaje de tiempo que el procesador está ocupado.

*

Otros criterios orientados al sistema.

*

Equidad: en ausencia de criterios externos, todos los procesos deberían ser tratados de igual forma.

*

Prioridades: el algoritmo debe favorecer a los procesos con mayor prioridad.

*

Ocupación de recursos: deben intentar mantener los recursos ocupados el mayor tiempo posible.

#

Elementos del planificador a corto plazo.

Debe contener los siguientes aspectos:

*

Un modo de decisión.

*

Una función de selección.

*

Una regla de arbitraje.

#

Modo de decisión.

Indica en qué instante en el tiempo se aplica la función de selección. Las decisiones del planificador deben efectuarse en una de las cinco circunstancias siguientes:

1ª. Cuando un proceso cambia del estado de ejecución al estado de bloqueado.

2ª. Cuando un proceso cambia del estado de ejecución al estado de listo.

3ª. Cuando un proceso cambia del estado de bloqueado al estado de listo.

4ª. Cuando llega un proceso nuevo. Del estado de nuevo al estado de listo.

5ª. Cuanto termina un proceso.

Cuando la planificación tiene lugar únicamente en las situaciones 1ª y 5ª, decimos que el esquema de planificación es no apropiativo. Solo pierde el control del procesador cuando se bloquea por una operación de E/S o porque ha terminado.

En los otros casos decimos que tenemos un esquema de planificación apropiativo. Se le puede retirar el dominio del procesador a un proceso.

*

Planificación apropiativa.

El proceso que se está ejecutando puede ser interrumpido en cualquier momento y movido a la cola de listos, antes de su completa ejecución.

*

Planificación no apropiativa.

Una vez asignada la CPU a un proceso, no se le puede retirar hasta que la libera, ya sea por que termina la ejecución o porque cambia a un estado de espera. (se auto bloquea)
#

Función de selección.

Algoritmo que decide a que proceso s4 le asigna la CPU.

#

Regla de arbitraje.

En caso de que al aplicar uno de los algoritmos de selección, resulte un empate, alguna regla para seleccionar uno determinado.
#

Algoritmos de planificación.
#

Primero en llegar, primero en salir (FIFO).

Se les asigna la CPU conforme al orden de llegada. Selecciona al proceso más viejo en la cola de listos, situado al principio de la cola.

Es no apropiativo. No es útil en sistemas interactivos, al no garantizar buenos tiempos de respuesta. Es el más sencillo, se implementa con una cola FIFO. Es justo, pero los procesos largos hacen esperar a los procesos cortos.
#

El trabajo más corto primero (SJF).

Se le asigna la CPU al proceso que tenga un tiempo estimado de ejecución más corto. Los procesos cortos pasan a la cabeza de la cola de listos.

Reduce los tiempos de espera. Favorece a los procesos cortos frente a los largos. No es apropiado para tiempo compartido por ser no apropiativo. Lo difícil es predecir que tiempo va a durar cada proceso.

'Sistemas Operativos'
'Sistemas Operativos'
'Sistemas Operativos'
#

El tiempo restante más corto (SRT).

El siguiente en ser ejecutado es el proceso de tiempo estimado de ejecución menor para llegar a su terminación, incluyendo las nuevas llegadas.

Es una variante apropiativa del anterior. Tiene cierta sobrecarga al tener que llevar registro del tiempo transcurrido de proceso. Los procesos cortos se ejecutan casi inmediatamente y los largos tienen mayor retraso que en SJF.
#

Prioridades.

Consiste en asignarle a cada proceso una prioridad, consistente en un número. Las prioridades pueden determinarse:

* Internamente por el sistema.
* Externamente por el usuario.

Las prioridades pueden ser:

* Estáticas: una vez que se le asigna una al proceso, esta no cambia.
* Dinámicas: las prioridades cambian, adaptándose al ambiente.

El mayor problema de las prioridades es la inanición. Una solución es la técnica del envejecimiento, que consiste en cada cierto período de tiempo de espera, se le aumenta la prioridad al proceso.
#

Asignación en rueda (Round Robin).

Se define una pequeña unidad de tiempo denominada quantum. A cada proceso se le asigna un quantum de tiempo. Si no termina en ese periodo, el control de la CPU pasa al siguiente proceso de la cola.

La cola de procesos en estado de listo se organiza según un esquema FIFO. Es un sistema con modo de decisión apropiativo basado en el reloj. Es apropiado para sistemas de tiempo compartido. El problema es determinar el tamaño del quantum:

* Pequeño: todo el tiempo se pierde en el cambio de contexto.
* Grande: degenera en un FIFO.

#

Razón de respuesta más alta (HRRN).

Se selecciona el proceso de la cola que tenga un valor de RR más alto. Es no apropiativo.

'Sistemas Operativos'
#

Múltiples colas.

Se crean distintas colas a las que son asignados los trabajos. Cada cola puede tener su propio algoritmo de planificación. Los procesos nunca cambian de cola.

Tipos de cola:

*
  Primer plano.
*
  Segundo plano.
*
  Tareas del sistema.
*
  Interactivas.
*
  Edición.
*
  Lote.
*
  Superusuario.
*
  Usuarios privilegiados.
*
  Usuarios trabajadores.
*
  Usuarios que juegan.

#

Múltiples colas con realimentación.

Este algoritmo consigue:

* Favorecer los trabajos cortos.
* Favorecer los trabajos limitados por E/S.
* Determinar la naturaleza del proceso lo más rápidamente posible.
* Un proceso entra por la cola superior y espera hasta obtener la CPU.
* Sale de la red si termina la ejecución o realiza una operación de E/S.
* Si no termina en el quantum baja de nivel y entra en la parte trasera de la cola.
* Los procesos de una cola de nivel inferior no se ejecutan hasta que la de nivel superior está vacía.
* La última cola funciona como una cola circular.

Puede presentar inanición. Para evitar esto:

* Permitir subir de cola periódicamente o si no agota el quantum.

Dar varias vueltas en cada cola antes de bajar de nivel.

*

Recordar la cola de donde salió.

Parámetros:

*

Número de colas.

*

Algoritmo de planificación por cada cola.

*

Modo de subir de cola.

*

Modo de bajar de cola.

*

En qué cola entra inicialmente.

#

Evaluación de algoritmos de planificación.

Primero tendremos que decidir un criterio de selección:

*

Maximalización de la utilización de la CPU.

*

Maximalización de la productividad.

Para comparar los algoritmos tendremos en mente los siguientes tiempos:

*

Tiempo de retorno (Tq): Tiempo total.

*

Tiempo de respuesta: desde que entra hasta que toma por primera vez la CPU.

*

Tiempo de espera: tiempo en la cola de listo.

*

Tiempo de retorno normalizado: Tq/Ts (Tiempo de servicio)

Los métodos de evaluación son:

*

Evaluación analítica:

*

Modelo determinista.

*

Modelos de colas.

*

Simulación

*

Implementación: se implementa y se prueba. Si funciona, bien, sino se cambia. Es muy costoso. No se utiliza.

#

Planificación en Linux.

Es un round-robin de q=0'1 segundos, pero con prioridades. Utiliza un contador = prioridad + tiempo de espera. En función de este contador, ordena la cola de listos.

Las prioridades: - 20: Máxima. Superusuario = -5

20: Mínima. Nuevo proceso = 0

El contador: 1.000: Tarea de tiempo real. (Preparado para STR blando)

-1.000: Proceso bloqueado.
#

Sincronización y comunicación.
#

Introducción.

Multiprogramación consiste en ir ejecutando varios procesos simultáneamente en una CPU. Los procesos se intercalan.

Multiproceso consiste en tener n procesos y además n CPU. Se pueden ejecutar procesos simultáneamente, solapándose.

Procesos concurrentes:

* Sistemas multiprogramados: son los que están activos en el sistema. (en ejecución o en la cola de listos o bloqueados)
* Multiprocesadores: Son los que se están ejecutando.

Concurrencia, presenta tres tipos de problemas:

* Todos los procesos comparten unos recursos globales.
* La gestión óptima de los recursos no es fácil.
* Dificultad en localizar los errores de programación.

La concurrencia aparece:

* Surge cuando queremos ejecutar varias aplicaciones simultáneamente.
* En las aplicaciones estructuradas. Están formadas por varios procesos.
* En la propia estructura del sistema operativo.

Con la concurrencia, el resultado de un proceso es independiente de la velocidad relativa de ejecución del proceso.
#

Interacción entre procesos.

Se pueden clasificar según el grado de interacción en tres:

*

No tienen conocimiento de la existencia de otros procesos. Son independientes. Los procesos compiten por los recursos.

*

Tienen conocimiento indirecto de la existencia de otros procesos. Los procesos comparten un recurso común. Los procesos se coordinan pera usar ese recurso común.

*

Tienen conocimiento directo de la existencia de otros procesos. Los procesos trabajan en una fin común. Los procesos se comunican entre sí.

#

Competencia entre procesos por los recursos.

Se dice que dos procesos están compitiendo cuando quieren acceder al mismo tiempo a un recurso. Este recurso es único y no compartido: Recurso crítico.

Problema que se dan cuando los procesos compiten por recursos:

*

Exclusión mutua: Dos procesos no pueden ejecutar a la vez la sección crítica. Un proceso posee una sección crítica y otra no crítica. Sección crítica: sección del código que hace uso del recurso crítico.

*

Interbloqueo: dos procesos se bloquean mutuamente.

*

Inanición o bloqueo indefinido: un proceso se queda a la espera de un recurso, de manera indefinida.

#

Cooperación entre procesos por compartición.

Tenemos varios procesos que comparten un recurso común. Estos recursos normalmente poseen operaciones de lectura y de escritura. Tienen los problemas de exclusión mutua, interbloqueo e inanición pero en las operaciones de escritura. Además aparece el problema de la coherencia de datos:

Por ejemplo: se tiene que cumplir siempre a = b

P1 P2

a = a + 2 a = a * 2

b = b + 2 b = b * 2
#

Cooperación entre procesos por comunicación.

Saben de la existencia de otros procesos y además saben la identidad de esos procesos. Los procesos necesitan intercambiar información, que se realiza a través de mensajes. Mecanismos proporcionados por el sistema operativo.

Los problemas que se plantean son el interbloqueo y la inanición.

interbloqueo: P1 espera un mensaje de P2

P2 espera un mensaje de P1

Inanición: Un proceso espera un mensaje de un proceso que no existe o que no sabe que está esperando el mensaje.
#

Exclusión mutua.

Tenemos n procesos y un recurso crítico. Cada proceso posee una sección crítica y otra no crítica. No puede haber dos procesos ejecutando su sección crítica simultáneamente.
#

Requisitos para las soluciones a la exclusión mutua.
#

Debe de garantizar la exclusión mutua.
#

Un proceso que está bloqueado en la sección no crítica, no debe interferir en las demás.
#

La solicitud a la sección crítica no debe postergarse de forma indefinida. (No haya interbloqueo ni inanición)
#

Si no hay procesos en la sección crítica y un proceso quiere hacerlo, se realiza inmediatamente.
#

No se deben hacer suposiciones sobre la velocidad relativa ni del número de procesos.
#

Un proceso estará en la sección crítica por un tiempo finito.

Tipos de soluciones:
#

Sin ningún tipo de ayuda Algoritmos. Son las más complejas.
#

Con ayuda del hardware.
#

Con ayuda del sistema operativo Semáforos.
#

Con ayuda del lenguaje de programación Monitores.

Si no existe memoria compartida: mediante paso de mensajes.
#

Soluciones software.

Toda la responsabilidad, recae sobre el programador. Primer intento: se definía una variable global, turno. Problemas: Debe haber una alternancia estricta, en la utilización del recurso. Y es un bucle de estera ocupada, usa ciclos de CPU.

La primera solución válida la dio Dekker. Era muy compleja. Peterson hizo una más simplificada.

Después se generalizó para n procesos en el algoritmo de Lamport o de la panadería.
#

Soluciones con ayuda del hardware.

Desabilitando las interrupciones. No es válida. Se pensó en unas instrucciones especiales, las cuales se caracterizan por tener una ejecución atómica:
#

Test_and_set. (comprobar y establecer)
#

Exchange. (intercambiar)

Ventajas:
#

Es muy simple y fácil.
#

Es independiente tanto del número de procesos como del número de procesadores.
#

Se puede generalizar a n recursos críticos.

Inconvenientes:
#

La implementación en un sistema multiprocesador es bastante compleja, independientemente de sí tiene o no, memoria compartida.
#

Consume ciclos por que es una espera ocupada.
#

Es posible la inanición.
#

Es posible que se dé el abrazo mortal, pero es independiente al algoritmo. Es por culpa del algoritmo de planificación de la CPU: por prioridades, apropiativo.

P1(5) P2(3) C P3(10) Mayor prioridad.

L1 = 0 L2 = 0 1

L1 = 1 L2 = 0 0 Toma la CPU

s. c. espera L3 = 0

ocupada espera ocupada

En espera Interbloqueo En espera

de la CPU del recurso
#

Semáforos. (Con asistencia del SO)

Dijkstra, en 1965, introduce los semáforos. Definición: variable entera no negativa. (>=0) Sólo puede ser modificada mediante dos operaciones atómicas. (wait y signal)

Wait(S): Si S>0

entonces S = S - 1

Si no

bloquear proceso.

Signal(S): Si hay procesos bloqueados en S

Entonces despertar a uno de ellos

Si no

S = S + 1

Propiedades:
#

Wait y signal son las únicas operaciones que puede modificar el valor del semáforo y son atómicas.
#

Siempre tendrá valores iniciales no negativos.
#

Cuando existe más de un proceso bloqueado en la cola del semáforo, se despierta a un proceso de forma aleatoria.
#

Wait y signal siempre se van a ejecutar de forma secuencial, tanto en sistemas monoprocesadores como multiprocesadores.
#

Cumple las siguientes invarianzas:

S >= 0

S = S0 + nº signal - nº wait

Puede presentar inanición. Para ello se organiza la cola de bloqueados como una cola FIFO. Posee un inconveniente, y es que es el programador el encargado de poner los wait y signal.

Por ejemplo: productor - consumidor. Permite tener n productores y m consumidores.

Estos semáforos son semáforos contadores o generales. Existen otro tipo de semáforos que sólo puede tomar valores 0 ó 1.
#

Monitores. (Con ayuda de los lenguajes de programación)

Introducidos por Hoare, en 1974. Es un constructor de un lenguaje de programación concurrente. ADA, modula-2, modula-3, pascal-plus, pascal concurrente.

Se caracteriza por tener 2 elementos:

*

Datos variables. Sólo pueden ser accedidos por estos procesos.
*

Procedimientos que modifican estos datos.

Consigue la exclusión mutua, asegurando, garantizando que sólo habrá un proceso dentro del monitor. Inconveniente: supongamos que el P1 ejecuta el Monitor1 y dentro del Monitor1 se llama a Monitor2. P2 está ejecutando Monitor2 y quiere Monitor1 interbloqueo.

Para resolver este problema aparecen las variables de condición. Para cada variable condición existe asociada una cola de procesos.

Cwait: siempre suspende al proceso.

Csignal: Cuando un proceso ejecuta csignal, se reanuda el primer proceso que esté en la cola, sino, no hace nada.

Diferencias entre un semáforo y una variable condición:
#

La variable condición no tiene un valor mientras que el semáforo sí.
#

La operación cwait de la variable condición siempre va a suspender a un proceso, mientras que wait del semáforo lo bloque sólo si vale 0.
#

La operación csignal sólo actúa cuando haya procesos suspendidos, mientras que el signal del semáforo despierta un proceso o incrementa el semáforo, siempre hace algo.
#

- Semáforos: Consiguen la exclusión mutua y la sincronización. La responsabilidad recae en el programador.

- Monitor: nos garantiza la exclusión mutua. De manera simple y fácil.

- Variable de condición: nos permite la sincronización. De manera simple y fácil.
#

Paso de mensajes.

*****
#

Mecanismos de concurrencia en Linux.

*****
#

Interbloqueos.
#

Introducción.

El número de recursos es finito y limitado y normalmente es considerablemente menor que el de procesos. Los recursos pueden ser la CPU, memoria, dispositivos (impresoras, disco, cinta), ficheros, interrupciones, semáforos. Los procesos compiten entre sí, por esos recursos.

Un proceso puede solicitar como máximo el número máximo de recursos del sistema.

El interbloqueo también llamado deadlock, bloqueo mutuo o abrazo mortal, es: el bloqueo permanente de un conjunto de procesos que o bien compiten por recursos del sistema, o bien se comunican entre ellos.

Otra definición: interbloqueo es el conjunto de recursos bloqueados, donde cada proceso espera una señal que nunca se va a producir.

Inanición: se da cuando un proceso no toma nunca el control de un recurso.

No existe una solución óptima a diferencia de la exclusión mutua. Que se produzca un interbloqueo depende del orden concreto en que se ejecuten las órdenes.
#

Ejemplos de interbloqueos.

*

Con recursos reutilizables

P1 P2

solicita(f) solicita(c)

... ...

solicita(c) solicita(f)

... ...

libera(c) libera(f)

... ...

libera(f) libera(c)

*

Con recursos consumibles

P1 P2

receive(P2, M) receive(P1, Q)

send(P2, N) send(P1, R)

#

Condiciones necesarias para el interbloqueo.

Según Coffan, Elphick, Sho, deben darse 4 condiciones para que se produzca interbloqueo:
#

Exclusión mutua, es decir, que existan recursos que deban ser usados en exclusión.
#

Condición de retener y esperar, es decir, un proceso retiene los recursos mientras está a la espera de que se le asignen otros recursos.
#

No apropiación: un proceso que tiene asignados una serie de recursos, no se le puede apropiar esos recursos.
#

Espera circular: hay una cadena de procesos en la cual cada uno tiene asignados uno o más recursos que son requeridos por el siguiente proceso de la cadena.

Las condiciones 1, 2 y 3 son necesarias pero no suficientes. Se deben dar las cuatro (necesarias y suficientes)

Las tres primeras son decisiones del diseño del sistema operativo. Mientras que la cuarta es una circunstancia que se puede dar.
#

Modelado del interbloqueo.

Para ello se recurre al grafo de recursos del sistema o grafo de asignaciones de recursos:

*

Vértices:

*

círculos: procesos

*

cuadrados: clases o tipos de recursos, con tantos círculos en el interior como número de recursos haya en el sistema.

*

Arcos orientados:

P R: arista de solicitud. P está solicitando un recurso de tipo R.

R P: arista de asignación. Un recurso de tipo R asignado a un proceso P.

Del tipo de R P: El proceso P produce el recurso R.

Ejemplos:

Interbloqueo No interbloqueo

El interbloqueo aparece en el grafo cuando haya un grafo circular.
#

Estrategias para tratar los interbloqueos.

Hay tres tipos de soluciones:

*

Prevención: intentan prevenir que aparezca un interbloqueo negando una de las tres condiciones.

*

Predicción: hace un reparto cuidadoso de los recursos, intentando predecir cuando se puede producir un interbloqueo. Intenta que no se dé la cuarta condición.

*

Detección y recuperación: permiten el interbloqueo pero periódicamente comprueba el sistema. Si existe interbloqueo, recupera el sistema.

Las dos primeras utilizan un protocolo que asegura que no se produzca interbloqueos. La tercera, deja que se produzca el interbloqueo, detectarlo y recuperar el sistema. No hacer nada.
#

Prevención.

Es la técnica que más se suele emplear, por ser simples y no necesitar mucha información. Niega una de las cuatro condiciones.

*

Negación de la exclusión mutua

No es posible negar esta condición por que existen recursos que requieren un acceso exclusivo (impresora, fichero en modo escritura)

*

Negación de retener y esperar

Hace que el proceso nunca retenga recursos mientras está esperando. Tiene tres estrategias:

*

Todo o nada: se fuerza al proceso a solicitar todos los recursos que necesita, a lo largo de su ejecución. Si se le puede adjudicar, se ejecuta, si no, se bloquea el proceso. Inconvenientes: baja utilización de los recursos y posibilidad de inanición.

*

División de peticiones: se divide el programa en fases que se ejecuten de manera independiente. Antes de empezar una fase se deben solicitar todos los recursos que necesita en esa fase. Al terminar esa fase se liberan los recursos y se solicitan los de la fase siguiente. Igual que todo o nada pero por etapas. Inconvenientes: dificulta el diseño de aplicaciones.

*

Petición incremental y liberación: El proceso solicita los recursos a medida que los necesita. Si está disponible se le asigna, si no, tiene que liberar los que tenga asignados. Y debe pedir los recursos que necesita más los que ya tenía asignados. Este método no es aplicable a todos los recursos. Solo es aplicable a recursos que mantengan la integridad del sistema, que cuando lo recupere, sea fácil de restaurar.

*

Negación de la no apropiación

En la no apropiación el proceso libera el recurso voluntariamente. No se le puede quitar a la fuerza. Al negar esta, el sistema operativo, puede retirar un recurso a un proceso. El sistema operativo tiene que guardar el estado del recurso.

Algoritmo de prevención que niega la condición de no apropiación:

proceso Pi solicita un recurso

si esta disponible

entonces

se le asigna el recurso al proceso Pi

en caso contrario

si asignado a proceso Pj en espera de recursos

entonces

el sistema se apropia del recurso

se asigna el recurso al proceso Pi

en caso contrario

el proceso Pi pasa a estado de espera del recurso

fin-si

fin-si

A la hora de apropiar un recurso hay que tener en cuenta si es fácil restaurar el estado del recurso y si no hay pérdida de información. La CPU se puede apropiar, la impresora no.

*

Negación de la espera circular

Se ponen dos condiciones:
#

Se establece un orden entre los recursos: r1 < r2 < ... < rn

Se establece el siguiente protocolo:

Un proceso puede solicitar ri solo

Si tiene asignados rj < ri

Si un proceso solicita ri

Tiene que liberar rj > ri
#

Se exigen a los procesos que pidan los recursos en ese orden.

Problemas:

* Escoger el orden de los recursos.
* La libertad de programador esta limitada.
* Si se añade un nuevo recurso se debe de reordenar y por lo tanto reescribir todos los programas.

#

Predicción.

Permite las tres primeras condiciones y evita la espera circular. Requiere una información:

* De los recursos: cuantos existen, cuales están disponibles.

De los procesos: que recursos necesitan, que recursos tienen asignados.

Dos enfoques:
#

Denegar la inicialización del proceso

Para ello debemos de disponer de información sobre el sistema.

*

Vector de recursos: unidades totales de cada tipo de recurso.

Recurso = [Ri] = (R1, R2,..., Rn)

*

Vector de disponibles: unidades disponibles de cada tipo de recurso.

Disponible = [Di] = (D1, D2,..., Dm)

*

Matriz demanda máxima: exigencias máximas de cada proceso. Demanda del proceso i de recurso del tipo j.

DM11 ... DM1m

Demanda máxima = [DMij] = ... DMij ...

DMn1 ... DMnm

*

Matriz asignación: recurso de tipo j asignado al proceso i.

A11 ... A1m

Asignación = [Aij] = ... Aij ...

An1 ... Anm

#

'Sistemas Operativos'
: la cantidad de recursos del sistema es igual a la que está disponible más la que está asignada.
#

'Sistemas Operativos'
: la demanda de recursos nunca podrá ser mayor que los recursos del sistema.
#

'Sistemas Operativos'
: a un proceso solo se le podrá asignar como máximo lo que él solicitó.
#

'Sistemas Operativos'
: para admitir a un proceso el número de recursos debe ser igual o mayor que el número de demanda máxima actual más la demanda del nuevo proceso. Ejemplo:

R1 = 7

P1 1

P2 3

P3 2 6 + 2 = 8 no <= 7 no se inicia

P4 2

No es óptimo por que se basa en la máxima demanda.
#

Denegación de asignación de recursos

Además hace falta otra matriz:

Matriz necesidad: Nij es la necesidad del proceso i respecto del recurso j.

N11 ... N1m

Necesidad = [Nij] = ... Nij ...

Nn1 ... Nnm

Necesidad = Demanda - Asignación

Algoritmo del banquero

Dijkstra (65). Si un sistema siempre está en estado seguro, nunca se va a producir interbloqueo.

Estado del sistema: se define como el estado actual de asignación de los recursos a los procesos. Viene dado por las matrices de Demanda, Asignación y Necesidad.

Sistema seguro: existe un orden en el cual todos los procesos se pueden ejecutar hasta finalizar su ejecución sin que se produzca interbloqueo. Si no es posible encontrar esta secuencia, el estado está inseguro. Un estado inseguro indica una posibilidad de que se dé interbloqueo.

Estos algoritmos consisten en intentar que el sistema esté siempre en estado seguro.

Inconvenientes:

*

Los procesos tienen que declarar por anticipado esa demanda máxima.

*

El número de procesos del sistema debe ser constante.

*

El número de recursos del sistema debe ser constante.

*

El coste del algoritmo es bastante elevado. El número de operaciones que hay que realizar es n2 * m, donde n es el número de procesos y m es el número de recursos.

*

El algoritmo asume el peor caso. Los procesos usan todos los recursos al mismo tiempo.

*

Los procesos que requieren muchos recursos están muy castigados, al estar constantemente denegándole la asignación de recursos.

*

Los procesos del sistema deben ser independientes.

*

No se sabe cuanto tiempo puede estar un proceso esperando un recurso.

#

Detección y recuperación.

Permite que se produzca interbloqueo. No impone condiciones a los procesos. Supone una sobrecarga del sistema.

Lo primero es ver cuando ejecutar ese algoritmo, el de comprobación de interbloqueo:

*

Cada vez que se asigne un recurso a un proceso.

*

Cada solicitud denegada: algoritmo más sencillos pero sobrecargan al sistema por que están constantemente ejecutándose.

*

Cada cierto tiempo: se puede ejecutar más veces de las que realmente son necesarias.

*

Cuando el rendimiento del sistema cae por debajo de algún nivel.

Detección por grafos. Un grafo puede ser reducido si el proceso no espera ningún recurso o está esperando recursos que están disponibles desaparecen todas las aristas de petición y espera y libera todos los recursos asignados. Si el proceso es productor de recurso, supondremos que se dispone de recursos suficientes para satisfacer todas las peticiones.

Ejemplo:

R2 Consumible P2, P3 productores

R1 y R3 Reutilizable

Para la recuperación existen varias técnicas:
#

Terminación de procesos:

*

Todos los bloqueados: se terminan y tienen que empezar otra vez. Tiene un alto coste.

*

Uno a uno: se elimina uno, si sigue habiendo interbloqueo se termina otro y así sucesivamente. Tarda más que la anterior, hay que ejecutar el algoritmo cada vez que se termina uno. Existen muchos criterios para terminarlos: prioridad, el que solicita más recursos, el que tiene más o menos recursos asignados, el que lleva más o menos tiempo ejecutándose.

#

Apropiación de recursos: tiene un problema. No se puede aplicar a todos los recursos.

Problemas:

*

Escoger la víctima: se usan criterios similares a los anteriores, pero hay que evitar la inanición, escoger siempre a la misma víctima

*

Apropiación del recurso: no todos se pueden apropiar.

#

Administración de la memoria.
#

Jerarquía del almacenamiento.

Principio de localidad o principio de cercanía: Durante el curso de la ejecución de un proceso, las referencias a memoria, tanto a instrucciones como a datos, tienden a estar agrupadas.

#

Carga de un programa en memoria.

Dirección lógica o simbólica: referencia a una localización de memoria independiente de la asignación actual de datos a la memoria. Habrá que traducirlo a memoria física.

Dirección relativa: caso particular de dirección lógica, en la cual la dirección se expresa como una localización relativa a algún punto conocido.

Dirección física: localización en memoria principal.

Las direcciones simbólicas deben traducirse a relativas y absolutas. Esta traducción se puede hacer en varios momentos:

En tiempo de compilación: el programa es poco flexible, el cargador recibe el programa con sus direcciones absolutas. Se debe conocer el esquema de memoria del sistema a la hora de hacer el programa.

En tiempo de carga: el cargador se encuentra con un programa que contiene direcciones relativas y va a sumarle la dirección de comienzo a cada dirección relativa. Pero si tuviéramos que sacar el programa de memoria, a la hora de volver a memoria, tiene que hacerlo en el mismo sitio.

En tiempo de ejecución: A medida que van haciendo falta las direcciones se van traduciendo, estén estas expresadas de forma simbólica o relativa.
#

Funciones del administrador de la memoria.

Las funciones del administrador de la memoria son las siguientes:

*

Llevar el control de toda la memoria, de las partes que están en uso y de que partes están disponibles.

*

Asignar y desasignar memoria a los procesos, según la necesiten.

*

Determinar donde va a situar los procesos en memoria principal.

*

Gestionar el intercambio entre memoria principal y secundaria.

Requisitos del administrador de memoria:

*

Proporcionar reubicación de procesos, posibilidad de que un proceso pueda cambiar a otra zona de memoria.

*

Proporcionar protección de los espacios de memoria de cada proceso, no permitir que un proceso acceda a direcciones de memoria que no le correspondan.

*

Proporcionar compartición de los procesos que se ejecutan simultáneamente, a nivel de código y datos.

#

Esquemas de asignación de la memoria principal.

#

Sistemas de monoprogramación.

Sólo hay que proteger el espacio de memoria del sistema operativo. Es necesario un registro límite que será protegido, el cual posee la dirección límite a la que puede acceder el usuario.
#

Sistemas de multiprogramación con particiones fijas.

Requiere políticas de gestión de memoria que permitan la colocación simultánea de más de un proceso en memoria. Se divide el espacio físico de direcciones en un número determinado de particiones de tamaño fijo. El número y tamaño de las particiones se determina en tiempo de generación del sistema.
#

Selección del tamaño de las particiones.

Va a determinar el grado de multiprogramación y el tamaño máximo de los procesos.
#

Algoritmo de colocación.

Si son del mismo tamaño no hace falta un algoritmo. Pero si las particiones son de distintos tamaños, podemos utilizar una cola independiente para cada partición, o una única cola para todas las particiones con los siguientes algoritmos:

* Primer ajuste.
* Siguiente ajuste.
* Mejor ajuste.

Peor ajuste.

#

Protección de las particiones.

Se emplean uno de los dos métodos siguientes:

*

Dos registros límites, superior e inferior.

*

Registro base y longitud de la partición.

Cada vez que una dirección es traducida, el hardware del sistema comprueba que está dentro de los límites del proceso y en caso erróneo se produce una interrupción.

Fragmentación.

Es el espacio de memoria que no es utilizado por ningún proceso. Existen dos tipo de fragmentación:

*

Interna: Es la memoria que tiene asignada un proceso pero no la utiliza. Es el espacio no utilizado de la partición.

*

Externa: Memoria que no está asignada a ningún proceso y que no se puede asignar por no existir un proceso del tamaño adecuado.

En sistemas con particiones fijas, se dan ambas fragmentaciones. Un sistema con una única cola de procesos tendrá menos probabilidad de

tener fragmentación externa que un sistema con una cola de procesos por partición. En los dos sistemas se da fragmentación interna.
#

Elementos de control.

Se usa una tabla de particiones con tres datos por entrada:

* Registro base.
* Tamaño de la partición.
* Estado: libre o asignada.

Inconvenientes.

* Restricciones de tamaño: El programa más grande que podemos ejecutar será el de la partición más grande.
* Fragmentación externa.

#

Multiprogramación con particiones variables o dinámicas.

Las particiones son variables en tamaño y número. Un programa no puede estar en varias particiones a la vez, es contigua. Los programas ocupan el espacio de memoria física que realmente necesitan. No existe fragmentación interna.
#

Compactación.

Consiste en agrupar todo el espacio libre en una sola zona, desplazando todos los programas a una zona de memoria. Requiere reubicación dinámica de las direcciones. No produce fragmentación externa.

Inconvenientes:

* Durante la reubicación no se puede hacer otra cosa, el sistema se para. No es válido para STR.
* Consume muchos recursos del sistema.
* Requiere gran cantidad de tiempo.

La compactación se puede realizar según los siguientes criterios:

* Cada vez que termina un proceso. Es muy costosa.
* Cuando tengamos un porcentaje de la memoria ocupada (70-80%). Es algo menos costosa, pero puede ser innecesaria.
* Cada cierto intervalo de tiempo. También puede ser innecesaria.
* Cuando existan procesos esperando. Es la mejor. Tiene que estar comprobando si hay procesos a la espera.

#

Algoritmo de colocación.

Se utilizan los mismos algoritmos que para los sistemas con particiones fijas.
#

Mecanismos de control del uso de la memoria.

Hay dos mecanismos para representarla:
#

Mapa de bits.

La memoria se divide en unidades de asignación. Puede variar según el sistema. A cada unidad de asignación se le hace corresponder un bit del mapa de bit.

*

  bit = 0 -> Unidad libre
*

  bit = 1 -> Unidad ocupada.

Las operaciones de asignación y desasignación se pueden realizar rápidamente. Pero es difícil de gestionar la búsqueda de huecos de un determinado tamaño.
#

Listas enlazadas.

Se usa una lista con los siguientes elementos:

*

Tipo (P ó H).
*

Dirección de comienzo.
*

Longitud.
*

Puntero al siguiente elemento.

Cuando termina un proceso hay que fusionar los huecos. Para optimizar esto se usa una lista doblemente enlaza: con un puntero al anterior. Sigue teniendo problemas a la hora de encontrar un hueco muy lejos, hay que recorrer mucha lista.

Para solucionar esto se usan dos listas doblemente enlazadas, una para llevar el control del espacio ocupado y la otra para llevar el control de los huecos libres. Es fácil encontrar huecos de determinado tamaño y la fusión de huecos adyacentes.

La lista de huecos se puede ordenar por direcciones (primer ajuste), por tamaño (>: peor ajuste, <: mejor ajuste),...

Se puede aprovechar la misma memoria libre. El tamaño de la lista será sólo el de la lista de procesos.
#

El sistema compañero.

Se limita el tamaño de memoria asignada a los procesos.

Se busca en la lista de bloques libres, un bloque de tamaño 2i. Si no hay ninguno, se coge uno de tamaño 2i+1 y se divide en 2 de 2i. A estos dos bloque se les llama bloques compañeros, a los dos que resultan de la división, cuando uno se libere, se vuelven a unir, si el otro está libre. Posee una lista por cada tamaño de bloque posible:

*

64K
*

128K
*

256K ...

#

Paginación.

El espacio de direcciones lógicas del programa se divide en fracciones de igual tamaño denominadas páginas. La memoria física también se divide en partes, del mismo tamaño que las páginas, denominados marcos de página.

Carga del programa.

Se realiza en varios pasos:
#

Se divide el espacio de direcciones del programa en páginas. Si P es el tamaño del marco de página, y T el tamaño del programa, número de páginas es N=T/P. Se redondea por exceso.
#

Se buscan N marcos libres en la tabla de marcos. Tienen que estar todas las páginas en memoria aunque no tienen por que estar contiguas.
#

Se asignan los marcos a las páginas, y se guarda en la tabla de páginas.

Tabla de marcos de página.

Contiene información de qué marcos están ocupados y cuales libres. Posee una entrada por cada marco de memoria física. Sólo existe una tabla de marcos de página, con tamaño fijo. El número de marcos viene determinado por el tamaño de la memoria física dividido por el tamaño de un marco.

Tabla de páginas.

Existe una tabla de páginas por cada proceso que hay en memoria. Se crea durante la carga del proceso. Tendrá tantas entradas como marcos pueda tener.
#

Páginas compartidas.

Las tablas de páginas de procesos distintos que apuntan a los mismos marcos de memoria física. En la tabla de marcos tenemos algún tipo de contador que nos indica cuantos procesos están compartiendo ese marco, para saber cuando no está siendo usado por ningún proceso y está libre para poder asignarle el marco a otro proceso distinto con otro código.
#

Protección.

Si la página tiene código, no se puede modificar, el acceso será en modo lectura. Si contiene datos se accederá en modo L/E. Esto se representa mediante bits de protección en la tabla de páginas.

Para evitar que el proceso intente acceder a zonas de memoria que no le correspondan, se usan los bit de validez, para cada una de las entradas de página. O registro de longitud de la tabla de página, contiene el número de páginas que posee ese proceso.
#

Fragmentación

Se puede dar fragmentación interna por que se le asignan páginas enteras. Por termino medio cada proceso tendrá media página de fragmentación interna.
#

Tamaño de las páginas.

Viene determinado por la arquitectura, proporciona varios tamaños, el diseñador lo elige.

Factores que influyen a la hora de elegir el tamaño de la página:

* Fragmentación interna: cuanto más pequeña la página, menos fragmentación interna.
* Tamaño de la tabla de páginas: cuantas más pequeñas las páginas, más páginas y por lo tanto más grande las tablas de páginas.

Si S es el tamaño medio de los procesos, P el tamaño de la página y E el número de bytes que ocupa una entrada en la tabla de páginas:

* S/P número promedio de páginas que ocupa un proceso.
* E*S/P tamaño medio que ocupa la tabla de páginas.
* E*S/P + P/2 cantidad de memoria extra que se gasta por proceso en el sistema.

Si derivamos esto e igualamos a 0, obtenemos el tamaño de página óptimo ==> P= 2*S*E

#

Segmentación.

El proceso se divide en segmentos. Los segmentos pueden ser de tamaños diferentes. En la tabla de segmentos se guarda la longitud y la dirección base de cada segmento. Hay una tabla de segmentos por cada proceso. Puede contener también bits de protección (modo L o modo L/E). Cada vez que queremos acceder a memoria, accedemos realmente 2 veces. Para ello se usan también las TLB.

La gestión del espacio libre y ocupado se puede hacer a través de los mapas de bits o de las listas enlazadas.
#

Traducción de direcciones.

Las direcciones lógicas vienen dadas por el número del segmento y por el desplazamiento. Registro STLR: longitud de la tabla de segmentos, STBR: dirección base de la tabla de segmentos.

En un sistema donde las direcciones lógicas constan de m bit y el tamaño máximo del segmento es 2n:
#

Se extrae el número de segmento de los (m-n) bits más a la izquierda de la dirección lógica.
#

Se comprueba si ... comparándolo con el registro de longitud de la tabla de segmentos.
#

Se usa el número de segmento como índice en la tabla de segmento, y
#

Se compara el desplazamiento, ... , con la longitud del segmento. Si el desplazamiento es mayor, sería una dirección no válida.
#

La dirección física se obtiene sumando a la dirección base del segmento, el desplazamiento.

Características de la segmentación.

* Se requieren dos accesos a memoria.
* Las direcciones físicas se van a obtener en tiempo de ejecución procesos reubicables.
* Se requiere una política de colocación de procesos en memoria.
* No se da fragmentación interna, pero sí externa.
* Los segmentos se pueden compartir

#

Segmentación paginada.

La memoria se divide en segmentos, pero a su vez estos segmentos, se dividen en páginas del mismo tamaño. Se elimina la fragmentación externa y el algoritmo de colocación de los segmentos en memoria.

La dirección lógica está formada por tres componentes: número de segmento, número de página y desplazamiento. Se tiene una tabla de

segmentos por proceso y una tabla de páginas por cada segmento del proceso.
#

Memoria virtual.
#

Introducción
#

Principios de operación.
#

Estructuras hardware y de control en la memoria principal.

Tabla de mapa de página.

Cada entrada en la tabla posee los siguientes elementos:

* Número de página.
* Número de marco.
* Bit de fallo: indica si la página se encuentra en memoria principal o no.
* Bit de modificación: indica si la página ha sido modificada o no.
* Bit de referenciado: indica si la página ha sido recientemente referenciada.
*

* Bit de protección: indica las operaciones permitidas.
* Bit de bloqueo: indica si el marco de página está o no bloqueado.

Tabla de mapa de fichero.

Su función es indicar al sistema en qué lugar de la memoria secundaria se encuentran las páginas de un proceso. Habrá una entrada por cada página del proceso, con los siguientes elementos:

* Número de página.
* Posición en memoria secundaria.

Tabla de marcos de página.

Su función es llevar el control de los marcos que están libres y ocupados. Cada entrada tiene los siguientes elementos:

* Número de marco.
* Identidad del programa.
* Número de página.
* Bit de estado.

Bit de referencia.

*

Bit de cambio.

Implementación de la tabla de página.

*

En conjunto de registros especializados.

Si el número de páginas es pequeño. Cada cambio de contexto hay que realizar la carga de los registros. Solo es posible en sistemas con pequeña memoria.

*

En memoria principal.

Si el número de páginas posibles es alto. Inconvenientes:

*

Para acceder a una posición de memoria necesitamos dos accesos a memoria.

*

Se ralentiza el acceso a memoria por un factor de dos

*

En memoria caché (TLB).

En estas memorias no estarían las tablas completas, sino las entradas correspondientes a las páginas más usadas. El sistema buscaría primero en esta tabla rápida la página buscada.

La búsqueda en la memoria caché no se realiza por indexación, sino que es una búsqueda por contenido. En el TLB no están todas las páginas de cada proceso. Cada entrada en la TLB tiene que incluir el número de la página.

Métodos de reducción del tamaño de las tablas de página.

* Tablas de páginas de múltiples niveles.

Ejemplo: un sistema con una espacio de direcciones lógicas de 32 bits y con un tamaño de página de 4kb (212) necesita:

232/212 = 220 entradas en la tabla de páginas (1.048.576). Si cada entrada tiene 4 bytes, necesita 4Mb.

Para disminuir el tamaño de las tablas de página se implementan tablas de más de un nivel. Ejemplo: Un sistema con direcciones de 32 bits lo dividimos en:

* 10 bits para el primer nivel: Una tabla de 210 entradas (1.024).
* 10 bits para el segundo nivel: 1.024 tablas de 210 entradas.
* 12 bits para el tamaño de la página: 1.024 * 1.024 páginas de 212=4Kb.

* Tablas de páginas invertida.

Se caracteriza por tener una entrada por marco de página. Solo tiene información de las páginas que están en memoria física. Cada entrada posee:

* PID del proceso.
* Número de página virtual.
* Bits de bandera.

Requiere de la tabla de páginas externa por procesos, que se encuentra en el almacenamiento secundario. Esto incrementa el tiempo de búsqueda de la tabla de páginas. Para reducir este tiempo se usa una tabla de dispersión (Hash): primero busca en la tabla hash y luego en la tabla de páginas.

Gestión de un fallo de página.

Si una página no pertenece al conjunto residente se produce un fallo de página, se comunica al sistema operativo y este la busca en el disco:
#

Se produce una ecepción por parte del proceso y el control pasa al SO.
#

Se realiza un cambio de contexto.
#

Comprueba que tipo de ecepción se ha producido: un fallo de direccionamiento.
#

Comprueba la validez de la página y si tiene los permisos adecuados (tipo de acceso).
#

Obtener un marco libre. Si existen no hay problema, pero sino, tendrá que escoger uno y seleccionarlo como marco libre. Se sustituye la página, pero para ello se comprueba si la página a sido modificada y si es así, reescribirla en el disco. Este marco se marca como ocupado.
#

El sistema operativo realiza la petición al sistema de E/S de la página al disco. Cuando el dispositivo finalice producirá una interrupción para avisar que ha acabado.
#

Se realiza un cambio de contexto, si se estaba ejecutando otro proceso.
#

Actualizar la tabla de páginas y pasar el proceso al estado de listo. En el paso 6ª se produce una operación de E/S y para que la CPU no este ociosa, se bloquea al proceso.
#

El proceso solo tiene que esperar a que el planificador de procesos le otorgue la CPU.
#

Software del sistema operativo.
#

Política de lectura.

Qué elemento (página o segmento) hay que traer a memoria y cuándo. Puede haber dos posibilidades:

*

Por demanda: la demanda es la que dicta que elemento y cuando hay que traerlos. Es decir, el fallo de página. Inicialmente se producirán muchos fallos de página.

*

Prepaginación (paginación previa): conocer el futuro. No tiene mucha utilidad.

#

Política de colocación

Dónde colocar el elemento en memoria principal. Depende del esquema que haya debajo de la memoria virtual:

*

Si el sistema es paginado: Es inmediata, cualquier marco acepta a cualquier página. Todos los marcos son iguales.

*

Si el sistema es segmentado: Requiere de algún algoritmo: mejor ajuste, peor ajuste, siguiente ajuste,...

#

Política de sustitución.

Escoger qué marco es el que va a salir. Existen diferentes algoritmos:

*

Algoritmo óptimo.

*

Algoritmo FIFO.

*

Algoritmo LRU.
*

Algoritmo del reloj (de la segunda oportunidad).

No tan bueno como LRU pero más fácil de implementar. Se introduce un bit de referencia o bit de uso que se almacena en la tabla de páginas. Cuando la página se carga en memoria, ese bit se pone a uno, y cada vez que la página es referenciada. Considera al conjunto de páginas candidatas como un buffer circular.

Cuando se produce un fallo, si la página a la que apunta el puntero tiene el bit a 0 se selecciona, sino se le pone a 0 y se avanza el puntero.

*

Algoritmo del reloj mejorado.

Comprueba si la página ha sido modificada. Para ello utiliza un bit de referencia y un bit de modificación. Establece 4 tipos de páginas:

Referencia Modificación

0 0 No ha sido accedida recientemente ni ha sido modificada

0 1 Debe ser escrita. Mejor opción.

1 0 Ha sido accedida recientemente. Peor opción.

1 1 Ha sido accedida recientemente y además ha cambiado.

El algoritmo sigue los tres pasos siguientes:
#

Buscar una página (0, 0).
#

Si no hay

Buscar una página (0, 1)

Ir cambiando el bit de referencia, de las páginas que lo tengan a 1, a 0.
#

Si no hay volver al paso 1º.

Lo que intenta este algoritmo es reducir el número de escrituras al disco, al sacar primero las páginas que no se han modificado.

Almacenamiento intermedio. Buffering de páginas.

Es una técnica aplicable a cualquier algoritmo. Mejora el rendimiento del algoritmo. Las páginas no salen de memoria. Lo utilizan VAX/VMS y Linux, que considera 2 buffer distintos, uno con las páginas modificadas y otro que no.
#

Gestión del conjunto residente

El conjunto residente está formado por los marcos de páginas asignados y por las páginas candidatas.
#

Tamaño del conjunto residente

No todas las páginas del proceso se almacenan en memoria principal. El sistema operativo debe de decidir qué número de páginas le asigna a cada proceso.

Hay 3 consideraciones a tener en cuenta:

*

A menor cantidad de memoria a cada proceso, mayor número de procesos residentes en memoria.

*

Un número pequeño de marcos de página por proceso implica un número elevado de fallos de página.

*

A partir de un cierto número de marcos asignados por procesos, no influye en el número de fallos de página.

Existen dos tipos de políticas de asignación:

*

Fija: siempre mantiene un número de marcos fijos por proceso. Si provoca un fallo de página, habrá que sacar de memoria un marco de página del propio proceso.

*

Variable: Varía el número de marcos asignados por proceso. Es más compleja pero da mejores resultados, lleva una sobrecarga al sistema. Es conveniente un apoya hardware.

#

Alcance del reemplazamiento.

El alcance puede ser:

*

Local: solamente puede seleccionar una página de las que le pertenecen al proceso que provoca el fallo de página.

*

Global: Puede seleccionar todas las páginas residentes en memoria principal, excepto las bloqueadas.

Según la asignación y el alcance, tenemos las siguientes combinaciones:
#

Asignación fija y alcance local.

Cuando se produce un fallo de página se saca una de sus páginas. Se decide de antemano el tamaño del conjunto residente.
#

Asignación fija y alcance global.

No es posible por que al producirse un fallo de página, se escogería un marco de otro proceso. Un proceso aumentaría y el otro disminuiría.
#

Asignación variable y alcance global.

Cuando se produce un fallo de página, el sistema añade un marco de página más al proceso, si existen marcos libres. Si no hay marcos libres se selecciona una página víctima de entre todos los marcos de la memoria principal. Esto implica que a algún proceso se le reducirá el tamaño de su conjunto residente.

Es la más fácil de implementar. Se puede provocar la hiperpaginación: muchos procesos cargados en memoria principal con un conjunto residente muy pequeño. Esto produce muchos fallos de página.
#

Asignación variable y alcance local.
#

Se realiza una asignación fija al proceso en el momento de su carga en memoria principal.
#

Cuando se produce un fallo de página se selecciona una página de entre el conjunto residente del propio proceso.
#

Cada ciertos intervalos de tiempo, se examina su conjunto de trabajos con el fin de aumentar o disminuir el número de marcos. Esta estrategia es la más completa de todas.

Modelo del conjunto de trabajo.

Se basa en el principio de localidad. La cantidad de memoria que requiere un proceso se puede estimar basándonos en el comportamiento del programa en el tiempo.

Define el conjunto de páginas a las que hace referencia activamente el proceso.

W(t, ð) = conjunto de páginas referenciadas en las últimas ð unidades de tiempo virtual (Tiempo que tarda una instrucción en ejecutarse).

ð: ventana del conjunto de trabajo.

Es una función no decreciente respecto a ð W(t, ð) ð w(t, ð +1)

1 ð W(t, ð) ð min(ð, n) n: número de páginas del proceso.

El tamaño del conjunto de trabajo no es fijo. Puede decrementar si durante las últimas ð unidades de tiempo se hace más de una referencia a una misma página.

Inconvenientes:

*

Sobrecarga: lleva el control de qué páginas referencia cada proceso y una cola ordenada por tiempo, para cada proceso.

*

Determinar cual es el tamaño de la ventana óptimo:

*

Si es mayor: puede tener más marcos de los que necesita.

*

Si es menor: puede tener demasiados cambios.

*

Hay que evaluar el valor de ð en cada referencia a memoria.

Principio del grupo de trabajo:

*

Un proceso debe ejecutarse solo si su conjunto de trabajo reside en memoria principal.

*

Una página no debe ser retirada de memoria principal si es miembro del grupo de trabajo de un proceso.

Algoritmo de frecuencias de fallos de páginas (PFF).

Se usa un bit de uso: se pone a 1 cuando se hace referencia (y el sistema calcula el tiempo virtual transcurrido entre dos fallos de página).

Se establece un valor umbral F. Cuando se produce un fallo de página, si se supera el umbral de F, se descartan todas las páginas que tengan el bit a 0, y se pone a 0 las que lo tuvieran a 1. Si no se llega al umbral, se aumenta el conjunto residente con una nueva página, por que se están produciendo demasiados fallos de página. Se suele establecer un límite inferior y un límite superior.

#

Políticas de limpieza.

Hace referencia a cuando debe ser reescrita una página modificada.

*

Limpieza por demanda: se escribe la página cuando se necesita su marco. Se realizan las necesarias. Inconveniente: un fallo de página son dos accesos a disco.

*

Limpieza previa: se escriben antes de necesitarse.

Ventaja: un fallo de página sólo es un acceso a memoria.

Inconveniente: se realizan escrituras innecesarias.

Con almacenamiento intermedio de páginas se mejoran ambas.

#

Control de la carga e hiperpaginación.

Está relacionado con el grado de multiprogramación que tiene el sistema. Hiperpaginación: el grado de multiprogramación se eleva tanto que el sistema cae debido a que el conjunto residente de los procesos es muy pequeño y se producen muchos fallos de página.

*

Criterios a seguir para aumentar o disminuir el grado de multiprogramación:

#

Criterio L=S.

Mantener un grado de multiprogramación que haga que se cumpla la ecuación L=S, donde L: Tiempo medio entre dos fallos de página y S: Tiempo de servicio del fallo de página (tiempo medio requerido para procesar un fallo de página).
#

Criterio del 50%.

Mantener un grado de multiprogramación tal que el dispositivo de paginación está ocupado el 50% del tiempo.

*

Si hay que disminuir el grado de multiprogramación que criterio se va a seguir para seleccionar el proceso a sacar de la memoria. Depende del programador.

*

Menos prioridad del sistema.
*

El conjunto residente más grande.
*

El conjunto residente más pequeño.
*

Último proceso activado.
*

El que esté produciendo más fallos de página.
*

El proceso más corto, menos tiempo de ejecución.

#

Influencia del tamaño de la página sobre la tasa de fallos de página.

(Gráficas)
#

Gestión de memoria en Linux.

Utiliza una memoria segmentada-paginada. Utiliza un sistema paginado con demanda y un sistema de páginas intermedio.
#

Estructura de la memoria.

Espacio de direccionamiento = 232 = 4Gb, de 32 bits. El proceso de usuario solo puede acceder a 3Gb. El otro Gb. es para el núcleo, estructuras de datos, páginas de 1er y 2º nivel. A un proceso solo se le carga en memoria el BCP y la pila y a medida que lo pide, el resto de las páginas.

* La tabla de páginas tiene 32 bits.
* 20 bits para la dirección del marco.
* 3 bits del SO para política de sustitución.
* Bit de modificación.
* Bit de acceso o referencia.

*

Bit de tipo de página (usuario o núcleo).

*

Bit de protección.

*

Bit de presencia.

Si el bit de presencia esta a 0, la dirección almacena la dirección en memoria secundaria.

*

La tabla de marcos:

*

Bit de bloqueado: Cuando está implicada en operaciones de E/S.

*

Bit de requerido: Se activa cuando la página que solicitamos se encuentra en la lista de marcos libres.

*

Bit en transito.

*

Número de procesos que comparten esa página.

*

Demonio de paginación.

Se encarga de mantener la lista de marcos libres. Utiliza el algoritmo del reloj con una pequeña modificación: Bit de referencia y un contador, que

indica el uso que se hace de la página. Cuando se referencia se aumenta en uno y cuando pasa el demonio se decrementa, el marco que tenga (0, 0) está lista para ser sacado.

*

Área de swapping.

Puede ser una partición o un fichero. Solo se guardan las páginas de memoria que han sido modificadas. Utiliza un mapa de bits.
#

Traducción de direcciones.
#

La dirección virtual se traduce a dirección lineal, para ello se emplea el esquema de segmentación.
#

La dirección lineal se traduce a dirección física, se emplea es esquema de paginación. Tamaño de página de 4Kb y una tabla de páginas de dos niveles.

Dos tipos de fallos de página:

*

Fallo de validez: se intenta acceder a una página cuyo bit de presencia indica que no está cargada en memoria. Pero puede estar cargado en memoria en la lista libre.

*

Fallo de protección: se intenta acceder a una página que está en memoria pero los bits de acceso no me lo permite. Ejemplo: se intenta acceder a una página de lectura en modo escritura.

Llamadas al sistema disponibles:

*

brk: especificar tamaño del segmento de datos.
*

malloc: solicita memoria.
*

free: libera memoria.
*

swapon: activa área de intercambio.
*

swapoff: desactiva área de intercambio.

#

Sistema de Entrada/Salida
#

Introducción.

La parte del SO encargada es el sistema de E/S. Dos objetivos fundamentales:

*

Eficiencia: son más lentos y forman un cuello de botella en el sistema.
*

Generalidad: se pretende que todos los dispositivos de E/S se puedan tratar de la misma forma.

#

Organización del sistema de E/S.

Se estructura el software en capas:

*

Manejadores de dispositivos: se entienden con las controladoras del dispositivo y con el sistema de E/S.

*

Sistema de E/S: hace que todos los dispositivos se traten de la misma forma.

#

Dispositivos de E/S.

Se clasifican en:
#

Dispositivos de bloque: almacenan la información en bloques de tamaño fijo. Se puede leer/escribir los bloques de manera independiente (discos).
#

Dispositivos de caracteres: reciben o envían un flujo de caracteres. No son direccionables (impresoras, terminales).

Están formados por dos componentes:

*

Controladoras: componente electrónico. Actúan de intermediarios entre el sistema de E/S y los dispositivos.

*

Dispositivos: son los componentes mecánicos.

Puerto de E/S. Posee tres tipos de registros:

* Ordenes: lo usa la CPU para indicarle al dispositivo la orden a realizar.
* Datos: Almacenan datos de manera temporal.
* Estado: Lo utiliza el dispositivo para indicarle a la CPU su estado.

Cuando el espacio de direcciones es propio del dispositivo se habla de E/S mapeada en E/S. Pero cuando el espacio de direcciones es general, como el de la memoria, se habla de E/S mapeada en memoria.

Modos de realizar las operaciones de E/S:
#

E/S controlada por programa o escrutación.

El procesador tiene el control completo de la operación. Se encarga del proceso completo y no se puede dedicar a otras cosas.
#

E/S controlada por interrupciones.

El procesador coloca la orden en el registro de ordenes de la controladora, el proceso se bloquea y se ejecutaría otro proceso. Cuando el dispositivo termina, emite una señal de interrupción a la CPU. Se necesita además una tabla de estado de los dispositivos y un manejador de interrupciones.
#

E/S controlada por acceso directo a memoria (DMA).

Pasos:
#

El proceso de usuario solicita hacer una operación de E/S.
#

El procesador inicializa la operación de E/S. Le pasa a la controladora los datos necesarios: Lectura o escritura, dirección de memoria donde hay que empezar a leer/escribir y el número de palabras que van a ser leídas o escritas.
#

El proceso pasa a estado bloqueado y el procesador continúa con otro proceso.
#

La controladora va tomando el control del bus cuando sea necesario.
#

Cuando complete la transferencia, el bloque, la controladora emite una interrupción. El procesador toma otra vez el control y comprueba que la operación se ha realizado con éxito.
#

Se desbloquea al proceso.
#

Canales de E/S.

Compuesto por una unidad de control y un buffer de datos, utiliza la memoria del sistema. Posee un juego de instrucciones propio.

Funciones:

* Seleccionar dispositivos para realizar la E/S.
*

Enviar órdenes al dispositivo.
*

Define el área de memoria para realizar la transferencia de datos.
*

Define la acción a realizar cuando termina la transferencia de datos.
*

Indica a la CPU mediante una interrupción que la operación ha terminado.
*

Comprobar el estado del dispositivo.
*

Funciones de conversión de datos.

#

Principio del software de E/S.

Hay 4 capas:
#

Manejadores de interrupciones. Es el nivel más bajo.
#

Manejadores de dispositivos. Controladores del dispositivo. Es la única capa que está en contacto con el dispositivo. Reciben solicitudes del tipo “quiero el bloque físico 20” y tiene que traducirlo al cilindro, pista y sector correspondiente.
#

Software independiente del dispositivo. Es el software que se encarga de las funciones comunes para todos los dispositivos. Funciones:

*

Interfaz uniforme para los manejadores.

*

Nombre de los dispositivos: que se reconozcan por un nombre simbólico.

*

Protección de los dispositivos.

*

Uso de buffers: Dispositivos de bloques: cache de disco. Y dispositivos de caracteres: técnicas de buffering.

#

Proceso de usuario.

Algunos SO incluyen los manejadores en el núcleo. Cuando se quiere instalar un dispositivo se toca el código del núcleo y se recompila: Manejadores reconfigurables. Se enlaza dinámicamente las funciones de los manejadores de dispositivos al núcleo del SO. Para ello se usa la tabla de referencias indirectas.

Optimización de las operaciones de E/S.

*

Técnicas de buffering.

Área de memoria donde se almacena temporalmente los datos. Otra técnica es el doble buffer, cuando se dispone de 2 buffers. Cuando el número de buffer es mayor, se habla de buffer circular.

*

Caché de disco.

Ampliación del concepto de buffer. Dispone de varios sectores del disco accedidos recientemente almacenados en memoria. Política de sustitución: cualquiera de las vistas para la sustitución de páginas. Por ejemplo: LRU: Bloque menos recientemente referenciado. LFU: bloque referenciado menos veces.

*

Planificación de discos.

Objetivos:

*

.

*

Se debe minimizar el tiempo medio.

*

Se debe minimizar la varianza, la desviación de una petición con respecto al promedio.

Parámetros de rendimiento:

*

Tiempo de búsqueda: tiempo empleado en localizar la pista sobre la que queremos realizar la operación de E/S. Se puede expresar como: Ts=m*n+s

m = tiempo de saltar de una pista a otra.

n = número de pistas que debemos cruzar.

s = tiempo inicial de arranque de la cabeza.

*

Tiempo de latencia o demora rotacional: depende de la velocidad a la que gire el disco.

= 1⁄2 * r, r = tiempo que tarda en una revolución.

A la suma de estos dos tiempos se le conoce como tiempo de acceso.

*

Tiempo de transferencia = b / (W*n)

b = bytes a transferir, W = velocidad de rotación en rev/s, n = bytes por pista.

A la suma de los 3 tiempos: Tt = Ts + 1⁄2 * r + b / (W*n)

Es en el tiempo de búsqueda donde se centran los algoritmos de planificación de disco.
#

Planificación FCFS (FIFO).

Las peticiones se van atendiendo en el mismo orden de llegada. Es el más simple pero no cumple ninguno de los objetivos.

Número de pistas recorridas = 640
#

SSTF (shortest seek time first).

Primero el de tiempo de búsqueda más corto. Mira el número de pistas que tiene que recorrer y va hacia la más cercana. Garantiza que el desplazamiento entre peticiones sea el mínimo pero no la varianza de

esas pistas. Inconveniente: Se centra en las pistas más cercanas. Puede producir inanición en determinadas zonas del disco. Para dos pistas a la misma distancia, una política de desempate: orden de llegada,...

Número de pistas recorridas =236
#

SCAN.

Consiste en desplazar la cabeza de un extremo al otro e ir atendiendo las peticiones que nos vamos encontrando. Y luego hacia el otro extremo. No es posible la inanición, sin embargo, favorece a las pistas centrales frente a los extremos. Además tiene movimientos extra, si estamos en la 20 y se dirige hacia la 14 y el siguiente es en la pista 65, la cabeza llegará a la 0.

Número de pistas recorridas = 331
#

LOOK.

Algoritmo del ascensor. Intentan eliminar esos movimientos de llegada hasta los extremos. Cuando atiende la 183, no llega a la 199, si no que retrocede inmediatamente

Número de pistas recorridas = 299
#

C-SCAN.

Intenta favorecer a todas las pistas por igual, por lo que trata al disco como una lista circular de pistas, cuando llega al final salta inmediatamente al principio, solo hay un movimiento que atiende las pistas.
#

N-SCAN o planificación de exploración en n-etapas.

Con todos estos algoritmos, si llega un flujo continuado hacia la misma pista (p.e. 53), la cabeza no se mueve, se queda estancada en esa pista atendiendo esas peticiones. Excepto el FIFO.

Divide la cola de peticiones en subcolas de tamaño N. Hasta que no atienda la primera subcola no atiende a la 2ª. Cuando termina con la 1ª, empieza la 2ª,...
#

FSCAN.

Tiene dos colas de peticiones: Unas las atiende actualmente y en la otra cola están las peticiones que se recibieron durante la atención de la cola anterior.
#

E/S en Linux.

Los dispositivos se tratan como ficheros (/dev) hay dispositivos de bloque (b), y de caracteres (c). No poseen tamaño, solo posee 2 números:

*

El número de dispositivo mayor: indica el tipo de dispositivo. Indice para acceder al código del manejador del dispositivo.

*

El número de dispositivo menor: indica una unidad concreta dentro del tipo de dispositivo.

Utiliza:

*

Cache de buffer para los dispositivos de bloques. Utiliza un algoritmo LRU para las sustituciones. Para la planificación utiliza el algoritmo LOOK.

*

Listas-C para los dispositivos de caracteres.

#

Sistemas de ficheros.
#

Introducción.

Requisitos:

* Que almacene gran cantidad de información.
* La información debe sobrevivir a la terminación del proceso que la generó.
* Múltiples procesos puedan acceder a la información concurrentemente.

Parte SO encargada de la gestión de los ficheros: Sistema de ficheros o sistema de gestión de ficheros.

Funciones del sistema de gestión de ficheros: ...
#

Interfaz del sistema de ficheros.
#

Ficheros.

Un fichero tiene asociado unos atributos. El SO proporciona unas llamadas al sistema para poder realizar operaciones sobre esos ficheros: creación, lectura, escritura, búsqueda, borrado.

Un SO proporciona varios tipos de ficheros: normales, directorios, ficheros especiales (de dispositivos). En la estructura del fichero se debe distinguir entre estructura lógica u organización del fichero y estructura física o visión interna.

El método de acceso puede ser: secuencial, secuencial indexado, directo. #

Directorios.

Son ficheros con estructura especial y forma de acceso diferente a los ficheros normales. Su tarea principal es la de asociar el nombre de los ficheros a su localización en el almacenamiento secundario. Operaciones sobre directorios:

* Ver los atributos de los ficheros.
* Listar su contenido.
* Renombrar, copiar, mover, crear y borrar ficheros.

Estructura de los directorios:

* En árbol: los nombres de ficheros son únicos (absolutos o relativos). Un directorio puede contener tanto ficheros como otros directorios.
*

Grafo acíclico: Un directorio puede tener ficheros compartidos, un fichero puede estar en dos directorios diferentes. Puede tener dos nombres (caminos) diferentes. Problemas: cuando se recorre el árbol de directorio, pueden aparecer varias veces. En el borrado hay que tener en cuenta cuantas referencias hay de ese fichero.

*

Grafo general: Existe la posibilidad de que se formen ciclos.

#

Diseño del sistema de ficheros.
#

Tamaño del bloque.

Longitud fija o variable: normalmente es fija.

Tamaño del bloque con respecto al tamaño promedio del registro. Cuanto mayor sea el tamaño con respecto al registro, más registros se transmitirán en una operación de E/S.

En sistemas paginados conviene que coincida con el tamaño de la página o que sea un múltiplo de ella.

Agrupamiento de registros:

*

Agrupamiento fijo: Registros de longitud fija. En cada bloque se almacena un número determinado de registros. A ese número se le conoce como factor de bloque. Problemas: Fragmentación interna. No podemos tener registros de tamaño superior al bloque.

*

Agrupamiento de longitud variable extendido: Registros de longitud variable. Si un registro no cabe en el bloque se almacena parte de él en el siguiente bloque. Inconvenientes: Dificultad de implementación. Se pueden requerir 2 operaciones de E/S para leer un registro.

*

Agrupamiento de longitud variable no extendido: Como el anterior pero sin la expansión al otro bloque. Se puede dar la fragmentación interna y están limitados el tamaño del registro al del bloque.

#

Métodos de asignación de espacio a los ficheros.

*

Asignación contigua: a cada fichero se le asigna un conjunto contiguo de bloques en el momento de su creación. Información en el directorio: nombre, bloque de inicio y longitud (número de bloques).

Permite el acceso secuencial como el acceso directo. Podría aparecer fragmentación externa. Necesita una política de asignación de disco. Se necesita saber el tamaño máximo que va a tener. Son difíciles las actualizaciones de los ficheros.

*

Asignación enlazada o encadenada: Los bloques se asignan individualmente al fichero, de uno en uno cuando se necesitan. Se distinguen dos zonas: una de datos y otra que apunta al siguiente bloque. Los datos que se necesitan en el directorio son el nombre, el bloque de inicio y el fin. No posee fragmentación externa.

Permite acceso secuencial, directo (que sería más complejo que en el anterior). Se plantea el problema de la fiabilidad. Soluciones: listas doblemente enlazadas y almacenar en el bloque el nombre del fichero.

Información de la FAT: bloque disponible, si es el último bloque del fichero, siguiente bloque, bloque en mal estado.

*

Asignación indexada: No hay fragmentación externa y soporta bien el acceso directo. Cada fichero posee un bloque índice, donde se almacena las direcciones de los bloques donde está almacenado el fichero. En el directorio: nombre y bloque índice.

Los bloques índices se almacenan en memoria principal al leer el fichero. Problema: por cada fichero estamos gastando un bloque extra. ¿qué capacidad tiene el bloque índice? que pasa cuando no caben más datos en el bloque índice. Soluciones: esquema enlazado: la última entrada del bloque índice apunta al siguiente bloque índice. Indice multinivel: Un bloque apunta a los bloques índices de primer nivel, de esta a los de segundo nivel,... el último nivel almacena los datos. Esquema combinado: Algunas entradas son punteros directos a bloques de datos, una entrada indirecta a bloque de datos, otra entrada doble indirecta y una más triple indirecta.
#

Gestión del espacio libre.

Saber que bloques están libres y cuales ocupados. Para ello se emplea:

*

Mapa de bits: 1 bit por bloque. Necesita un espacio fijo que depende del tamaño del disco. Ejemplo: HDD: 20 Mb, bloque: 1 Kb, direcciones: 16 bits 20.000 bits. Es simple y eficiente de encontrar espacios libres y contiguos, pero tiene que estar en memoria principal.
*

Lista enlazada: un puntero apunta al primer bloque libre, este al 2º, etc. No necesita espacio extra, solo un puntero. No es muy eficiente para encontrar espacio contiguo, hay que ir recorriendo toda la lista.

Para evitar tener que ir a cada bloque, se usa:

*

Agrupamiento: (similar a la indexada) tenemos un bloque con direcciones de bloques libres. Se necesita mucho espacio. Ejemplo: HDD: 20 Mb, bloque: 1 Kb, direcciones: 16 bits 20 Mb/1 Kb = 20.000 bloques; 1 Kb/16 bits = 512 direcciones/bloque 20.000/512 = 40 bloques para la lista.

*

Recuento: cada elemento de la lista posee 3 componentes:

*

Dirección del bloque libre.

*

Número de bloques libres consecutivos.

*

Puntero al siguiente elemento.

#

Implementación de directorios.

Lista lineal de entradas. Para crear un fichero solo hay que añadir una entrada y para borrar, solo se elimina la entrada y se “juntan”. El problema es para buscar un fichero, si no está ordenado hay que recorrerla entera y si lo ordenamos se complican las inserciones y borrados. Una solución es utilizar una tabla de dispersión, para facilitar las búsquedas.

La primera vez que se accede a un directorio, esta información se guarda en la caché de disco.

Estructura interna de un directorio en MS-DOS.

En Linux:

Pasos para encontrar un fichero: /usr/ast/buzon

#

Fiabilidad del sistema de ficheros.

Dos aspectos a tener en cuenta:

*

Si la información se pierde, que se puede recuperar de forma fácil copias de seguridad: completas o incrementales.

*

Consistencia del sistema de ficheros

Un sistema de ficheros es consistente si la información que él mantiene es la real. El SF mantiene las estructuras en memoria. Se debe de chequear de vez en cuando:

Linux fsck.

1º Comprueba que cada bloque esté o libre u ocupado, para ello utiliza 2 contadores:

2º Verifica el sistema de directorios. Crea una tabla con los nodo-i y su contador y cuenta las referencias a cada nodo-i y luego compara esos contadores con el número de enlaces que hay en la información de cada nodo-i.
#

Rendimiento del sistema de ficheros.

*

Reducir el número de accesos al disco: Caché de buffer (caché de disco).

*

Reducir el desplazamiento de la cabeza del disco: planificación de las peticiones, agrupando los bloques que forman parte de un fichero.

#

El sistema de ficheros en Linux.

Se organiza en particiones. Puede reconocer otros SF gracias a VFS, capa intermedia entre el usuario y los SF.

Elementos:

*

superbloque: contiene información de todo el sistema de ficheros.

*

nodo-i: contiene información sobre el fichero.

*

enlace: dos tipos:

*

Duro.

*

Simbólico: rápido (máx. 60 car.) o normal.

*

fragmento: es una optimización del SF. Intenta optimizar los bloques grandes. El fragmento divide al bloque. Es la unidad mínima que se le asigna al fichero. Un mismo bloque puede tener fragmentos de diversos ficheros. Se puede direccionar individualmente.

*

EXT2: tamaño máximo de SF = 4Tb, tamaño máximo del fichero = 2 Gb, tamaño máximo del nombre de fichero = 256 caracteres, en algunos casos hasta 1.012. La estructura física está dividida en grupos de cilindros, con sector de arranque. Cada grupo contiene:

*

Superbloque.

*

Descriptores del SF.

*

Mapa de bits bloques.

*

Mapa de bits nodos-i.

*

Tabla de nodos-i.

*

Bloques de datos.

#

Introducción a los sistemas distribuidos.

#

Introducción

#

Definición de sistema distribuido.

Un sistema distribuido es un conjunto de máquinas independientes, que para el usuario se comporta como un único ordenador.

Clasificación de los sistemas multiprocesadores:

* Hardware fuertemente o estrechamente acoplado: los multiprocesadores comparten memoria.
* Software estrechamente o fuertemente acoplado: las estructuras de control que posee tienen información global del sistema.

#

Sistemas paralelos o de tiempo compartido.

Hardware estrechamente acoplado y software estrechamente acoplado. Las CPU se comunican por la memora. Utilizan:

* Multiprocesamiento simétrico: todas las CPU ejecutan una misma copia del sistema operativo.
* Multiprocesamiento asimétrico: cada procesador tiene una tarea específica u uno maestro que reparte las tareas.

#

Sistemas operativos en red.

Hardware y software poco acoplados. Compuesto por n máquinas independientes conectadas por una red. Cada una posee su propio sistema operativo y hardware. El objetivo principal es la interoperatividad: flexibilidad o capacidad para intercambiar información entre sistemas heterogéneos.
#

Verdadero sistema distribuido.

Hardware poco acoplado y software estrechamente acoplado. La característica principal es la transparencia. Todos poseen el mismo sistema operativo. El usuario ve todas las máquinas como si fueran una única máquina. Características:

* Proporcionan un mecanismo global de comunicación entre procesos, ya sea dentro de la propia máquina o en una máquina remota.
* Proporcionan un control local de los recursos. Cada sistema operativo es el encargado de gestionar sus recursos locales.
* Los núcleos de los sistemas operativos que se ejecutan en las máquinas son idénticos.
* Existe un esquema global de protección.

#

Comparativas entre los diferentes sistemas.

Ventajas:

*

Paralelo monoprocesador:

*

Posibilidad de aumentar la velocidad de ejecución de un proceso.

*

Posibilidad de aumentar la concurrencia, solapándose la ejecución de distintos procesos.

*

Red centralizado:

*

Comunicación entre usuarios de distintas máquinas.

*

Compartición de recursos.

*

Mayor fiabilidad

*

Distribuido red:

*

La compartición de recursos se produce de forma transparente.

*

Mayor fiabilidad.

#

Transparencia de los sistemas distribuidos.

*

Transparencia de localización: Dado el nombre del recurso, los usuarios no son capaces de indicar la localización del recurso.

*

Transparencia de migración: los recursos se van a poder cambiar de un lugar a otro sin cambiarle el nombre.

*

Transparencia de réplica: el sistema es capaz de hacer copias de los ficheros cuando sean necesarios y el usuario no sabe ni cuantas hay ni a cual está accediendo.

*

Transparencia de concurrencia: los usuarios no notan que otros usuarios también están accediendo a los recursos.

*

Transparencia de paralelismo: las actividades pueden ejecutarse en paralelo pero el usuario no se da cuenta.

A parte de estos objetivos también poseen:

*

Flexibilidad: facilidad de adaptarse a los cambios.

*

Fiabilidad.

*

Rendimiento.

*

Escalabilidad.

#

Modelo cliente-servidor.

Es una forma de describir la interacción entre procesos de un sistema distribuido. Los procesos o bien ofrecen servicios (servidores) o bien piden servicios (clientes).

1

Caché

Cinta magnética

Disco magnético

Estructura de los SO

Núcleo complejo

Núcleo mínimo

Monolítico

Modular

Capas

Estructurado

Micronúcleo

Máquinas virtuales

Orientado a objetos

Capas de abstracción

Capas funcionales

Hardware

Interfaz de núcleo con el hardware

Sistema de ficheros

Señales

Sistema de E/S

Manejo de memoria

Planificación de la CPU

Intercambio

Interfaz de llamadas al sistema con el núcleo

Bibliotecas del sistema

Compiladores e intérpretes

Intérprete de órdenes

Interfaz con el usuario

Programas de usuario

Programas del sistema

Núcleo

UNIX

Programas de aplicación

Programas residentes del sistema

Manejadores de dispositivos de MSDOS

Manejadores de dispositivos en ROM BIOS

MSDOS

Capa N

Capa N-1

Operaciones existentes

Operaciones ocultas

Operaciones nuevas

Clasificación de los SO

Número de usuarios

Tipo de interacción

Monousuarios

Multiusuarios

Número de programas

Por lotes

Interactivos

Monoprogramados

Multiprogramados

Tiempo compartido

No Ejecución

Ejecución

Entrada

Pasar a Ejecución

Terminación

Parada

Listo

Admisión

Pasar a Ejecución

Salida

Tiempo excedido

Nuevo

Ejecución

Terminado

Bloqueado

Despertar

Bloqueo

Listo

Admisión

Pasar a Ejecución

Salida

Tiempo excedido

Nuevo

Ejecución

Terminado

Bloqueado

Despertar

Bloqueo

Suspendido

listo

Suspendido

bloqueado

Despertar

Admisión

Suspender

Suspender

Reanudar

Reanudar

Mediante memoria compartida

R1

P1

R1

P1

R1

P1

R1

P1

R2

R3

R4

P2

P3

P1

P2

P3

P4

R1

R2

Memoria

Principal

Caché de disco

Registros

Disco óptico

Volátil

Interna

* Disminuye el coste por bit.
* Aumenta la capacidad.

*

Aumenta el tiempo de acceso.

*

Disminuye la frecuencia de accesos por parte del procesador.

Mayor capacidad, menos costosos.

Menor tiempo de acceso, más costosos.

Esquemas de gestión de memoria.

Asignación contigua

Asignación no contigua

Máquina desnuda

Monoprogramación

Particiones

Sistema compañero

Variables

EXAMEN

1.- ¿Qué es un sistema operativo?
R=Es un programa que controla la ejecución de los programas de aplicación. Actúa como interfaz entre el usuario y el hardware. Proporciona al usuario un entorno cómodo y eficiente para ejecutar sus programas.

2.- .- ¿Qué es la información?
R= Es un conjunto de datos organizados de tal modo que adquieren un valor adicional más allá del propio.

El tipo de información creada depende de las relaciones definidas entre los datos existentes Adicionar datos nuevos o diferentes significa la posibilidad de redefinir las relaciones y de crear nueva información.

3.- ¿Qué es un sistema?
R= Es un conjunto de elementos y componentes que interactúan entre sí para cumplir un determinado objetivo.

4.- ¿Qué es el modelado de un sistema?
R= Es una abstracción o aproximación que sirve para representar la realidad además nos permiten examinar situaciones reales y obtener una mejor comprensión del sistema.

5.- ¿Qué es un sistema de información?
R= Es un conjunto de elementos o componentes interrelacionados para recolectar (entrada), manipular (proceso) y diseminar (salida) datos e información y proveer un mecanismo de retroalimentación en pro del cumplimiento de un objetivo.

6.- ¿Qué requieren los sistemas multiprogramados?
R= Planificación CPU, Planificación de los dispositivos. Control de la concurrencia, Control de la memoria, Proteccion

7.- ¿Qué es la salida de información?
R= es la capacidad de un Sistema de Información para sacar la información procesada o bien datos de entrada al exterior

8.-¿Qué son los Sistemas de tiempo compartido.?
R=Son sistemas multiprogramados, multiusuarios e interactivos.

9.- ¿Qué es Procesamiento de Información?
R= Es la capacidad del Sistema de Información para efectuar cálculos de acuerdo con una secuencia de operaciones preestablecida

10.-¿Deque se compone el núcleo de un sistema operativo Linux?
R= Administrador de procesos, Administrador de memoria, Administrador de dispositivos de red, Administrador de sistemas de ficheros

NOTA: LAS RESPUESTAS ESTAN MARCADAS EN **LETRAS NEGRITAS**

ANEXAR ESTA HOJA EN LA PARTE POSTERIOR DE SU TRABAJO. LISTA PARA REVISAR POR SU PROPIA CUENTA EL VALOR DEL DOCUMENTO
Antes de presentar su documento, por favor utilice esta página para determinar si su trabajo cumple con lo establecido por AIU.

Si hay más que 2 elementos que no puede verificar adentro de su documento, entonces, por favor, haga las correcciones necesarias para ganar los créditos correspondientes.

___√__ Yo tengo una página de cobertura similar al ejemplo de la página 89 o 90 del Suplemento.
__√___ Yo incluí una tabla de contenidos con la página correspondiente para cada componente.
_____ Yo incluí un abstracto del documento (exclusivamente para la Tesis).
___√__ Yo seguí el contorno propuesto en la página 91 o 97 del Suplemento con todos los títulos o casi.
__√___ Yo usé referencias a través de todo el documento según el requisito de la página 92 del Suplemento.
___√__ Mis referencias están en orden alfabético al final según el requisito de la página 92 del Suplemento.
__√___ Cada referencia que mencioné en el texto se encuentra en mi lista o viceversa.
__√___ Yo utilicé una ilustración clara y con detalles para defender mi punto de vista.
___√__ Yo utilicé al final apéndices con gráficas y otros tipos de documentos de soporte.
__√___ Yo utilicé varias tablas y estadísticas para aclarar mis ideas más científicamente.
_____ Yo tengo por lo menos 50 páginas de texto (15 en ciertos casos) salvo si me pidieron lo contrario.
__√___ Cada sección de mi documento sigue una cierta lógica (1,2,3…) _____ Yo no utilicé caracteres extravagantes, dibujos o decoraciones.
__√___ Yo utilicé un lenguaje sencillo, claro y accesible para todos.
___√__ Yo utilicé Microsoft Word (u otro programa similar) para chequear y eliminar errores de ortografía.
___√__ Yo utilicé Microsoft Word / u otro programa similar) para chequear y eliminar errores de gramática.
__√___ Yo no violé ninguna ley de propiedad literaria al copiar materiales que pertenecen a otra gente.
__√___ Yo afirmo por este medio que lo que estoy sometiendo es totalmente mi obra propia.

Gabriel Alejandro Moran Valdes___19/09/2007_____ Firma del Estudiante Fecha Muchos de estos puntos están cubiertos. Pero requerimos que ponga atención en otros.