UDS

Mi Universidad

LIBRO

Nombre de la materia: SISTEMAS OPERATIVOS DE RED

Nombre de la Licenciatura: INGENIERIA EN SISTEMAS COMPUTACIONALES

Cuatrimestre: 5to.

Periódo

Enero-Abril

Marco Estratégico de Referencia

Antecedentes históricos

Nuestra Universidad tiene sus antecedentes de formación en el año de 1979 con el inicio de actividades de la normal de educadoras "Edgar Robledo Santiago", que en su momento marcó un nuevo rumbo para la educación de Comitán y del estado de Chiapas. Nuestra escuela fue fundada por el Profesor Manuel Albores Salazar con la idea de traer educación a Comitán, ya que esto representaba una forma de apoyar a muchas familias de la región para que siguieran estudiando.

En el año 1984 inicia actividades el CBTiS Moctezuma Ilhuicamina, que fue el primer bachillerato tecnológico particular del estado de Chiapas, manteniendo con esto la visión en grande de traer educación a nuestro municipio, esta institución fue creada para que la gente que trabajaba por la mañana tuviera la opción de estudiar por las tardes.

La Maestra Martha Ruth Alcázar Mellanes es la madre de los tres integrantes de la familia Albores Alcázar que se fueron integrando poco a poco a la escuela formada por su padre, el Profesor Manuel Albores Salazar; Víctor Manuel Albores Alcázar en julio de 1996 como chofer de transporte escolar, Karla Fabiola Albores Alcázar se integró en la docencia en 1998, Martha Patricia Albores Alcázar en el departamento de cobranza en 1999.

En el año 2002, Víctor Manuel Albores Alcázar formó el Grupo Educativo Albores Alcázar S.C. para darle un nuevo rumbo y sentido empresarial al negocio familiar y en el año 2004 funda la Universidad Del Sureste.

La formación de nuestra Universidad se da principalmente porque en Comitán y en toda la región no existía una verdadera oferta Educativa, por lo que se veía urgente la creación de una institución de Educación superior, pero que estuviera a la altura de las exigencias de los

jóvenes que tenían intención de seguir estudiando o de los profesionistas para seguir preparándose a través de estudios de posgrado.

Nuestra Universidad inició sus actividades el 18 de agosto del 2004 en las instalaciones de la 4ª avenida oriente sur no. 24, con la licenciatura en Puericultura, contando con dos grupos de cuarenta alumnos cada uno. En el año 2005 nos trasladamos a nuestras propias instalaciones en la carretera Comitán – Tzimol km. 57 donde actualmente se encuentra el campus Comitán y el corporativo UDS, este último, es el encargado de estandarizar y controlar todos los procesos operativos y educativos de los diferentes campus, así como de crear los diferentes planes estratégicos de expansión de la marca.

Misión

Satisfacer la necesidad de Educación que promueva el espíritu emprendedor, aplicando altos estándares de calidad académica, que propicien el desarrollo de nuestros alumnos, Profesores, colaboradores y la sociedad, a través de la incorporación de tecnologías en el proceso de enseñanza-aprendizaje.

Visión

Ser la mejor oferta académica en cada región de influencia, y a través de nuestra plataforma virtual tener una cobertura global, con un crecimiento sostenible y las ofertas académicas innovadoras con pertinencia para la sociedad.

Valores

- Disciplina
- Honestidad
- Equidad
- Libertad

Escudo

El escudo del Grupo Educativo Albores Alcázar S.C. está constituido por tres líneas curvas que nacen de izquierda a derecha formando los escalones al éxito. En la parte superior está situado un cuadro motivo de la abstracción de la forma de un libro abierto.

Eslogan

"Mi Universidad"

ALBORES

Es nuestra mascota, un Jaguar. Su piel es negra y se distingue por ser líder, trabaja en equipo y obtiene lo que desea. El ímpetu, extremo valor y fortaleza son los rasgos que distinguen.

Intervención Psicopedagógica.

Objetivo de la materia:

El alumno conocerá y diferenciará las técnicas de asignación de tareas al procesador de administración de procesos, así como el manejo de interrupciones, entradas/salidas y las diferentes técnicas de asignación de memoria.

Criterios de evaluación:

No	Concepto	Porcentaje
1	Trabajos Escritos	30%
2	Actividades áulicas	20%
3	Examen	50%
4	Total	100%
5	Escala de calificación	7- 10
6	Mínima aprobatoria	7

INDICE

UNIDAD IV

CONTROL DE PROCESOS Y RECURSOS

UNIDAD I

INTRODUCCIÓN A LOS SISTEMAS OPERATIVOS

Un sistema operativo es un programa que actúa como intermediario entre el usuario de una computadora y el hardware de este. El propósito del sistema operativo es crear un entorno en el que un usuario pueda ejecutar programas. Así pues, el objetivo primario de un sistema operativo es hacer el sistema de la computadora cómodo de usar. Un objetivo secundario es utilizar el hardware de la computadora de forma eficiente.

1.1.- Definición y concepto de sistema operativo.

Un sistema operativo es un conjunto de programas que permite manejar la memoria, disco, medios de almacenamiento de información y los diferentes periféricos o recursos de nuestra computadora, como son el teclado, el mouse, la impresora, la placa de red, entre otros.

Los periféricos utilizan un driver o controlador y son desarrollados por los fabricantes de cada equipo. Encontramos diferentes sistemas operativos como Windows, Linux, MAS OS, en sus diferentes versiones. También los teléfonos y tablets poseen un sistema operativo.

Dentro de las tareas que realiza el sistema operativo, en particular, se ocupa de gestionar la memoria de nuestro sistema y la carga de los diferentes programas, para ello cada programa tiene una prioridad o jerarquía y en función de la misma contará con los recursos de nuestro sistema por más tiempo que un programa de menor prioridad.

El sistema operativo se ocupa también de correr procesos. Llamamos proceso a la carga en memoria de nuestro programa, si no está cargado en memoria nuestro programa simplemente “no corre”.

Existen diversas definiciones de lo que es un Sistema Operativo, pero no hay una definición exacta, es decir una que sea estándar; a continuación, se presentan algunas:

1.- Se pueden imaginar un Sistema Operativo como los programas, instalados en el software o firmware, que hacen utilizable el hardware. El hardware proporciona la "capacidad bruta de cómputo"; los sistemas operativos ponen

dicha capacidad de cómputo al alcance de los usuarios y administran cuidadosamente el hardware para lograr un buen rendimiento.

2.- Los Sistemas Operativos son ante todo administradores de recursos; el principal recurso que administran es el hardware del computador; además de los procesadores, los medios de almacenamiento, los dispositivos de entrada/salida, los dispositivos de comunicación y los datos.

3.- Un Sistema Operativo es un programa que actúa como intermediario entre el usuario y el hardware del computador y su propósito es proporcionar el entorno en el cual el usuario pueda ejecutar programas. Entonces, el objetivo principal de un Sistema Operativo es, lograr que el sistema de computación se use de manera cómoda, y el objetivo secundario es que el hardware del computador se emplee de manera eficiente. 4.- Un Sistema Operativo es un conjunto de programas que controla la ejecución de programas de aplicación y actúa como una interfaz entre el usuario y el hardware de una computadora, esto es, un Sistema Operativo explota y administra los recursos de hardware de la computadora con el objeto de proporcionar un conjunto de servicios a los usuarios del sistema.

En resumen, se podría decir que los Sistemas Operativos son un conjunto de programas que crean la interfaz del hardware con el usuario, y que tiene dos funciones primordiales, que son:

Gestionar el hardware. - Se refiere al hecho de administrar de una forma más eficiente los recursos de la máquina.

Facilitar el trabajo al usuario. -Permite una comunicación con los dispositivos de la máquina.

El Sistema Operativo se encuentra almacenado en la memoria secundaria. Primero se carga y ejecuta un pedazo de código que se encuentra en el procesador, el cual carga el BIOS, y este a su vez carga el Sistema Operativo que carga todos los programas de aplicación y software variado.

1.2.- Características de los sistemas operativos.

En general, se puede decir que un Sistema Operativo tiene las siguientes características:

- Conveniencia. Un Sistema Operativo hace más conveniente el uso de una computadora.
- Eficiencia. Un Sistema Operativo permite que los recursos de la computadora se usen de la manera más eficiente posible.
- Habilidad para evolucionar. Un Sistema Operativo deberá construirse de manera que permita el desarrollo, prueba o introducción efectiva de nuevas funciones del sistema sin interferir con el servicio.
- Encargado de administrar el hardware. El Sistema Operativo se encarga de manejar de una mejor manera los recursos de la computadora en cuanto a hardware se refiere, esto es, asignar a cada proceso una parte del procesador para poder compartir los recursos.
- Relacionar dispositivos (gestionar a través del kernel). El Sistema Operativo se debe encargar de comunicar a los dispositivos periféricos, cuando el usuario así lo requiera.
- Organizar datos para acceso rápido y seguro.
- Manejar las comunicaciones en red. El Sistema Operativo permite al usuario manejar con alta facilidad todo lo referente a la instalación y uso de las redes de computadoras.
- Procesamiento por bytes de flujo a través del bus de datos.
- Facilitar las entradas y salidas. Un Sistema Operativo debe hacerle fácil al usuario el acceso y manejo de los dispositivos de Entrada/Salida de la computadora.
- Técnicas de recuperación de errores.
- Evita que otros usuarios interfieran. El Sistema Operativo evita que los usuarios se bloqueen entre ellos, informándoles si esa aplicación esta siendo ocupada por otro usuario.
- Generación de estadísticas.
- Permite que se puedan compartir el hardware y los datos entre los usuarios.

El software de aplicación son programas que se utilizan para diseñar, tal como el procesador de palabras, lenguajes de programación, hojas de cálculo, etc.

El software de base sirve para interactuar el usuario con la máquina, son un conjunto de programas que facilitan la ambiente plataforma, y permite el diseño del mismo.

El Software de base está compuesto por:

- Cargadores.
- Compiladores.
- Ensambladores.
- Macros.

1.3.- Clasificación de los sistemas operativos.

Con el paso del tiempo, los Sistemas Operativos fueron clasificándose de diferentes maneras, dependiendo del uso o de la aplicación que se les daba. A continuación, se mostrarán diversos tipos de Sistemas Operativos que existen en la actualidad, con algunas de sus características:

Sistemas Operativos por lotes.

Los Sistemas Operativos por lotes, procesan una gran cantidad de trabajos con poca o ninguna interacción entre los usuarios y los programas en ejecución. Se reúnen todos los trabajos comunes para realizarlos al mismo tiempo, evitando la espera de dos o más trabajos como sucede en el procesamiento en serie. Estos sistemas son de los más tradicionales y antiguos, y fueron introducidos alrededor de 1956 para aumentar la capacidad de procesamiento de los programas.

Cuando estos sistemas son bien planeados, pueden tener un tiempo de ejecución muy alto, porque el procesador es mejor utilizado y los Sistemas Operativos pueden ser simples, debido a la secuenciabilidad de la ejecución de los trabajos.

Algunos ejemplos de Sistemas Operativos por lotes exitosos son el SCOPE, del DC6600, el cual está orientado a procesamiento científico pesado, y el EXEC II para el UNIVAC 1107, orientado a procesamiento académico.

Algunas otras características con que cuentan los Sistemas Operativos por lotes son:

- Requiere que el programa, datos y órdenes al sistema sean remitidos todos juntos en forma de lote.
- Permiten poca o ninguna interacción usuario/programa en ejecución.

- Mayor potencial de utilización de recursos que procesamiento serial simple en sistemas multiusuarios.
- No conveniente para desarrollo de programas por bajo tiempo de retorno y depuración fuera de línea.
- Conveniente para programas de largos tiempos de ejecución (ej, análisis estadísticos, nóminas de personal, etc.)
- Se encuentra en muchos computadores personales combinados con procesamiento serial.
- Planificación del procesador sencilla, típicamente procesados en orden de llegada.
- Planificación de memoria sencilla, generalmente se divide en dos: parte residente del S.O. y programas transitorios.
- No requieren gestión crítica de dispositivos en el tiempo.
- Suelen proporcionar gestión sencilla de manejo de archivos: se requiere poca protección y ningún control de concurrencia para el acceso.

Sistemas Operativos de tiempo real.

Los Sistemas Operativos de tiempo real son aquellos en los cuales no tiene importancia el usuario, sino los procesos. Por lo general, están subutilizados sus recursos con la finalidad de prestar atención a los procesos en el momento que lo requieran. se utilizan en entornos donde son procesados un gran número de sucesos o eventos.

Muchos Sistemas Operativos de tiempo real son construidos para aplicaciones muy específicas como control de tráfico aéreo, bolsas de valores, control de refinerías, control de laminadores. También en el ramo automovilístico y de la electrónica de consumo, las aplicaciones de tiempo real están creciendo muy rápidamente. Otros campos de aplicación de los Sistemas Operativos de tiempo real son los siguientes:

- Control de trenes.
- Telecomunicaciones.
- Sistemas de fabricación integrada.
- Producción y distribución de energía eléctrica.
- Control de edificios.
- Sistemas multimedia.

Algunos ejemplos de Sistemas Operativos de tiempo real son: VxWorks, Solaris, Lyns OS y Spectra. Los Sistemas Operativos de tiempo real, cuentan con las siguientes características:

- Se dan en entornos en donde deben ser aceptados y procesados gran cantidad de sucesos, la mayoría externos al sistema computacional, en breve tiempo o dentro de ciertos plazos.
- Se utilizan en control industrial, conmutación telefónica, control de vuelo, simulaciones en tiempo real., aplicaciones militares, etc.
- Objetivo es proporcionar rápidos tiempos de respuesta.
- Procesa ráfagas de miles de interrupciones por segundo sin perder un solo suceso.
- Proceso se activa tras ocurrencia de suceso, mediante interrupción.
- Proceso de mayor prioridad expropia recursos.
- Por tanto, generalmente se utiliza planificación expropiativa basada en prioridades.
- Gestión de memoria menos exigente que tiempo compartido, usualmente procesos son residentes permanentes en memoria.
- Población de procesos estática en gran medida.
- Poco movimiento de programas entre almacenamiento secundario y memoria.
- Gestión de archivos se orienta más a velocidad de acceso que a utilización eficiente del recurso.

Sistemas Operativos de multiprogramación (o Sistemas Operativos de multitarea).

Se distinguen por sus habilidades para poder soportar la ejecución de dos o más trabajos activos (que se están ejecutado) al mismo tiempo. Esto trae como resultado que la Unidad Central de Procesamiento (UCP) siempre tenga alguna tarea que ejecutar, aprovechando al máximo su utilización.

Su objetivo es tener a varias tareas en la memoria principal, de manera que cada uno está usando el procesador, o un procesador distinto, es decir, involucra máquinas con más de una UCP.

Sistemas Operativos como UNIX, Windows 95, Windows 98, Windows NT, MAC-OS, OS/2, soportan la multitarea.

Las características de un Sistema Operativo de multiprogramación o multitarea son las siguientes:

- Mejora productividad del sistema y utilización de recursos.
- Multiplexa recursos entre varios programas.
- Generalmente soportan múltiples usuarios (multiusuarios).
- Proporcionan facilidades para mantener el entorno de usuarios individuales.
- Requieren validación de usuario para seguridad y protección.
- Proporcionan contabilidad del uso de los recursos por parte de los usuarios.
- Multitarea sin soporte multiusuario se encuentra en algunos computadores personales o en sistemas de tiempo real.
- Sistemas multiprocesadores son sistemas multitareas por definición ya que soportan la ejecución simultánea de múltiples tareas sobre diferentes procesadores.
- En general, los sistemas de multiprogramación se caracterizan por tener múltiples programas activos compitiendo por los recursos del sistema: procesador, memoria, dispositivos periféricos.

Sistemas Operativos de tiempo compartido.

Permiten la simulación de que el sistema y sus recursos son todos para cada usuario. El usuario hace una petición a la computadora, esta la procesa tan pronto como le es posible, y la respuesta aparecerá en la terminal del usuario.

Los principales recursos del sistema, el procesador, la memoria, dispositivos de E/S, son continuamente utilizados entre los diversos usuarios, dando a cada usuario la ilusión de que tiene el sistema dedicado para sí mismo. Esto trae como consecuencia una gran carga de trabajo al Sistema Operativo, principalmente en la administración de memoria principal y secundaria.

Ejemplos de Sistemas Operativos de tiempo compartido son Multics, OS/360 y DEC-10.

Características de los Sistemas Operativos de tiempo compartido:

- Populares representantes de sistemas multiprogramados multiusuario, ej: sistemas de diseño asistido por computador, procesamiento de texto, etc.
- Dan la ilusión de que cada usuario tiene una máquina para sí.
- Mayoría utilizan algoritmo de reparto circular.
- Programas se ejecutan con prioridad rotatoria que se incrementa con la espera y disminuye después de concedido el servicio.
- Evitan monopolización del sistema asignando tiempos de procesador (time slot).
- Gestión de memoria proporciona protección a programas residentes.
- Gestión de archivo debe proporcionar protección y control de acceso debido a que pueden existir múltiples usuarios accediendo un mismo archivo.

Sistemas Operativos distribuidos.

Permiten distribuir trabajos, tareas o procesos, entre un conjunto de procesadores. Puede ser que este conjunto de procesadores esté en un equipo o en diferentes, en este caso es trasparente para el usuario. Existen dos esquemas básicos de éstos. Un sistema fuertemente acoplado esa es aquel que comparte la memoria y un reloj global, cuyos tiempos de acceso son similares para todos los procesadores. En un sistema débilmente acoplado los procesadores no comparten ni memoria ni reloj, ya que cada uno cuenta con su memoria local.

Los sistemas distribuidos deben de ser muy confiables, ya que si un componente del sistema se compone otro componente debe de ser capaz de reemplazarlo.

Entre los diferentes Sistemas Operativos distribuidos que existen tenemos los siguientes: Sprite, Solaris-MC, Mach, Chorus, Spring, Amoeba, Taos, etc.

Características de los Sistemas Operativos distribuidos:

- Colección de sistemas autónomos capaces de comunicación y cooperación mediante interconexiones hardware y software.
- Gobierna operación de un S.C. y proporciona abstracción de máquina virtual a los usuarios.
- Objetivo clave es la transparencia.

- Generalmente proporcionan medios para la compartición global de recursos.
- Servicios añadidos: denominación global, sistemas de archivos distribuidos, facilidades para distribución de cálculos (a través de comunicación de procesos internodos, llamadas a procedimientos remotos, etc.).

Sistemas Operativos de red.

Son aquellos sistemas que mantienen a dos o más computadoras unidas a través de algún medio de comunicación (físico o no), con el objetivo primordial de poder compartir los diferentes recursos y la información del sistema.

El primer Sistema Operativo de red estaba enfocado a equipos con un procesador Motorola 68000, pasando posteriormente a procesadores Intel como Novell Netware.

Los Sistemas Operativos de red más ampliamente usados son: Novell Netware, Personal Netware, LAN Manager, Windows NT Server, UNIX, LANtastic.

1.4.- Historia y desarrollo de los sistemas operativos.

La informática tal y como se le conoce hoy día, surgió a raíz de la II Guerra Mundial, en la década de los 40. En esos años no existía siquiera el concepto de "Sistema Operativo" y los programadores interactuaban directamente con el hardware de las computadoras trabajando en lenguaje máquina (esto es, en binario, programando únicamente con 0s y 1s).

El concepto de Sistema Operativo surge en la década de los 50. El primer Sistema Operativo de la historia fue creado en 1956 para un ordenador IBM 704, y básicamente lo único que hacía era comenzar la ejecución de un programa cuando el anterior terminaba.

En los años 60 se produce una revolución en el campo de los Sistemas Operativos. Aparecen conceptos como sistema multitarea, sistema multiusuario, sistema multiprocesadores y sistema en tiempo real.

Es en esta década cuando aparece UNIX, la base de la gran mayoría de los Sistemas Operativos que existen hoy en día.

En los años 70 se produce un boom en cuestión de ordenadores personales, acercando estos al público general de manera impensable hasta entonces. Esto hace que se multiplique el desarrollo, creándose el lenguaje de programación C (diseñado específicamente para reescribir por completo el código UNIX).

Como consecuencia de este crecimiento exponencial de usuarios, la gran mayoría de ellos sin ningún conocimiento sobre lenguajes de bajo o alto nivel, hizo que en los años 80, la prioridad a la hora de diseñar un sistema operativo fuese la facilidad de uso, surgiendo así las primeras interfaces de usuario. En los 80 nacieron sistemas como MacOS, MS-DOS, Windows.

En la década de los 90 hace su aparición Linux, publicándose la primera versión del núcleo en septiembre de 1991, que posteriormente se uniría al proyecto GNU, un sistema operativo completamente libre, similar a UNIX, al que le faltaba para funcionar un núcleo funcional. Hoy en día la mayoría de la gente conoce por Linux al Sistema Operativo que realmente se llama GNU/Linux.

1.5.- Estructura del sistema.

Definir una arquitectura para el Sistema Operativo; esta estará influida en alguna medida por el hardware que manejará. Sin embargo, es posible identificar algunos componentes comunes como el núcleo y las llamadas al sistema. El núcleo contiene, básicamente, todo el código e información necesaria para la gestión de procesos y memoria y en la mayoría de los sistemas operativos también se incluye buena parte de la gestión de entrada/salida. Según la arquitectura que tengamos así es como el usuario se comunica con el sistema operativo y el sistema operativo o el núcleo con los componentes del hardware, por tanto, cada sistema operativo tiene una arquitectura específica en dependencia de las necesidades de este.

Sistemas monolíticos

En los sistemas monolíticos, todos los componentes de gestión y programas del sistema están escritos en un solo código o espacio lógico. Pueden decirse que el núcleo es el Sistema Operativo. La división más evidente que puede hacerse es entre procesos de aplicación o usuario y procesos del sistema.

Los sistemas monolíticos son los más comunes puesto que su implementación y diseño son los menos complejos. La desventaja es que como todo el sistema se

ejecuta en el mismo nivel de privilegio que el núcleo (el Sistema Operativo es el núcleo) es muy probable que haya problemas (el sistema se apague, se bloquee o se cuelguen procesos, por ejemplo) si ocurre algún fallo del hardware o existe algún error sin depurara en el código del sistema. GNU/Linux es monolítico, aunque con éste se introdujo una variante interesante. A pesar de ser monolítico, es modular, lo que significa que es posible quitar o añadir componentes al núcleo incluso en caliente (o sea, en pleno funcionamiento). Así, si un módulo tiene problemas se puede reemplazar, arreglar o eliminar sin afectar al resto de funcionalidades. Otra ventaja interesante de los sistemas monolíticos es su velocidad. Ya que todos los componentes del Sistema Operativo comparten los privilegios y direcciones y la separación funcional solo se hace entre procesos del sistema y los de aplicación, la demora para ejecutar las llamadas al sistema es mínima.

Sistemas por capas o niveles de privilegio

En la medida que el hardware se desarrolló para incorporar mecanismos de protección para la gestión de procesos, memoria y entrada/salida, los Sistemas Operativos se adaptaron a este diseño. El objetivo de la arquitectura por capas o niveles de privilegio, es separar la acción del código de los procesos del sistema, del núcleo y de los procesos de usuario. Idealmente se separa en el nivel de máximo privilegios o protección, al código base del núcleo.

En el siguiente nivel de privilegios se ubica a los procesos del sistema para la gestión de procesos y memoria; en otro nivel a los procesos de gestión de entrada salida incluyendo los drivers y en el nivel de menos privilegio, se ejecutan el resto de los procesos de aplicación. Esto por supuesto, no siempre es así y pueden existir muchísimas variantes en la implementación. Una práctica común es ubicar los procesos y componentes de entrada/salida en los dos primeros niveles, para acelerar su ejecución. Lo relevante en esta arquitectura es que se necesitan realizar restricciones en cuanto a las llamadas al sistema que puedan ejecutar los procesos de determinados niveles. Por ejemplo, se ha de evitar que un proceso pueda efectuar una llamada al sistema que requiera la atención de un componente de menor privilegio o tal vez evitar que los programas de aplicación hagan llamadas directamente sobre el núcleo. Así puede obligarse a utilizar como mediadores a otros niveles.

Sistemas de Micronúcleo

La posibilidad de separar funcionalmente los programas del sistema de los programas de aplicación y asegurar protección adicional con el hardware, origina otra arquitectura, la de micronúcleo. En esta, se trata de combinar el rendimiento y sencillez de la arquitectura monolítica con la protección y organización de la arquitectura por capas. La idea fundamental es obtener un núcleo lo más pequeño y rápido posible y tratar el resto de las funciones y componentes como procesos de aplicación. En esta nueva concepción, es usual que el núcleo solo contenga lo necesario para la gestión de memoria y procesos.

El resto se ejecutan como aplicaciones de usuario; es decir, con el nivel mínimo de privilegios. En la práctica es un poco difícil conseguir esto sin una pérdida apreciable de rendimiento. En alguna medida deben incluirse en el núcleo otras funciones como el manejo de hardware y algunos drivers.

Sistemas cliente-servidor

Separar funcionalmente el núcleo, los procesos del sistema y los procesos de aplicación, induce una idea interesante: ¿Se podrán separar físicamente los procesos del sistema y/o los de aplicación? La arquitectura dónde cada proceso se ejecuta de manera independiente, es una de las más difíciles de lograr. Esta independencia implica que los procesos podrían ser ejecutados en sistemas (no solo CPU, sino todo el hardware) diferentes y distantes geográficamente. En cada sistema independiente solo estarían presentes el núcleo y los componentes mínimos para la ejecución de uno o algunos procesos. En esta arquitectura, se manifiesta con mayor peso la necesidad de establecer llamadas al sistema robustas, para la comunicación entre procesos. Se escoge el modelo cliente --servidor para esta comunicación, porque se establece que cada proceso (independiente o no) actúa como servidor del resto. En esta arquitectura, el objetivo fundamental del núcleo es garantizar la comunicación entre procesos. Esta filosofía propició la creación de los Sistemas Operativos distribuidos que son, básicamente, implementaciones basadas en el modelo cliente-servidor. Aunque los Sistemas Operativos distribuidos no serán estudiados en este curso, explotan un mecanismo interesante de la comunicación entre procesos, que es el paso de mensajes. La tendencia actual es desarrollar aplicaciones distribuidas en vez de Sistemas Operativos distribuidos, aprovechando las redes de computadoras.

1.5.1.- Metodologías de diseño.

El diseño de sistemas es el proceso de definición de la arquitectura, módulos, interfaces y datos de un sistema para satisfacer unos requisitos previamente especificados. El diseño de sistemas podría verse como la aplicación de teoría de sistemas al desarrollo de un nuevo producto. Existe cierta superposición con las disciplinas de análisis de sistemas, arquitectura de sistemas e ingeniería de sistemas.

Si el tema más amplio del desarrollo de un producto "combina la perspectiva del marketing, el diseño y la fabricación en un único enfoque para el desarrollo de productos," I entonces el diseño es el acto de usar la información del marketing y crear el diseño del producto para ser posteriormente fabricado. El diseño de sistemas es, por tanto, el proceso de definir y desarrollar sistemas para satisfacer requisitos especificados por el usuario.

Hasta los años 90, el diseño de sistemas tuvo una función crucial y respetada en la industria de procesamiento de datos. En los 90, la normalización del hardware y el software resultó en la capacidad de construir sistemas modulares. La creciente importancia del software que se ejecuta en plataformas genéricas ha realzado la disciplina de ingeniería de software.

Diseño arquitectónico

El diseño arquitectónico de un sistema enfatiza el diseño de la arquitectura de sistema que describe la estructura, el comportamiento y más vistas de ese sistema y análisis.

Diseño lógico

El diseño lógico de un sistema se refiere a una representación abstracta del flujo de datos, entradas y salidas del sistema. Esto se lleva a cabo a menudo a través de la modelización, utilizando un modelo muy abstracto (y a veces gráfico) del sistema real. En el contexto de los sistemas, los diseños son incluidos. El diseño lógico incluye diagramas de entidad-relación.

Diseño físico

El diseño físico se relaciona con los procesos de entrada y salida reales del sistema. Esto está explicado en términos de cómo se introducen los datos a un sistema, cómo son verificados o autenticados, cómo son procesados y cómo se

acaban mostrando. En el diseño físico, se deciden los siguientes requisitos sobre el sistema.

- Requisito de entrada,
- Requisitos de salida,
- Requisitos de almacenamiento,
- Requisitos de procesamiento,
- Control de sistema y copia de seguridad o recuperación.

En otras palabras, la parte física del diseño de un sistema generalmente se puede dividir en tres subtareas:

- Diseño de la Interfaz del Usuario
- Diseño de los Datos
- Diseño del Proceso.

El Diseño de Interfaz del Usuario se preocupa por la manera en la que los usuarios añaden información al sistema y la forma en la que el sistema presenta la información a estos. El Diseño de los Datos se centra en cómo el dato está representado y almacenado dentro del sistema. Finalmente, el Diseño del Proceso se ocupa de la forma en la que los datos son manejados en el sistema, y de cómo y dónde se validan, aseguran y/o transforman a medida que fluyen dentro, a través y fuera del sistema. Al final de la fase de diseño del sistema, se produce la documentación que describe las tres subtareas y se pone a su disposición para su uso en la siguiente fase.

El diseño físico, en este contexto, no se refiere al diseño físico tangible de un sistema de información. Utilizando una analogía, el diseño físico de un ordenador personal implica la entrada a través de un teclado, el procesamiento dentro de la CPU, y su correspondiente salida a través de un monitor, impresora, etc. No se trataría del diseño real del hardware tangible, que en el caso de un PC sería un monitor, CPU, placa base, disco duro, módems, tarjetas gráficas, ranuras de USB, etc. Implica un diseño detallado de un usuario y un procesador de estructura de base de datos de productos y un procesador de control. La especificación personal H/S se desarrolla para el sistema propuesto.

1.5.2.- Núcleo (Kernel) y niveles de un sistema operativo.

Es el software que constituye el núcleo del sistema operativo, dónde se realizan las funcionalidades básicas como la gestión de procesos, la gestión de memoria y de entrada salida.

El " kernel " del sistema operativo controla todas las operaciones que implican procesos y representa solo una pequeña porción del código de todo el Sistema Operativo pero es de amplio uso.

Niveles del Sistema Operativo

Concepto de Kernel

Para que una computadora pueda arrancar y funcionar, no es necesario que tenga un núcleo para poder usarse. Los programas pueden cargarse y ejecutarse directamente en una computadora «vacía», siempre que sus autores quieran desarrollarlos sin usar ninguna abstracción del hardware ni ninguna ayuda del sistema operativo. Ésta era la forma normal de usar muchas de las primeras computadoras: para usar distintos programas se tenía que reiniciar y reconfigurar la computadora cada vez. Con el tiempo, se empezó a dejar en memoria (aún entre distintas ejecuciones) pequeños programas auxiliares, como el cargador y el depurador, o se cargaban desde memoria de sólo lectura. A medida que se fueron desarrollando, se convirtieron en los fundamentos de lo que llegarían a ser los primeros núcleos de sistema operativo.

El kernel presenta al usuario o los programas de aplicación una interfaz de programación de alto nivel, implementando la mayoría de las facilidades requeridas por éstos. Reúne el manejo de una serie de siguientes conceptos ligados al hardware de nivel más bajo:

- Procesos (tiempo compartido, espacios de direccionamiento protegidos);

- Señales y Semáforos;

- Memoria Virtual ("swapping", paginado);

- Sistema de Archivos;

- Tubos ("pipes") y Conexiones de red.

Una parte del kernel es independiente de los dispositivos presentes en el sistema, pero otra contiene los controladores necesarios para manejar partes específicas del hardware. El kernel interpreta los pedidos de los programas y los

traduce en secuencias de bits que, presentadas a los registros de los controladores, operan sobre los dispositivos físicos. Por ejemplo, el código de un kernel de Linux está escrito casi todo en C, salvo una pequeña parte en lenguaje ensamblador para los procesos de bajo nivel. El tamaño puede ir desde unos 400 KB hasta más de 50 MB de código fuente.

Funcionamiento del Kernel

Tipos de Núcleos.

En función del tamaño y de las funcionalidades que posea el kernel podemos clasificarlo. Realmente, y pese a seguidores incondicionales en un modelo u otro, existe una tendencia básica a reducir el tamaño del núcleo proporcionando menos funcionalidades, que son desplazadas a módulos que se cargan en tiempo de ejecución. En función a esta idea tenemos tres tipos fundamentales de kernel:

Kernel monolítico

Todas las funcionalidades posibles están integradas en el sistema. Se trata de un programa de tamaño considerable que deberemos recompilar al completo cada vez que queramos añadir una nueva posibilidad. Esta es la estructura original de Linux. Por tratarse de una técnica clásica y desfasada el creador de Linux fue muy criticado.

Por ejemplo, los primeros kernels de Linux, UNIX y DOS

Kernel modular

Se trata de la tendencia actual de desarrollo. En el kernel se centran las funcionalidades esenciales como la administración de memoria, la planificación de procesos, etc. Sin embargo, no tiene sentido que el núcleo de un sistema operativo englobe toda la parafernalia para comunicarse con todas las posibles de tarjetas de vídeo o de sonido. En otros sistemas operativos esto se soluciona con unos ficheros proporcionados por el fabricante llamados drivers. En Linux se creó un interfaz adecuado para posibilitar el desarrollo de módulos que cumplieran esas funcionalidades. Esos módulos pueden ser compilados por separado y añadidos al kernel en tiempo de ejecución.

Por ejemplo: Actualmente el kernel de linux.

Estructura de Microkernel

Esta técnica pretende reducir a su mínima expresión el kernel, dejando a los niveles superiores el resto de las funcionalidades. Existen algunos kernels que lo utilizan, si bien el que centra nuestra atención es Hurd. Se trata del último kernel GNU llamado a sustituir a Linux como núcleo del sistema operativo. Aunque esta estrategia de diseño es tan antigua como la modular, no ha sido tenida en cuenta hasta ahora debido a las limitaciones de rendimiento que tenía.

Ejemplos: AIX, La familia de micronúcleos L4, El micronúcleo Mach (usado en GNU Hurd y en Mac OS X), Minix, MorphOS, QNX, RadiOS, VSTa

Además de estos existen otra gran cantidad de núcleos, por ejemplo

· Nanokernel (AdeOS, Eros, KeyKOS, Brix-OS).

· VOiD (unununium, TUNES, Vapour).

· Sasos (Opal, Mungi, BriX).

· VM (Merlin, Argante).

· Exokernel (MIT exokernel).

· Cache kernel (universidad de stanford).

Niveles del Sistema Operativo

Explicación a Detalle

Para grandes sistemas operativos, que van desde cientos de miles a millones de líneas de código, la programación modular por si sola no es suficiente. En su lugar, ha ido creciendo el uso de conceptos como los de niveles jerárquicos y abstracción de la información. La estructura jerárquica de un sistema operativo moderno separa sus funciones de acuerdo a su complejidad, su escala característica de tiempo y su nivel de abstracción. Se puede contemplar al sistema como una serie de niveles. Cada nivel lleva a cabo un determinado subconjunto de funciones requeridas por el sistema operativo. Este se basa en el nivel inferior para llevar a cabo funciones más primitivas y ocultar los detalles de dichas funciones. De este modo, se descompone un problema en un número de subproblemas más manejables.

En general, las capas más bajas trabajan con escalas de tiempo más cortas. Algunas partes del sistema operativo deben interactuar directamente con el hardware del computador, donde los sucesos pueden tener una escala de tiempo tan breve como unas pocas billonésimas de segundo.

El modelo está definido en la tabla de arriba y consta de los siguientes niveles :

• Nivel 1: Circuitos electrónicos, los objetos que se tratan son registros, celdas de memoria y puertas lógicas. Las operaciones definidas sobre estos objetos son acciones tales como borrar un registro o leer una posición de memoria.

• Nivel 2: Es el conjunto de instrucciones del procesador. Las operaciones a este nivel son aquellas permitidas por el conjunto de instrucciones del lenguaje de la máquina, tales como SUMAR, RESTAR, CARGAR y DEPOSITAR.

• Nivel 3: Añade el concepto de procedimiento o subrutina, así como las operaciones de llamada y retomo.

• Nivel 4: Introduce las interrupciones, lo cual involucra a una rutina de tratamiento de la interrupción.

Estos primeros cuatro niveles no forman parte del sistema operativo, sino que constituyen el hardware del procesador. Es en el nivel 5 en el que comienza a alcanzarse el sistema operativo.

• Nivel 5: Se introduce la noción de proceso como un programa en ejecución. Entre los requisitos fundamentales de un sistema operativo que ofrezca soporte para múltiples procesos se incluye la capacidad de suspender y reanudar los procesos. Esto exige salvaguardar los registros del hardware, de modo que la ejecución pueda cambiar de un proceso a otro. Además, si los procesos necesitan cooperar, hace falta algún método de sincronización. Una de las técnicas más simples, pero un concepto importante en el diseño de sistemas operativos, es el semáforo.

• Nivel 6: Tiene que ver con los dispositivos de almacenamiento secundario. En este nivel se sitúan las funciones de ubicación de las cabezas de lectura y escritura, y se producen las transferencias reales de bloques. Este nivel se apoya en el nivel 5 para planificar las operaciones y notificar al proceso que hizo la solicitud que la operación ha culminado.

• Nivel 7: Crea un espacio de direcciones lógicas para los procesos. Este nivel organiza el espacio de direcciones virtuales en bloques que se pueden mover entre la memoria principal y la memoria secundaria. Son tres los esquemas de uso más habitual: los que utilizan páginas de longitud fija, los que usan segmentos de longitud variable y los que utilizan los dos.

Hasta este punto, el sistema operativo se ocupa de los recursos de un solo procesador. Empezando por el nivel 8, el sistema operativo trata con objetos externos.

• Nivel 8: Se dedica a la comunicación de información y mensajes entre los procesos. Mientras que el nivel 5 proporciona el mecanismo de señalización primitivo que permite la sincronización entre procesos, este nivel trata con una forma más completa de compartir información. Una de las herramientas más potentes en este nivel es el pipe, que es un canal lógico para el flujo de datos entre los procesos. Un pipe se define con su salida en un proceso y su entrada en otro proceso. También se pueden usar para enlazar dispositivos externos o archivos con los procesos.

• Nivel 9: Da soporte al almacenamiento a largo plazo de los archivos con nombre. En este nivel, los datos del almacenamiento secundario se contemplan en términos de entidades abstractas de longitud variable, en contraste con el enfoque orientado al hardware del nivel 6, en términos de pistas, sectores y bloques de tamaño fijo.

• Nivel 10: Es el que proporciona acceso a los dispositivos externos mediante interfaces estandarizadas.

• Nivel 11: Es responsable de mantener la asociación entre los identificadores externos e internos de los recursos y objetos del sistema. El identificador externo es un nombre que puede ser empleado por una aplicación o un usuario. El identificador interno es una dirección que es utilizada en los niveles inferiores del sistema operativo para ubicar y controlar un objeto. Estas asociaciones se mantienen en un directorio. Las entradas incluyen no sólo las asociaciones externo/interno, sino otras características, como los derechos de acceso.

• Nivel 12: Proporciona servicios completos de soporte a los procesos. Esto va mucho más allá que lo que se ofrece en el nivel 5. En el nivel 5, sólo se

mantienen los contenidos de los registros del procesador asociados con un proceso, junto a la lógica para expedir a los procesos. En el nivel 12, se da soporte a toda la información necesaria para la gestión ordenada de los procesos. Esto incluye el espacio de direcciones virtuales del proceso, una lista de objetos y procesos con los que puede interactuar y las limitaciones de dicha interacción, los parámetros pasados al proceso en su creación y cualesquiera otras características del proceso que pudieran ser utilizadas por el sistema operativo para su control.

• Nivel 13: Ofrece al usuario una interfaz con el sistema operativo. Se denomina shell y separa al usuario de los detalles, le presenta el sistema operativo como un simple conjunto de servicios. El shell acepta las órdenes del usuario o las sentencias de control de trabajos, las interpreta, crea y controla los procesos según sea necesario.

1.5.3.- Programación de entrada/salida.

Principios del Software de Entrada y Salida

Los principios de software en la entrada - salida se resumen en cuatro puntos: el software debe ofrecer manejadores de interrupciones, manejadores de dispositivos, software que sea independiente de los dispositivos y software para usuarios.

· Manejadores de Interrupciones. El primer objetivo referente a los manejadores de interrupciones consiste en que el programador o el usuario no debe darse cuenta de los manejos de bajo nivel para los casos en que el dispositivo está ocupado y se debe suspender el proceso o sincronizar algunas tareas. Desde el punto de vista del proceso o usuario, el sistema simplemente se tardó más o menos en responder a su petición.

· Manejadores de Dispositivos. El sistema debe proveer los manejadores de dispositivos necesarios para los periféricos, así como ocultar las peculiaridades del manejo interno de cada uno de ellos, tales como el formato de la información, los medios mecánicos, los niveles de voltaje y otros. Por ejemplo, si el sistema tiene varios tipos diferentes de discos duros, para el usuario o programador las diferencias técnicas entre ellos no le deben importar, y los manejadores le deben ofrecer el mismo conjunto de rutinas para leer y escribir datos.

· Software que sea independiente de los dispositivos. Este es un nivel superior de independencia que el ofrecido por los manejadores de dispositivos. Aquí el sistema operativo debe ser capaz, en lo más posible, de ofrecer un conjunto de utilerías para accesar periféricos o programarlos de una manera consistente. Por ejemplo, que para todos los dispositivos orientados a bloques se tenga una llamada para decidir si se desea usar 'buffers' o no, o para posicionarse en ellos.

· Software para Usuarios. La mayoría de las rutinas de entrada - salida trabajan en modo privilegiado, o son llamadas al sistema que se ligan a los programas del usuario formando parte de sus aplicaciones y que no le dejan ninguna flexibilidad al usuario en cuanto a la apariencia de los datos. Existen otras librerías en donde el usuario si tiene poder de decisión (por ejemplo la llamada a "printf" en el lenguaje "C"). Otra facilidad ofrecida son las áreas de trabajos encolados (spooling áreas), tales como las de impresión y correo electrónico.

Manejo de los Dispositivos de E/S

En el manejo de los dispositivos de E/S es necesario, introducir dos nuevos términos:

· Buffering

El buffering (uso de memoria intermedia) trata de mantener ocupados tanto la CPU como los dispositivos de E/S. La idea es sencilla, los datos se leen y se almacenan en un buffer, una vez que los datos se han leído y la CPU va a iniciar inmediatamente la operación con ellos, el dispositivo de entrada es instruido para iniciar inmediatamente la siguiente lectura. La CPU y el dispositivo de entrada permanecen ocupados. Cuando la CPU esté libre para el siguiente grupo de datos, el dispositivo de entrada habrá terminado de leerlos. La CPU podrá empezar el proceso de los últimos datos leídos, mientras el dispositivo de entrada iniciará la lectura de los datos siguientes. Para la salida, el proceso es análogo. En este caso los datos de salida se descargan en otro buffer hasta que el dispositivo de salida pueda procesarlos. Este sistema soluciona en forma parcial el problema de mantener ocupados todo el tiempo la CPU y los dispositivos de E/S. Ya que todo depende del tamaño del buffer y de la velocidad de procesamiento tanto de la CPU como de los dispositivos de E/S.

El manejo de buffer es complicado. Uno de los principales problemas reside en determinar tan pronto como sea posible que un dispositivo de E/S a finalizado

una operación. Este problema se resuelve mediante las interrupciones. Tan pronto como un dispositivo de E/S acaba con una operación interrumpe a la CPU, en ese momento la CPU detiene lo que está haciendo e inmediatamente transfiere el control a una posición determinada. Normalmente las instrucciones que existen en esta posición corresponden a una rutina de servicio de interrupciones. La rutina de servicio de interrupción comprueba si el buffer no está lleno o no está vacío y entonces inicia la siguiente petición de E/S. La CPU puede continuar entonces el proceso interrumpido. Cada diseño de computadora tiene su propio mecanismo de interrupción, pero hay varias funciones comunes que todos contemplan. El buffering puede ser de gran ayuda pero pocas veces es suficiente.

· Spooling

El problema con los sistemas de cintas es que una lectora de tarjetas no podía escribir sobre un extremo mientras la CPU leía el otro. Los sistemas de disco eliminaron esa dificultad, moviendo la cabeza de un área del disco a otra. En un sistema de discos, las tarjetas se leen directamente desde la lectora sobre el disco. Las posiciones de las imágenes de las tarjetas se registran en una tabla mantenida por el sistema operativo. En la tabla se anota cada trabajo una vez leído. Cuando se ejecuta un trabajo sus peticiones de entrada desde la tarjeta se satisfacen leyendo el disco. Cuando el trabajo solicita la salida, ésta se copia en el buffer del sistema y se escribe en el disco. Cuando la tarea se ha completado se escribe en la salida realmente. Esta forma de procesamiento se denomina spooling, utiliza el disco como un buffer muy grande para leer tan por delante como sea posible de los dispositivos de entrada y para almacenar los ficheros hasta que los dispositivos de salida sean capaces de aceptarlos. La ventaja sobre el buffering es que el spooling solapa la E/S de un trabajo con la computación de otro. Es una característica utilizada en la mayoría de los sistemas operativos. Afecta directamente a las prestaciones. Por el costo de algo de espacio en disco y algunas tablas, la CPU puede simultanear la computación de un trabajo con la E/S de otros. De esta manera, puede mantener tanto a la CPU como a los dispositivos de E/S trabajando con un rendimiento mucho mayor. Además mantiene una estructura de datos llama Job spooling, que hace que los trabajos ya leídos permanezcan en el disco y el sistema operativo puede seleccionar cual ejecutar, por lo tanto se hace posible la planificación de trabajos.

Manejo de entradas y salidas (UNIX)

El sistema de entrada/salida se divide en dos sistemas complementarios: el estructurado por bloques y el estructurado por caracteres. El primero se usa para manejar cintas y discos magnéticos, y emplea bloques de tamaño fijo (512 o 1024 bytes) para leer o escribir. El segundo se utiliza para atender a las terminales, líneas de comunicación e impresoras, y funciona byte por byte.

En general, el sistema Unix emplea programas especiales (escritos en C) conocidos como manejadores (drivers) para atender a cada familia de dispositivos de E/S. Los procesos se comunican con los dispositivos mediante llamadas a su manejador. Además, desde el punto de vista de los procesos, los manejadores aparecen como si fueran archivos en los que se lee o escribe; con esto se logra gran homogeneidad y elegancia en el diseño.

Cada dispositivo se estructura internamente mediante descriptores llamados número mayor, número menor y clase (de bloque o de caracteres). Para cada clase hay un conjunto de entradas, en una tabla, que aporta a los manejadores de los dispositivos. El número mayor se usa para asignar manejador, correspondiente a una familia de dispositivos; el menor pasa al manejador como un argumento, y éste lo emplea para tener acceso a uno de varios dispositivos físicos semejantes.

Las rutinas que el sistema emplea para ejecutar operaciones de E/S están diseñadas para eliminar las diferencias entre los dispositivos y los tipos de acceso. No existe distinción entre acceso aleatorio y secuencial, ni hay un tamaño de registro lógico impuesto por el sistema. El tamaño de un archivo ordinario está determinado por el número de bytes escritos en él; no es necesario predeterminar el tamaño de un archivo.

El sistema mantiene una lista de áreas de almacenamiento temporal (buffers), asignadas a los dispositivos de bloques. El Kernel usa estos buffers con el objeto de reducir el tráfico de E/S. Cuando un programa solicita una transferencia, se busca primero en los buffers internos para ver si el bloque que se requiere ya se encuentra en la memoria principal (como resultado de una operación de lectura anterior). Si es así, entonces no será necesario realizar la operación física de entrada o salida.

Existe todo un mecanismo de manipulación interna de buffers (y otro de manejo de listas de bytes), necesario para controlar el flujo de datos entre los dispositivos de bloques (y de caracteres) y los programas que los requieren.

Por último, y debido a que los manejadores de los dispositivos son programas escritos en lenguaje C, es relativamente fácil reconfigurar el sistema para ampliar o eliminar dispositivos de E/S en la computadora, así como para incluir tipos nuevos

1.5.4.- Interrupciones del procesador.

Una interrupción es una instrucción que detiene la ejecución de un programa para permitir el uso de la UCP en un proceso prioritario. Una vez concluido este último proceso se devuelve el control a la aplicación anterior.

Las interrupciones ocurren muy seguido, sencillamente la interrupción que actualiza la hora del día ocurre aproximadamente 18 veces por segundo. Para lograr administrar todas estas interrupciones, la computadora cuenta con un espacio de memoria, llamado memoria baja, donde se almacenan las direcciones de cierta localidad de memoria donde se encuentran un juego de instrucciones que la UCP ejecutará, para después regresar a la aplicación en proceso.

La interrupción es generada por el hardware del sistema de cómputo. Cuando ocurre una interrupción:

A. El sistema operativo toma el control (es decir, el hardware pasa el control al sistema operativo).

B. El sistema operativo guarda el estado del proceso interrumpido.

C. El sistema operativo analiza la interrupción y transfiere el control a la rutina apropiada para atenderla; en muchos sistemas actuales el hardware se encarga de esto automáticamente.

D. La rutina del manejador de interrupciones procesa la interrupción.

E. Se restablece el estado del proceso interrumpido (o del “siguiente proceso”).

F. Se ejecuta el proceso interrumpido (o el “siguiente proceso”).

Una interrupción puede ser iniciada específicamente por un proceso en ejecución (en cuyo caso se suele denominar trampa (trap), y se dice que está

sincronizada con la operación del proceso) o puede ser causada por algún evento que puede estar relacionado o no con el proceso en ejecución (en cuyo caso se dice que es asíncrona con la operación del proceso).

Los sistemas orientados hacia las interrupciones pueden sobrecargarse. Si estás llegan con mucha frecuencia, el sistema no será capaz de atenderlas.

Existen seis clases de interrupciones:

Interrupciones SVC (supervisor call, llamadas al supervisor). Son iniciadas por un proceso en ejecución que ejecute la instrucción SVC. Una SVC es una petición generada por el usuario de un servicio particular del sistema, como realizar una operación de entrada/salida, obtener más memoria o comunicarse con el operador del sistema. El mecanismo de las SVC ayuda a proteger el sistema operativo de las acciones de los usuarios. Un usuario no puede entrar arbitrariamente al sistema operativo, sino que debe solicitar un servicio por medio de una SVC.

Interrupciones de E/S. Son iniciadas por hardware de entrada y salida. Estas interrupciones indican a la UCP el cambio de estado de un canal o dispositivo. Las interrupciones de E/S se producen cuando finaliza una operación de E/S o cuando un dispositivo pasa al estado listo.

Interrupciones Externas. Son causadas por diversos eventos, incluyendo la expiración de un cuanto de un reloj que interrumpe, la pulsación de la tecla de interrupción de la consola o la recepción de una señal procedente de otro procesador en un sistema de múltiples procesadores.

Interrupciones de Reinicio. Se produce cuando se presiona el botón de reinicio de la PC o cuando llega de otro procesador una instrucción de reinicio en un sistema de multiprocesamiento

Interrupciones de verificación del programa. Son causadas por una amplia clase de problemas que pueden ocurrir cuando se ejecutan las instrucciones en lenguaje máquina de un programa. Dichos problemas incluyen la división entre cero, el exceso o defecto de los números que pueden ser manejados por las operaciones aritméticas, el intento de hacer referencia a una localidad de memoria que esté fuera de los límites de la memoria real. Muchos sistemas

ofrecen a los usuarios la opción de especificar las rutinas que deben ejecutarse cuando ocurra una interrupción de verificación del programa.

Interrupciones de verificación de la máquina. Son ocasionadas por el mal funcionamiento del hardware.

UNIDAD II

ADMINISTRACIÓN DE PROCESOS

La Planificación de procesos tiene como principales objetivos la equidad, la eficacia, el tiempo de respuesta, el tiempo de regreso y el rendimiento. Equidad: Todos los procesos deben ser atendidos, Eficacia: El procesador debe estar ocupado el 100% del tiempo, Tiempo de respuesta: El tiempo empleado en dar respuesta a las solicitudes del usuario debe ser el menor posible, Tiempo de regreso: Reducir al mínimo el tiempo de espera de los resultados esperados por los usuarios por lotes y Rendimiento: Maximizar el número de tareas que se procesan por cada hora.

2.1.- Concepto de proceso.

Proceso es un conjunto o encadenamiento de fenómenos, asociados al ser humano o a la naturaleza, que se desarrollan en un periodo de tiempo finito o infinito y cuyas fases sucesivas suelen conducir hacia un fin específico.

La palabra proceso es un sustantivo masculino que se refiere de un modo general a la acción de ir hacia adelante. Proviene del latín processus, que significa avance, marcha, progreso, desarrollo.

Debido a su amplitud, podemos identificar procesos en una enorme cantidad de ámbitos dentro la actividad humana o fuera de ella, es decir, que tienen lugar en el medio natural. Los ejemplos los encontramos en nuestro día a día, en la manera cómo desarrollamos nuestras actividades o en nuestro entorno.

2.2.- Concurrencia y secuencialidad.

Es la existencia de varias actividades ejecutándose simultáneamente, y necesitan sincronizarse para actuar conjuntamente. Se trata, en este caso, de un concepto lógico, ya que sólo hace referencia a las actividades, sin importar el número de procesadores presentes.

Para que dos actividades, sean concurrentes, es necesario que tengan relación entre sí, como puede ser la cooperación en un trabajo determinado o el uso de información compartida.

Los procesos son concurrentes si existen simultáneamente. Los procesos concurrentes pueden funcionar en forma totalmente independiente unos de otros, o pueden ser asíncronos, lo cual significa que en ocasiones requiere cierta sincronización y cooperación.

En un sistema monoprocesador, la existencia de multiprogramación es condición necesaria, pero no suficiente para que exista concurrencia, ya que los procesos pueden ejecutarse independientemente. Por ejemplo, un editor y un compilador pueden estar ejecutándose simultáneamente en una computadora sin que exista concurrencia entre ellos. Por otro lado, si un programa se está ejecutando y se encuentra grabando datos en un archivo, y otro programa también en ejecución está leyendo datos de ese mismo archivo, sí existe concurrencia entre ellos, pues el funcionamiento de uno interfiere en el funcionamiento de otro.

Si un sistema es multiprocesador, también pueden presentarse situaciones de concurrencia siempre y cuando las actividades necesiten actuar entre sí, bien por utilizar información común, o por cualquier otra causa.

Los procesos del sistema pueden ejecutarse concurrentemente, puede haber múltiples tareas en el CPU con varios procesos. Existen varias razones para permitir la ejecución concurrente:

Compartir recursos físicos: Ya que los recursos del hardware de la computadora son limitados, nos podemos ver obligados a compartirlos en un entorno multiusuario.

Compartir recursos lógicos: Puesto que varios usuarios pueden interesarse en el mismo elemento de información (por ejemplo, un archivo compartido),

debemos proporcionar un entorno que permita el acceso concurrente a estos tipos de recursos.

Acelerar los cálculos: Si queremos que una tarea se ejecute con mayor rapidez, debemos dividirla en subtareas, cada una de las cuales se ejecutara, en paralelo con las demás.

Modularidad: Podremos construir el sistema en forma modular, dividiendo las funciones del sistema en procesos separados.

Comodidad: Un usuario puede tener que ejecutar varias tareas a la vez, por ejemplo, puede editar, imprimir y compilar en paralelo.

La ejecución concurrente que requiere la cooperación entre procesos necesita un mecanismo para la sincronización y comunicación de procesos, exclusión mutua y sincronización.

2.3.- Regiones críticas.

Una región crítica es una secuencia de instrucciones que no debe ser interrumpida por otros procesos, es decir, se debe tratar una región crítica como una sola instrucción atómica.

No es suficiente que los recursos usados en una región crítica no deban ser alterados por otros procesos, porque es posible que su valor o contenido en el momento de lectura no sean válidos; puede ser que estén en un estado transitorio. Sin embargo, si los accesos concurrentes solamente leen pueden estar permitidos (más sobre el tema veremos más adelante).

Normalmente se protege en los lenguajes de programación solamente el código directamente, los datos están protegidos indirectamente por su código de acceso.

Por ejemplo, en Java no se puede declarar una clase como synchronized, sino solamente sus métodos. Eso requiere mucha disciplina por parte del programador en cuanto a acceder a las variables críticas solamente con métodos adecuados.

2.4.- Exclusión mutua.

Los algoritmos de exclusión mutua (comúnmente abreviada como mutex por mutual exclusion) se usan en programación concurrente para evitar que entre

más de un proceso a la vez en la sección crítica. La sección crítica es el fragmento de código donde puede modificarse un recurso compartido.

La mayor parte de estos recursos son las señales, contadores, colas y otros datos que se emplean en la comunicación entre el código que se ejecuta cuando se da servicio a una interrupción y el código que se ejecuta el resto del tiempo. Se trata de un problema de vital importancia porque, si no se toman las precauciones debidas, una interrupción puede ocurrir entre dos instrucciones cualesquiera del código normal y esto puede provocar graves fallos.

La técnica que se emplea por lo común para conseguir la exclusión mutua es inhabilitar las interrupciones durante el conjunto de instrucciones más pequeño que impedirá la corrupción de la estructura compartida (la sección crítica). Esto impide que el código de la interrupción se ejecute en mitad de la sección crítica.

En un sistema multiprocesador de memoria compartida, se usa la operación indivisible test-and-set sobre una bandera, para esperar hasta que el otro procesador la despeje. La operación test-and-set realiza ambas operaciones sin liberar el bus de memoria a otro procesador. Así, cuando el código deja la sección crítica, se despeja la bandera. Esto se conoce como spin lock o espera activa.

Algunos sistemas tienen instrucciones multioperación indivisibles similares a las anteriormente descritas para manipular las listas enlazadas que se utilizan para las colas de eventos y otras estructuras de datos que los sistemas operativos usan comúnmente.

La mayoría de los métodos de exclusión mutua clásicos intentan reducir la latencia y espera activa mediante las colas y cambios de contexto. Algunos investigadores afirman que las pruebas indican que estos algoritmos especiales pierden más tiempo del que ahorran.

A pesar de todo lo dicho, muchas técnicas de exclusión mutua tienen efectos colaterales. Por ejemplo, los semáforos permiten interbloqueos (deadlocks) en los que un proceso obtiene un semáforo, otro proceso obtiene el semáforo y ambos se quedan a la espera de que el otro proceso libere el semáforo. Otros efectos comunes incluyen la Inanición, en el cual un proceso esencial no se ejecuta durante el tiempo deseado, y la inversión de prioridades, en el que una

tarea de prioridad elevada espera por otra tarea de menor prioridad, así como la latencia alta en la que la respuesta a las interrupciones no es inmediata.

La mayor parte de la investigación actual en este campo, pretende eliminar los efectos anteriormente descritos. Si bien no hay un esquema perfecto conocido, hay un interesante esquema no clásico de envío de mensajes entre fragmentos de código que, aunque permite inversiones de prioridad y produce una mayor latencia, impide los interbloqueos.

Algunos ejemplos de algoritmos clásicos de exclusión mutua son:

El algoritmo de Dekker.

El algoritmo de Peterson.

2.5.- Sincronización.

Hablamos de la sincronización de datos como el proceso de alineación entre los datos provenientes de diversas fuentes, y su continua armonización en el tiempo. Vale decir, la coordinación de procesos que se ejecutan simultáneamente, a fin de obtener un orden de ejecución correcto y evitar errores del sistema.

Esta acción permite obtener datos precisos y fidedignos de todos los sistemas, sean éstos operativos o transaccionales. Así, cada organización podrá crear una lógica propia para la sincronización de sus datos en tiempo real, si se requiere, obteniendo una mayor precisión, coherencia de la información y una mejora sustancial del rendimiento.

Iniciativas:

- Proporciona acceso nativo a casi todos los tipos de datos de cualquier sistema.
- Sincroniza formatos de datos estructurados, no estructurados y semi-estructurados de sistemas operativos y transaccionales.
- Integra y suministra datos operativos en tiempo real.
- Facilita los datos más actuales en el lugar, momento y formato determinado.
- Relaciona y limpia los datos antes de realizar la sincronización.

- Proporciona una gestión de los permisos basada en funciones, flexible y a nivel individual para garantizar la seguridad de los datos, incluso a través de redes WAN y firewalls empresariales.
- Facilita la colaboración y reutilización entre proyectos.

UNIDAD III

INTERBLOQUEO (DEAD LOCK)

En general, una elevada utilización general de recursos y la posibilidad de la operación paralela de muchos dispositivos de entrada/salida gobernados por procesos concurrentes, contribuyen significativamente a aumentar el potencial de rendimiento de los sistemas multitarea y multiprogramación. Al mismo tiempo, la concurrencia y la elevada utilización de recursos también proporcionan las condiciones necesarias y la base para que se produzcan interbloqueos que puedan ir en detrimento del rendimiento del sistema.

3.1.- Análisis.

El interbloqueo es un problema que afecta a procesos concurrentes que utilizan recursos en un sistema.

Los procesos solicitan recursos al sistema y los liberan cuando ya no los necesitan. Un recurso puede estar disponible o bien asignado a algún proceso.

Muchas veces, el interbloqueo no es responsabilidad de las aplicaciones, sino del sistema de gestión de recursos.

En un sistema informático se ejecutan de forma concurrente múltiples procesos que, normalmente, no son independientes, sino que compiten en el uso exclusivo de recursos y se comunican y sincronizan entre sí. El sistema operativo debe encargarse de asegurar que estas interacciones se llevan a cabo apropiadamente, proporcionando la sincronización requerida por las mismas, tal como se analiza en el capítulo dedicado a ese aspecto. Sin embargo, generalmente, no basta con esto. Las necesidades de algunos procesos pueden entrar en conflicto entre sí provocando que estos se bloqueen permanentemente. Esta situación se denomina interbloqueo (en inglés, se usa normalmente el término deadlock). En este capítulo se estudiará este problema y se analizarán sus posibles soluciones. Se trata de un problema general ya identificado en la década de los sesenta del siglo XX y estudiado ampliamente desde entonces. En el campo de los sistemas operativos, como se explicará en este capítulo, se han adoptado algunas de las soluciones ideadas en el ámbito teórico para evitar que aparezca este problema en el funcionamiento interno del propio sistema operativo. Sin embargo, aunque pueda parecer sorprendente a priori, por razones que se comprenderán a lo largo de este tema, no ha sido así para los servicios que ofrecen los sistemas operativos. La mayoría de los sistemas operativos actuales no aseguran de forma completa que los procesos que usan sus servicios no se vean afectados por esta patología.

3.2.- Prevención.

La estrategia de prevención del interbloqueo consiste, a grandes rasgos, en diseñar un sistema de manera que esté excluida, a priori, la posibilidad de interbloqueo. Los métodos para prevenir el interbloqueo son de dos tipos. Los métodos indirectos consisten en impedir la aparición de alguna de las tres condiciones necesarias, antes mencionadas (condiciones 1 a 3). Los métodos directos consisten en evitar la aparición del círculo vicioso de espera (condición 4). Se examinarán a continuación las técnicas relacionadas con cada una de las cuatro condiciones.

Exclusión Mutua

En general, la primera de las cuatro condiciones no puede anularse. Si el acceso a un recurso necesita exclusión mutua, el sistema operativo debe soportar la exclusión mutua. Algunos recursos, como los archivos, pueden permitir varios accesos para lectura, pero sólo accesos exclusivos para escritura. Incluso en

este caso, se puede producir interbloqueo si más de un proceso necesita permiso de escritura.

Retención y Espera

La condición de retención y espera puede prevenirse exigiendo que todos los procesos soliciten todos los recursos que necesiten a un mismo tiempo y bloqueando el proceso hasta que todos los recursos puedan concederse simultáneamente. Esta solución resulta ineficiente por dos factores. En primer lugar, un proceso puede estar suspendido durante mucho tiempo, esperando que se concedan todas sus solicitudes de recursos, cuando de hecho podría haber avanzado con sólo algunos de los recursos. Y en segundo lugar, los recursos asignados a un proceso pueden permanecer sin usarse durante periodos considerables, tiempo durante el cual se priva del acceso a otros procesos.

No apropiación

La condición de no apropiación puede prevenirse de varias formas. Primero, si a un proceso que retiene ciertos recursos se le deniega una nueva solicitud, dicho proceso deberá liberar sus recursos anteriores y solicitarlos de nuevo, cuando sea necesario, junto con el recurso adicional. Por otra parte, si un proceso solicita un recurso que actualmente está retenido por otro proceso, el sistema operativo puede expulsar al segundo proceso y exigirle que libere sus recursos. Este último esquema evitará el interbloqueo sólo si no hay dos procesos que posean la misma prioridad. Esta técnica es práctica sólo cuando se aplica a recursos cuyo estado puede salvarse y restaurarse más tarde de una forma fácil, como es el caso de un procesador.

Círculo Vicioso de Espera

La condición del círculo vicioso de espera puede prevenirse definiendo una ordenación lineal de los tipos de recursos. Si a un proceso se le han asignado recursos de tipo R, entonces sólo podrá realizar peticiones posteriores sobre los recursos de los tipos siguientes a R en la ordenación. Para comprobar el funcionamiento de esta estrategia, se asocia un índice a cada tipo de recurso. En tal caso, el recurso R, antecede a R, en la ordenación sii <j.

Entonces, supóngase que dos procesos A y B se interbloquean, porque A ha adquirido R, y solicitado Ry, mientras que B ha adquirido R;y solicitado R¿. Esta situación es imposible porque implica que í<jyj<i. Como en la retención y espera, la prevención del círculo vicioso de espera puede ser ineficiente, retardando procesos y denegando accesos a recursos innecesariamente

3.3.- Detección y recuperación.

En vez de sacrificar el rendimiento previniendo o evitando interbloqueos, algunos sistemas conceden libremente los recursos disponibles a los procesos solicitantes, y comprueban ocasionalmente el sistema para determinar si existen interbloqueos con el fin de reclamar los recursos retenidos por los procesos interbloqueados, si es que los hay.

Se ha demostrado que la existencia de un ciclo (o un circuito) en un grafo general de recursos es una condición necesaria para la existencia de interbloqueo. La existencia de un nudo, es decir, un ciclo en el cual de ninguno de los nodos que lo forman sale un camino que no sea ciclo, es condición suficiente para la existencia de interbloqueos en un grafo general de recursos. En sistemas con una sola instancia de cada tipo de recursos, la existencia de un ciclo es condición necesaria y suficiente para la existencia de interbloqueo. Por tanto, la existencia de interbloqueos puede determinarse utilizando conocidos algoritmos para determinación de ciclos o nudos en grafos. Un método práctico es intentar reducir el grafo general de recursos suprimiendo todas las retenciones y peticiones de cada proceso cuyas solicitudes puedan ser concedidas, hasta que se efectúen todas las reducciones posibles, Si el grafo queda completamente reducido (no quedan arcos) después de esta operación, el estado del sistema no está interbloqueado. En caso contrario, el sistema está interbloqueado, y los procesos que quedan, irreducibles, son los afectados. La capacidad del algoritmo para identificar los procesos interbloqueados es importante, ya que facilita considerablemente la recuperación de interbloqueos después de la detección.

A continuación, se presenta una versión diferente del algoritmo de detección de interbloqueos que utiliza estructuras de datos similares a las ya introducidas en relación con la evitación de interbloqueos. El algoritmo utiliza la matriz ASIGNADOS y el vector entero DISPONIBLES como antes. En vez de la matriz RECLAMADOS, se utiliza la matriz SOLICITADOS del mismo formato para

identificar el número de recursos de cada tipo solicitados, pero aún no concedidos a los procesos activos. Naturalmente, la suma de recursos solicitados y asignados no puede exceder de la capacidad máxima del sistema.

El algoritmo funciona contabilizando todas las posibilidades de secuenciar los procesos que quedan por completar. Si existe una secuencia de terminación, el sistema no está en un estado de interbloquco. En caso contrario, el sistema está interbloqueado, ya que no hay modo de que todos los procesos activos terminen.

En los métodos de detección de interbloqueos, el asignador de recursos simplemente concede toda petición de recursos disponibles. Cuando es invocado para determinar Si un estado determinado del sistema es un interbloqueo, el algoritmo funciona así:

Construir las estructuras ASIGNADOS, SOLICITADOS y DISPONIBLES de acuerdo con el estado del sistema. Desmarcar todos los procesos activos.

Encontrar un proceso no marcado tal que

SOLICITADOS DISPONIBLES

Si se encuentra, marcar el proceso i, actualizar DISPONIBLES,

DISPONIBLES: = DISPONIBLES + ASIGNADOS

y repetir este paso. Cuando no pueda hallarse ningún proceso cualificado, proseguir con el paso siguiente.

Si todos los procesos están marcados, el sistema no está interbloqueado. En caso contrario, el sistema está interbloqueado, y el conjunto de procesos no marcados es el afectado.

El algoritmo presentado tiene la misma complejidad temporal que el algoritmo de evitación de interbloqueos de la sección anterior, r ´ p2. Existen algoritmos más eficientes para casos especializados, tales como sistemas con una sola instancia de cada tipo de recurso o con un solo tipo de recurso con múltiples instancias. Para el caso general en que las peticiones de recursos están ordenadas por tamaño y hay una cuenta del número de recursos solicitados asociados con cada proceso se ha ideado un algoritmo cuya complejidad es linealmente proporcional al número de procesos activos. Este recargo de

mantenimiento adicional está parcialmente compensado por el hallazgo facilitado de un proceso bloqueado que debe ser activado cuando otros procesos liberen recursos.

La frecuencia de detección de interbloqueos es un parámetro de diseño del sistema. Una posibilidad es comprobar los interbloqueos cada vez que se solicita un recurso. Otra alternativa es activar el algoritmo de detección de interbloqueos ocasionalmente, digamos, a intervalos regulares o cuando uno o más procesos queden bloqueados durante un tiempo sospechosamente largo. La comprobación frecuente de interbloqueos tiene la ventaja de descubrir con prontitud los interbloqueos, capacitando al sistema para actuar rápidamente. La comprobación infrecuente de interbloqueos reduce el gasto de tiempo de ejecución para la detección de interbloqueos, a expensas de dejar interbloqueos sin detectar durante largos períodos de tiempo. Esta estrategia puede dar lugar a menor utilización de recursos ya que los procesos interbloqueados continúan reteniendo los recursos ya concedidos sin realizar ningún progreso.

La detección de interbloqueos es sólo una parte de la tarea de gestión de los interbloqueos. La detección de un interbloqueo sólo revela la existencia del problema, el sistema debe entonces romper el interbloqueo para reclamar los recursos retenidos por los procesos bloqueados y asegurar que los procesos afectados puedan eventualmente completarse. El primer paso en la recuperación de interbloqueos es identificar los procesos interbloqueados. De aquí la conveniencia de que los algoritmos de detección proporcionen una indicación de los procesos interbloqueados, como hace el algoritmo descrito en esta sección. El siguiente paso es romper el interbloqueo volviendo atrás o reiniciando desde el principio uno o más de los procesos interbloqueados. Reiniciar un proceso implica la pérdida del trabajo completado por el proceso antes de quedar bloqueado. Puesto que presumiblemente no todos los procesos han llegado igualmente lejos, es deseable elegir las víctimas entre los procesos cuya reiniciación sea menos gravosa. La vuelta atrás de un proceso requiere una utilidad que registre los estados en tiempo de ejecución de los procesos, de modo que un proceso pueda regresar a un instante suficientemente anterior en el pasado para que el interbloqueo se rompa. Tal utilidad la proporcionan algunos sistemas en donde se desea un alto grado de fiabilidad y/o disponibilidad. Ejemplos habituales incluyen la utilización de puntos de comprobación en sistemas de tiempo real y el mecanismo de anotaciones en

sistemas de procesamiento de transacciones. Cuando están presentes, tales mecanismos pueden ser también empleados para recuperación de interbloqueos. En general, tanto la vuelta atrás como la reiniciación pueden ser difíciles, si no imposibles, para procesos que han efectuado cambios irreversibles sobre los recursos de que disponían antes de que se produjera el interbloqueo.

En resumen, la detección y recuperación de interbloqueos proporciona un mayor grado potencial de concurrencia que la prevención o la evitación de interbloqueos Además, el recargo en tiempo de ejecución de la detección de interbloqueos puede ser un parámetro ajustable en el sistema. El precio a pagar es el recargo de recuperación una vez que se detectan los interbloqueos, y el desaprovechamiento del uso de los recursos del sistema por los procesos que son reiniciados o vueltos atrás. La recuperación de interbloqueos puede ser atractiva en sistemas con baja probabilidad de interbloqueos. En sistemas con elevada carga, se sabe que la concesión sin restricciones de peticiones de recurso conduce a un compromiso excesivo de los recursos, lo que puede dar lugar a frecuentes interbloqueos. Los efectos negativos de permitir interbloqueos tienden a incrementarse con la carga del sistema y con la frecuencia de interbloqueos, aumentando así el desaprovechamiento de los recursos del sistema justo cuando son más necesarios.

3.4.- Mecanismos para evitarlo.

La estrategia empleada con más frecuencia por los diseñadores para tratar el bloqueo mutuo es la prevención. En esta sección se examinan los métodos de prevención, junto con los efectos que tienen sobre los usuarios y los sistemas, sobre todo desde la perspectiva del rendimiento.

Havender (68) llegó a la conclusión de que si falta alguna de las cuatro condiciones necesarias no puede haber un interbloqueo. Este autor sugiere las siguientes estrategias para negar varias de esas condiciones:

Cada proceso deberá pedir todos sus recursos al mismo tiempo y no podrá seguir la ejecución hasta haberlos recibido todos.

Si a un proceso que tiene recursos se le niegan los demás, ese proceso deberá liberar sus recursos y, en caso necesario, pedirlos de nuevo junto con los recursos adicionales.

Se impondrá un ordenamiento lineal de los tipos de recursos en todos los procesos; es decir, si a un proceso le han sido asignados recursos de un tipo específico, en lo sucesivo sólo podrá pedir aquellos recursos que siguen en el ordenamiento.

Como vemos Havender presenta tres estrategias y no cuatro. Cada una de ellas, está diseñada para negar una de las condiciones necesarias. La primera de estas condiciones, esto es, que los procesos exijan el uso exclusivo de los recursos que requieren, es una condición que no es deseable impedir, porque específicamente queremos permitir la existencia de recursos no compartibles o dedicados.

Negación de la condición de espera

La primera de las estrategias requiere que los recursos que necesita un proceso sean pedidos de una sola vez. El sistema debe proporcionarlos según el principio de todo o nada. Si está disponible el conjunto de los recursos que necesita un proceso, entonces el sistema puede asignarle todos los recursos y éste seguir su ejecución. Si no está disponible alguno de ellos, el proceso debe esperar. Mientras espera no puede tener ningún recurso. Con esto se elimina la condición de espera y no puede ocurrir un interbloqueo.

Todo esto suena bien, pero puede llevar a un grave desperdicio de recursos. Supongamos que un proceso necesita diez unidades de un determinado recurso para su ejecución. Como debe solicitarlas todas antes de comenzar, los mantendrá en su poder durante toda su ejecución. Pudiera suceder, que el programa únicamente utilice estos recursos al principio de su ejecución, por tanto, los recursos están ociosos el resto del tiempo.

Dividir el programa en varios pasos que se ejecuten de manera relativamente independiente es una técnica empleada con frecuencia para conseguir una mejor utilización de los recursos en estas circunstancias. La asignación de recursos se controla por etapas. Esta solución reduce el desperdicio, pero implica mucho trabajo extra tanto en el diseño de las aplicaciones como en la ejecución.

Por otro lado, esta estrategia puede provocar un aplazamiento indefinido, pues los recursos requeridos pueden no estar disponibles todos al tiempo. El sistema podría, entonces, permitir que se fueran acumulando recursos hasta conseguir

todos los que necesita un proceso. Pero mientras se acumulan no se pueden asignar a otros procesos y volvemos a infrautilizarlos.

Negación de la condición de no apropiación

La segunda estrategia de Havender consiste en liberar los recursos que un proceso tiene asignados cuando se le niegan peticiones de recursos adicionales. De esta forma, se anula la condición de no apropiación. Los recursos se pueden quitar al proceso que los tiene antes de que termine su ejecución.

En este caso también existe un costo excesivo. Cuando un proceso libera recursos puede perder todo el trabajo realizado hasta ese momento. El costo puede parecer muy alto, pero la pregunta es : ¿con qué frecuencia ha de pagarse ese precio ? Si ocurre de tarde en tarde, entonces éste parece ser un buen método para prevenir el interbloqueo. Si, por el contrario, es muy frecuente, entonces el costo es sustancial y sus efectos demasiado perjudiciales (por ejemplo, para procesos de alta prioridad o plazo fijo).

Esta estrategia también adolece de aplazamiento indefinido. Un proceso puede aplazarse continuamente mientras pide y libera muchas veces los mismos recursos. Si esto ocurre, el sistema puede verse obligado a eliminar el proceso para que otros puedan ejecutarse.

Negación de la condición de espera circular

La tercera estrategia de Havender anula la posibilidad de una espera circular. Como todos los recursos tienen una numeración única y como los procesos deben pedir los recursos en un orden lineal ascendente, es imposible que se presente una espera circular (figura 5.4). Esta estrategia presenta las siguientes dificultades:

Los recursos deben pedirse en un orden ascendente por número de recursos. El número de recurso es asignado por la instalación y debe tener un tiempo de vida largo (meses o años). Si se agregan nuevos tipos de recursos, puede ser necesario reescribir los programas y los sistemas.

Lógicamente, cuando se asignan los números de recursos, éstos deben reflejar el orden normal en que los usan la mayoría de las tareas. Pero los procesos que necesiten los recursos en un orden diferente que el previsto por el sistema, los

deberán adquirir y conservar, quizá durante tiempo antes de utilizarlos realmente, lo que significa un desperdicio considerable.

Una de las metas más importantes de los sistemas operativos actuales es crear ambientes amables con el usuario. Los usuarios deben ser capaces de desarrollar sus aplicaciones sin tener en cuenta molestas restricciones de hardware y software. El ordenamiento lineal impide al usuario escribir sus códigos libremente.

3.5.- Nivel de implantación de estrategias.

Estrategias

Existen diversas estrategias frente a los interbloqueos, que se pueden agrupar en:

- Omisión
- Detección y recuperación
- Prevención
- Predicción

Que se detallan a continuación.

Omisión

Considera que la probabilidad de un interbloqueo es muy baja, de modo que se confía en que no se van a producir. Por sorprendente que parezca, los sistemas operativos modernos convencionales suelen aplicar esta estrategia. Por justificar la decisión de los fabricantes de sistemas operativos cabe incidir en que las estrategias de resolución y prevención de interbloqueos tienen un coste alto desde el punto de vista del consumo de recursos de procesamiento y memoria.

Detección y Recuperación

Esta estrategia permite la detección de una situación de interbloqueo y su consiguiente resolución. De entre las medidas de detección consideramos las siguientes:

Grafo de relación recursos-procesos: Consiste en la representación gráfica de los recursos asignados a los procesos y los recursos que dichos procesos requieren para finalizar su ejecución. Esta técnica se basa en recorrer el grafo

yendo de un nodo a otro, por lo que si se consigue volver al nodo de partida estaremos en un recorrido circular. Para que este tipo de error sea detectado usamos algoritmos de detección, se lanzan cuando se solicita un recurso ocupado, es decir, hay una nueva arista dentro de nuestro grafo y debemos comprobar que no da lugar a un recorrido cíclico.

Matrices de relación recursos-procesos: Consiste en la representación matricial de los recursos asignados a los procesos y los recursos que dichos procesos requieren para finalizar su ejecución. Se distinguen dos tipos, el método mediante matrices binarias de relación, y el método de detección matricial:

Matrices binarias de relación

Una matriz binaria de relación es aquella que representa una relación R entre dos conjuntos, en la cual el primero de estos dos tiene múltiples asignaciones a elementos del segundo.

El método consiste en, aplicando matrices binarias de relación, utilizar el cierre transitivo para determinar si algún proceso está relacionado consigo mismo a través de otros, señalando así la existencia de ciclos. El procedimiento sería:

1.- Formar la matriz de espera (W: P->R): Los procesos P están a la espera de recursos R.

2.- Formar la matriz de asignación (A: R->P): Los recursos R están asignados a procesos P.

3.- Formar la matriz de procesos a la espera de procesos (T: WxA): Producto cartesiano de ambas matrices.

4.- Hallar el cierre transitivo de la matriz T: Que se puede obtener, por ejemplo, aplicando el Algoritmo de Warshall (algoritmo de análisis sobre grafos para encontrar el camino mínimo entre todos los pares de vértices en una única ejecución). El algoritmo es el siguiente:

```
Warshall(T, n){
  for (k=1 to n){
    for (i=1 to n){
      for (j=1 to n){
```

Tij = Tij V (Tik Λ Tkj)

}

}

}

}

Donde n es la dimensión de la matriz T

5.- Si hay procesos que tengan un 1 en la diagonal principal, forman parte de algún ciclo.

Se trata de un método fácil de implementar, ya que solo se realizan operaciones con matrices y bucles, algo muy sencillo para una máquina. Sim embargo, tiene dos inconvenientes:

El número de operaciones a realizar es muy alto teniendo en cuenta el tamaño que pueden alcanzar las matrices de recursos

Solo se puede usar cuando solo existe una instancia de cada recurso.

Detección matricial

Método matricial que trata aquellos casos en los que existen múltiples instancias de un mismo recurso. Aísla grupos de procesos que no pueden proseguir la ejecución porque no pueden ver satisfechas sus peticiones pendientes. Usan un método iterativo que:

1.-Marca procesos cuyas peticiones puedan satisfacerse con el actual vector de recursos disponibles.

2.-Suma al vector de disponibles los recursos asignados a los procesos marcados.

3.-Si todos los procesos están marcados: no hay interbloqueo.

4.-Si en una iteración no se marcan procesos: los procesos que quedan están interbloqueados.

esolución de interbloqueo

Tras la detección de un interbloqueo, se pueden aplicar algunas de las siguientes estrategias para resolverlo:

Eliminación: El sistema operativo selecciona a uno de los procesos que forma parte del interbloqueo y elimina el ciclo acabando con la ejecución de dicho proceso, si no es suficiente se eliminarán procesos hasta que se rompa el ciclo. La selección del proceso se realiza en base a un cierto criterio, por ejemplo, aquel proceso que lleve menos tiempo en ejecución o aquel que sea más voraz consumiendo recursos.Sin embargo, de una manera u otra el trabajo realizado por el proceso se pierde, algo que en algunos casos resulta inadmisible, como en sistemas en tiempo real. Aunque parezca una medida drástica, es la empleada en sistemas operativos convencionales. Aplicar el criterio de selección y eliminar procesos cuando el número de procesos es relativamente bajo puede solucionar el interbloqueo, pero si se da un bloqueo de por ejemplo, centenares de procesos, es una situación prácticamente inmanejable.

Apropiación temporal: Se retira la asignación de un recurso a un proceso (durante el tiempo necesario) para deshacer el interbloqueo (hemos de asegurarnos de que el proceso no se desbloquea al romperse el interbloqueo). Por ejemplo, supongamos que el recurso es una impresora: podríamos retirarle la asignación a un proceso P1 cuando este terminase de imprimir una página, asignarle la impresora a otro proceso P2 y volver a asignársela a P1 cuando P2 haya terminado su ejecución. El problema es que este método solo es posible dependiendo de la naturaleza del proceso. Con frecuencia es imposible recuperarse de esta manera ya que los recursos no pueden ser apropiados.

Puntos de conformidad,sincronismo o checkpoints: Consiste en tomar una imagen del estado del proceso, ya sea periódicamente o a instancia del propio proceso, de manera que si se produce un interbloqueo se vuelve a un estado de la ejecución anterior. Son muy poco usados ya que tienen un elevado coste en memoria y existe la posibilidad de que un proceso permanezca indefinidamente sin progresar, y no todos los recursos permiten almacenar y recuperar su estado. Además, puede darse el caso de que el estado del proceso sea externo al sistema (Como en el caso de una conexión a Base de Datos)

Prevención

La prevención apunta a una serie de estrategias que eviten el interbloqueo. Concretamente, son cuatro las estrategias de prevención posibles en base a los principios que Coffman estableció como interbloqueo. Dichas estrategias son:

Supresión de exclusión mutua: Un proceso no puede tener acceso exclusivo a un recurso. No siempre es posible, y puede que lo único que haga sea cambiar el problema de sitio. Es una solución drástica, inviable. Por ejemplo, permitir que dos procesos usaran a la vez una impresora sería caótico.

Supresión de retención y espera (1ª estrategia de Havender): El proceso debe tener asignado todos los recursos necesarios al inicio y no liberarlos hasta que éste finalice, se consigue utilizando un mismo semáforo para todos los recursos necesarios por el proceso. Esto presenta un inconveniente: si un recurso sólo se utiliza al final, estará ocupado durante toda la ejecución, no permitiendo ser usado por otros procesos. El aprovechamiento de recursos puede mejorarse mediante una programación más elaborada, dividiendo la ejecución del proceso en distintas fases y gestionando los recursos para cada una de ellas. Sin embargo, muchos procesos no saben cuántos recursos necesitarán hasta que hayan empezado a ejecutarse. No obstante, esta estrategia presenta unos inconvenientes: la posibilidad de aplazamiento indefinido por parte de aquellos procesos que usan más recursos, que siempre cuando no le falta uno le falta otro, lo que concluye en un mal aprovechamiento de recursos, obligando a los procesos a solicitar los recursos antes de que les haga falta y atentando contra el objetivo eficiente que nos proponemos.

Supresión de no apropiación (2ª estrategia de Havender): Si un proceso está en ejecución y no puede obtener un recurso, dicho proceso libera todos los recursos que está usando y espera a que todos los que necesita estén disponibles. Es una estrategia optimista, usada cuando la probabilidad de que se produzca un interbloqueo en el sistema es baja. Problemas: se puede perder trabajo, además de presentar una carga extra la realización de peticiones.

Supresión de espera circular (3ª estrategia de Havender): Si todos los recursos comunes a varios procesos se solicitan siempre en el mismo orden no se producen interbloqueos. De esta manera, se ordenan los recursos y se solicitan en ese orden. Por ejemplo: tenemos un proceso P1 y otro P2, de manera que ambos hacen uso de los recursos X e Y. En el siguiente caso, no pedirían los recursos en el mismo orden:

P1: P2:

down(X) down(Y)

down(Y) down(X)

… …

up(Y) up(X)

up(X) up(Y)

Si se ejecuta la instrucción down(X) de P1, se conmuta a P2 y se ejecuta down(Y), se producirá un interbloqueo, ya que ambos estarán esperando a que el otro libere el recurso que necesitan. Sin embargo, si pedimos los recursos siempre en el mismo orden de la siguiente forma:

P1: P2:

down(X) down(X)

down(Y) down(Y)

… …

up(Y) up(Y)

up(X) up(X)

Se puede comprobar que es imposible que se dé un interbloqueo como en el caso anterior, ocurriendo lo mismo con cualquier número de procesos y recursos.

El principal inconveniente radica en que a veces, debido a la variedad y al número de recursos y procesos, es muy engorroso mantener un criterio de orden para todos los procesos de todo tipo.

Predicción

El sistema operativo observa la evolución que siguen los procesos, y predice una posible situación de interbloqueo. Si detecta una alta probabilidad de que suceda, adopta una trayectoria de ejecución nueva para los procesos involucrados de manera que se garantice que no va a suceder un interbloqueo.

Si tuviéramos de antemano información sobre cómo los procesos van a usar los recursos, tal vez podríamos forzar un entrelazado de las asignaciones que nunca llevase a interbloqueo. Es un ejemplo el algoritmo del banquero. El inconveniente de este tipo de técnicas es que son poco realistas, ya que en sistemas reales no tenemos forma de predecir a la perfección el futuro de accesos a recursos.

UNIDAD IV

CONTROL DE PROCESOS Y RECURSOS

Los programas del procesador son los encargados de la preparación de los programas de usuario para su ejecución, así como de la asignación de tiempos en el procesador. Un sistema operativo contiene un conjunto de programas cuya misión es la de asignar y controlar el almacenamiento en la memoria interna y externa de la computadora, Por otra parte, un sistema puede mantener en un mismo instante un gran número de usuarios y procesos, y éstos pueden estar solicitando y manejando continuamente archivos en memoria externa, en ocasiones hasta compartiéndolos, y por ello será necesario la existencia de una serie de programas en el sistema operativo que nos aseguren el correcto funcionamiento del almacenamiento secundario.

4.1.- Descriptor de procesos.

Los sistemas operativos multiprogramados necesitan del concepto de proceso. El sistema operativo debe entremezclar la ejecución de un número de procesos para maximizar la utilización de los recursos del ordenador. Al mismo tiempo, los sistemas de tiempo compartido deben proporcionar un tiempo de respuesta razonable. El sistema operativo debe asignar recursos a los procesos de acuerdo a una política específica (ciertas funciones o aplicaciones son de mayor prioridad), mientras impide los interbloqueos. Por último, el sistema operativo debe ofrecer un soporte para llevar a cabo la comunicación entre procesos.

El concepto de proceso es clave en los sistemas operativos modernos. La gestión del procesador mediante multiprogramación, revolucionó la concepción de los sistemas operativos, e introdujo el término proceso como elemento necesario para realizar dicha gestión. Por lo demás, este tema trata sobre la

definición de proceso, el estudio de sus propiedades, y la gestión que realiza el sistema operativo para crear la abstracción de proceso, aunque esto último se completará en el tema de planificación. Por último, descubriremos que el concepto de proceso encierra, en realidad, dos características potencialmente independientes: por un lado, es una unidad a la que se le asigna y posee recursos y, por otro, es una unidad planificable. Basándonos en esta distinción emprenderemos el estudio de los threads (hebra o hilo), o también llamados procesos ligeros.

Qué es un proceso

Hasta ahora hemos utilizado siempre el término programa. A partir de ahora distinguiremos entre programa y proceso. Un programa es una secuencia de instrucciones escrita en un lenguaje dado. Un proceso es una instancia de ejecución de un programa, caracterizado por su contador de programa, su palabra de estado, sus registros del procesador, su segmento de texto, pila y datos, etc. Un programa es un concepto estático, mientras que un proceso es un concepto dinámico. Es posible que un programa sea ejecutado por varios usuarios en un sistema multiusuario, por cada una de estas ejecuciones existirá un proceso, con su contador de programa, registros, etc. El sistema operativo necesita el concepto de proceso para poder gestionar el procesador mediante la técnica de multiprogramación o de tiempo compartido, de hecho, el proceso es la unidad planificable, o de asignación de la CPU.

Modelo de tres estados

Para poder manejar convenientemente una administración de procesador es necesario contar con un cierto juego de datos. Ese juego de datos será una tabla (BCP o en inglés PCB) en la cual se reflejará en qué estado se encuentra el proceso, por ejemplo, si está ejecutando o no. Los procesos, básicamente, se van a encontrar en este caso, en tres estados:

- Ejecutando.

- Listos para la ejecución.

- Bloqueados por alguna razón.

Sobre la base de estos estados se construye lo que se denomina Diagrama de Transición de Estado (DTE). Estar en la cola de Listos significa que el único

recurso que a ese proceso le está haciendo falta es el recurso procesador. O sea, una vez seleccionado de esta cola pasa al estado de Ejecución. Se tiene una transición al estado de Bloqueados cada vez que el proceso pida algún recurso. Una vez que ese requerimiento ha sido satisfecho, el proceso pasará al estado de Listo porque ya no necesita otra cosa más que el recurso procesador.

Para manejar esa cola de Listos se requiere de una tabla, y esa tabla debe tener una identificación de los procesos (Ver figura).

Como los listos pueden ser muchos, hará falta un puntero al primero de esa cola de listos, y posiblemente un enganche entre los siguientes en el mismo estado. Esta tabla contiene los Bloques de Control de Procesos. En este caso se agrupan los BCP en una Tabla de Bloques de Control de Procesos (TBCP).

Modelo de cinco estados

El modelo anterior de dos estados funcionaría bien con una cola FIFO y planificación por turno rotatorio para los procesos que no están en ejecución, si los procesos estuvieran siempre listos para ejecutar. En la realidad, los procesos utilizan datos para operar con ellos, y puede suceder que no se encuentren listos, o que se deba esperar algún suceso antes de continuar, como una operación de Entrada/Salida. Es por esto que se necesita un estado donde los procesos permanezcan bloqueados esperando hasta que puedan proseguir. Se divide entonces al estado No ejecución en dos estados: Listo y Bloqueado. Se agregan además un estado Nuevo y otro Terminado.

Los cinco estados de este diagrama son los siguientes:

Ejecución: el proceso está actualmente en ejecución.

Listo: el proceso está listo para ser ejecutado, sólo está esperando que el planificador así lo disponga.

Bloqueado: el proceso no puede ejecutar hasta que no se produzca cierto suceso, como una operación de Entrada/Salida.

Nuevo: El proceso recién fue creado y todavía no fue admitido por el sistema operativo. En general los procesos que se encuentran en este estado todavía no fueron cargados en la memoria principal.

Terminado: El proceso fue expulsado del grupo de procesos ejecutables, ya sea porque terminó o por algún fallo, como un error de protección, aritmético, etc.

Los nuevos estados Nuevo y Terminado son útiles para la de procesos. En este modelo los estados Bloqueado y Listo tienen ambos una cola de espera. Cuando un nuevo proceso es admitido por el sistema operativo, se sitúa en la cola de listos. A falta de un esquema de prioridades ésta puede ser una cola FIFO Los procesos suspendidos son mantenidos en una cola de bloqueados. Cuando se da un suceso se pasan a la cola de listos los procesos que esperaban por ese suceso. Si existe un esquema con diferentes niveles de prioridad de procesos es conveniente mantener varias colas de procesos listos, una para cada nivel de prioridad, lo que ayuda a determinar cuál es el proceso que más conviene ejecutar a continuación.

Durante su existencia un proceso pasa por una serie de estados discretos, siendo varias las circunstancias que pueden hacer que el mismo cambie de estado. Debido a ello se puede establecer una "Lista de Listos" para los procesos "listos" y una "Lista de Bloqueados" para los "bloqueados". La "Lista de Listos" se mantiene en orden prioritario y la "Lista de Bloqueados" está desordenada, ya que los procesos se desbloquean en el orden en que tienen lugar los eventos que están esperando. Al admitirse un trabajo en el sistema se crea un proceso equivalente y es insertado en la última parte de la "Lista de Listos". La asignación de la cpu al primer proceso de la "Lista de Listos" se denomina "Despacho", que es ejecutado por una entidad del Sistema Operativo llamada "Despachador". El "Bloqueo" es la única transición de estado iniciada por el propio proceso del usuario, puesto que las otras transiciones son iniciadas por entidades ajenas al proceso

Cuando el Sistema Operativo cambia la atención de la cpu entre los procesos, utiliza las áreas de preservación del PCB para mantener la información que necesita para reiniciar el proceso cuando consiga de nuevo la cpu. Los sistemas que administran los procesos deben poder crear, destruir, suspender, reanudar, cambiar la prioridad, bloquear, despertar y despachar un proceso. La "creación" de un proceso significa:

Dar nombre al proceso.

Insertar un proceso en la lista del sistema de procesos conocidos.

Determinar la prioridad inicial del proceso.

Crear el bloque de control del proceso.

Asignar los recursos iniciales del proceso.

Un proceso puede "crear un nuevo proceso", en cuyo caso el proceso creador se denomina "proceso padre" y el proceso creado "proceso hijo" y se obtiene una "estructura jerárquica de procesos". La "destrucción" de un proceso implica:

Borrarlo del sistema.

Devolver sus recursos al sistema.

Purgarlo de todas las listas o tablas del sistema.

Borrar su bloque de control de procesos.

Los eventos que llevan a un proceso a cambiar de un estado a otro son:

Null --> New: El nuevo proceso es creado para ejecutar un programa.

New --> Ready: El S.O. realiza esta transición cuando está preparado para comenzar a ejecutarse.

Ready --> Running: El S.O. hace éste cambio de estado cuando es tiempo de elegir un nuevo proceso a ejecutar.

Running --> Exit: Este cambio de estado se produce cuando el proceso actualmente en ejecución es finalizado o abortado.

Running ---> Ready: Esta transición puede ocurrir cuando se ha alcanzado el límite máximo de tiempo de ejecución ininterrumpida (técnica encontrada comúnmente en los S.O. con multiprogramación) o cuando, al trabajar con distintos niveles de prioridad, un proceso es reemplazado por otro de mayor prioridad.

Running --> Blocked: Esto ocurre cuando un proceso solicita algo por lo qu deba esperar. Este pedido es realizado, por lo general, en la forma de un system call (llamado de un programa en ejecución a un procedimiento que es parte del código del S.O.).

Blocked --> Ready: Este cambio tiene lugar al ocurrir el evento por el que estaba esperando un proceso.

Ready --> Exit: Esto ocurre cuando un proceso hijo es finalizado; ya sea por pedido del proceso padre o porque el padre en sí fue finalizado.

Blocked --> Exit: Idem ready --> exit.

Modelo de siete estados

Procesos suspendidos Una de las razones para implementar el estado Bloqueado era poder hacer que los procesos se puedan mantener esperando algún suceso, por ejemplo, una Entrada/Salida. Sin embargo, al ser mucho más lentas estas operaciones, puede suceder en nuestro modelo de cinco estados que todos los procesos en memoria estén esperando en el estado Bloqueado y que no haya más memoria disponible para nuevos procesos. Podría conseguirse más memoria, aunque es probable que esto sólo permita procesos más grandes y no necesariamente nuevos procesos. Además, hay un costo asociado a la memoria y de cualquier forma es probable que se llegaría al mismo estado con el tiempo. Otra solución es el intercambio. El intercambio se lleva a cabo moviendo una parte de un proceso o un proceso completo desde la memoria principal al disco, quedando en el estado Suspendido. Después del intercambio, se puede aceptar un nuevo proceso o traer a memoria un proceso suspendido anteriormente. El problema que se presenta ahora es que puede ser que si se decide traer a memoria un proceso que está en el estado Suspendido, el mismo todavía se encuentre bloqueado. Sólo convendría traerlo cuando ya está listo para ejecutar, esto implica que ya aconteció el suceso que estaba esperando cuando se bloqueó. Para tener esta diferenciación entre procesos suspendidos, ya sean listos como bloqueados, se utilizan cuatro estados: Listo, Bloqueado, Bloqueado y suspendido y Listo y suspendido.

Descripción de un proceso

De algún modo, debemos hacer una pregunta fundamental: ¿cuál es la manifestación física de un proceso? Como mínimo debe incluir un programa o conjunto de programas que sean ejecutados. Asociados a estos programas hay un conjunto de ubicaciones de datos para las variables locales y globales, y las constantes definidas. Así pues, un proceso constará, al menos, de la memoria suficiente para albergar los programas y los datos del proceso. Además, en la

ejecución de un programa entra en juego normalmente una pila, que se utiliza para llevar la cuenta de las llamadas a procedimientos y de los parámetros que se pasan entre los procedimientos. Por último, asociado a cada proceso hay una serie de atributos que utiliza el sistema operativo para el control del proceso. Estos atributos se recogen en una estructura de datos que se conoce como bloque de control de proceso (Process Control Block, PCB) o descriptor de proceso. A esta colección de programa, datos, pila y atributos se le llama imagen o entornos del proceso.

El bloque de control de proceso

El bloque de control de proceso es la estructura de datos central y más importante de un sistema operativo. Cada bloque de control de proceso contiene toda la información de un proceso que necesita un sistema operativo para su control. Estos bloques son leídos y/o modificados por casi todos los módulos de un sistema operativo, incluyendo aquellos que tienen que ver con la Planificación, la asignación de recursos, el tratamiento de inte y el análisis y supervisión del rendimiento. Puede decirse que el conjunto de los bloques de control de procesos define el estado del sistema operativo. El conjunto de todos los PCB's se guarda en una estructura del sistema operativo llamada tabla de procesos, la cual se puede implementar como un vector o una lista enlazada. La tabla de procesos reside en memoria principal, debido a su alta frecuencia de consulta.

En un sistema de multiprogramación, se requiere una gran cantidad de información de cada proceso para su administración. Sistemas distintos organizarán esta información de modo diferente. En general, se puede agrupar la información de los PCB's en tres categorías:

Identificación del proceso.

Información del estado del procesador.

Información de control del proceso.

Con respecto a la identificación del proceso, en casi todos los sistemas operativos se le asigna a cada proceso un identificador numérico único (ID). Este identificador nos servirá para localizarlo dentro de la tabla de procesos. Cuando se permite que los procesos creen otros procesos, se utilizan

identificadores para señalar al padre y a los descendientes de cada proceso. Además de estos, un proceso también puede tener asignado un identificador de usuario que indica a quién pertenece el proceso (UID).

El siguiente conjunto de información es la información de estado del procesador.

Básicamente, está formada por el contenido de los registros del procesador. Por supuesto, mientras el proceso está ejecutándose, la información está en los registros. Cuando se interrumpe el proceso, toda la información de los registros debe salvarse de forma que pueda restaurarse cuando el proceso reanude su ejecución. La naturaleza y número de registros involucrados depende del diseño del procesador. Normalmente, en el conjunto de registros se incluyen los registros visibles para el usuario, los registros de control y de estado (contador de programa y palabra de estado) y los punteros a pila.

A la tercera categoría general de información del bloque de control de proceso se le podría llamar información de control del proceso. Esta es la información adicional necesaria para que el sistema operativo controle y coordine los diferentes procesos activos. Como, por ejemplo, información de planificación y estado (estado del proceso, su prioridad, información de planificación, suceso), apuntadores (punteros) a estructuras de datos (los procesos que esperan en un semáforo), punteros a zonas de memoria del proceso, recursos controlados por el proceso (ficheros abiertos), etc.

Así pues, el PCB es la entidad que define un proceso en el sistema operativo. Dado que los PCB necesitan ser manejados con eficiencia por el sistema operativo, muchos ordenadores tienen un registro Hardware que siempre apunta hacia el PCB del proceso que se está ejecutando. A menudo existen instrucciones hardware que cargan en el PCB información sobre su entorno, y la recuperan con rapidez.

Operaciones con procesos

Los sistemas que administran procesos deben ser capaces de realizar ciertas operaciones sobre y con los procesos. Tales operaciones incluyen:

-crear y destruir un proceso

-suspender y reanudar un proceso

-cambiar la prioridad de un proceso

-bloquear y "desbloquear" un proceso

-Planificar un proceso (asignarle la CPU)

- permitir que un proceso se comunique con otro (a esto se denomina comunicación entre procesos, y se estudiará en el tema de procesos concurrentes).

Crear un proceso implica muchas operaciones, tales como:

-buscarle un identificador

-insertarlo en la tabla de procesos

-determinar la prioridad inicial del proceso

-crear el PCB

-asignar los recursos iniciales al proceso

Un proceso puede crear un nuevo proceso. Si lo hace, el proceso creador se denomina proceso padre, y el proceso creado, proceso hijo. Sólo se necesita un padre para crear un hijo. Tal creación origina una estructura jerárquica de procesos, en la cual cada hijo tiene sólo un padre, pero un padre puede tener muchos hijos. En el sistema operativo UNIX la llamada al sistema 'fork' crea un proceso hijo.

Destruir un proceso implica eliminarlo del sistema. Se le borra de las tablas o listas del sistema, sus recursos se devuelven al sistema y su PCB se borra (es decir, el espacio de memoria ocupado por su PCB se devuelve al espacio de memoria disponible). La destrucción de un proceso es más difícil cuando éste ha creado otros procesos. En algunos sistemas un proceso hijo se destruye automáticamente cuando su padre es destruido; en otros sistemas, los procesos creados son independientes de su padre y la destrucción de este último no tiene efecto sobre sus hijos.

Un proceso suspendido o bloqueado no puede proseguir sino hasta que lo reanuda otro proceso. La suspensión es una operación importante, y ha sido puesta en práctica de diferentes formas en diversos sistemas. La suspensión dura por lo normal sólo periodos breves. Muchas veces, el sistema efectúa las

suspensiones para eliminar temporalmente ciertos procesos, y así reducir la carga del sistema durante una situación de carga máxima. Cuando hay suspensiones largas se debe liberar los recursos del proceso. La decisión de liberar o no los recursos depende mucho de la naturaleza de cada recurso. La memoria principal debe ser liberada de inmediato cuando se suspenda un proceso; una unidad de cinta puede ser retenida brevemente por un proceso suspendido, pero debe ser liberada si el proceso se suspende por un periodo largo o indefinido. Reanudar (o activar) un proceso implica reiniciarlo a partir del punto en el que se suspendió.

Cambiar la prioridad de un proceso normalmente no implica más que modificar el valor de la prioridad en el PCB.

Suspensión y reanudación

Algunas líneas más arriba se presentaron los conceptos de suspensión y reanudación de un proceso. Estas operaciones son importantes por diversas razones.

Si un sistema está funcionando mal, y es probable que falle, se puede suspender los procesos activos para reanudarlos cuando se haya corregido el problema.

Un usuario que desconfíe de los resultados parciales de un proceso puede suspenderlo (en lugar de abortarlo) hasta que verifique si el proceso funciona correctamente o no.

Algunos procesos se pueden suspender como respuesta a las fluctuaciones (bajas y altas) a corto plazo de la carga del sistema, y reanudarse cuando las cargas vuelvan a niveles normales.

4.2.- Descriptor de recursos.

Un recurso de sistema es nada menos que cualquier parte funcional de un ordenador capaz de ser controlada y asignada por el sistema operativo, de manera tal que todo el hardware y software en el ordenador pueda trabajar como un conjunto.

Los recursos de sistema pueden ser empleados por cualquier usuario al abrir programas y aplicaciones, al igual que por servicios que usualmente inician de manera automática junto a un sistema operativo.

Los recursos de sistema pueden reducirse o agotarse completamente al contar con un carácter limitado.

Un limitado acceso a cualquier recurso de sistema llevará a reducir el desempeño general del ordenador y puede incluso desembocar en diferentes clases de errores.

Nota: muchas veces se definen a los recursos de sistema como recursos de hardware, recursos de ordenadores, o simplemente recursos. Los recursos no tienen relación de ningún tipo con un los Localizadores Uniformes de Recursos o Uniform Resource Locator (URL).

4.3.- Operaciones de procesos y recursos.

La manifestación de un proceso en Un sistema operativo es Un bloque de control de proceso (PCB). El es una estructura de datos que contiene cierta información importante acerca del proceso, incluyendo:

- Estado actual del proceso
- Identificación única del proceso
- Prioridad del proceso
- Apuntadores para localizar la memoria del proceso
- Apuntadores para asignar recursos
- Área para preservar registros

Así pues, el PCB es la entidad que define un proceso en el sistema operativo. Dado que los PCB necesitan ser manejados con eficiencia por el sistema operativo, muchos ordenadores tienen un registro hardware que siempre apunta hacia el PCB del proceso que se está ejecutando. A menudo existen instrucciones hardware que cargan en el PCB información sobre su entorno, y la recuperan con rapidez

4.4.- Interrupciones y procesos de entrada/salida.

Interrupción

Es un evento que altera la secuencia en que el procesador ejecuta las instrucciones.

Suspensión de un proceso, como la ejecución de un programa, originada por un suceso externo a dicho proceso y llevada a cabo de forma que el proceso pueda reanudarse.

Tipos de interrupción

Interrupciones de programa

Generadas por alguna condición que se produce como resultado de la ejecución de una instrucción, como el desbordamiento aritmético, la división por cero, el intento de ejecutar una instrucción ilegal de la máquina, o una referencia a una zona de memoria fuera del espacio permitido al usuario.

Interrupciones de reloj

Generadas por el reloj interno del sistema. Esto permite al sistema operativo llevar a cabo ciertas funciones con determinada regularidad, por ejemplo, el no permitir que ciertos procesos monopolicen el sistema.

Interrupciones de entrada/salida

Generadas por los controladores de entrada/salida, para indicar que una operación ha terminado normalmente o para indicar diversas condiciones de error.

Interrupciones de reinicio

Ocurre cuando se presiona el botón de reinicio o llega desde otro procesador la instrucción de Reinicio.

Interrupciones de verificación de la máquina.

Ocasionadas por el mal funcionamiento del hardware o por fallas tales como un corte de energía.

¿Qué pasa cuando ocurre una interrupción?

- El sistema operativo toma el control (el hardware pasa el control al sistema operativo).

- El sistema operativo guarda el estado del proceso interrumpido en el PCB del proceso.
- El sistema operativo analiza la interrupción y transfiere el control a la rutina adecuada para atenderla. Actualmente, el hardware se encarga de esto automáticamente.
- La rutina del manejador de interrupciones procesa la interrupción.
- Se restablece el estado del proceso interrumpido.
- Se ejecuta el proceso interrumpido (pasa a su estado Listo).

4.5.- Métodos de asignación del procesador.

La administración del procesador es, prácticamente el tema central de la multiprogramación. Esta administración involucra las distintas maneras a través de las cuales el sistema operativo comparte los recursos del procesador entre distintos procesos que están compitiendo por su uso. Esto implica directamente la multiprogramación y conlleva simultáneamente la sincronización de los mismos.

La idea de administrar el procesador eficientemente está enfocada en dos aspectos: el primero es la cantidad de procesos por unidad de tiempo que se pueden ejecutar en un sistema; y el segundo, el que importa más al usuario, es el tiempo de respuesta de esos procesos.

La idea principal de la administración del procesador tiene que ver con el tiempo que permanecerá un proceso en el procesador, ¿qué proceso corre en que momento?. En un Computador personal es más difícil encontrar el caso que un usuario necesite ejecutar dos procesos al mismo tiempo, enviar un correo y escribir un texto. Mientras que en un servidor esta clase de ejecuciones es constante. Para eso se necesita realizar planificación del procesador.

El Administrador de recursos del sistema de Windows puede administrar los recursos del procesador a través de los destinos del porcentaje de la CPU o las reglas de afinidad del procesador.

Destinos del porcentaje de la CPU

El método más sencillo de asignar recursos de procesador es asignar un destino del porcentaje de la CPU a cada grupo de procesos que se ha definido mediante un criterio coincidente del proceso. Este destino es el porcentaje del ancho de

banda disponible de la CPU que se garantiza como mínimo para el grupo de procesos.

Debido a que el Administrador de recursos del sistema de Windows garantiza una disponibilidad mínima del ancho de banda de la CPU, en lugar de limitar el uso de este ancho de banda, la CPU real que usa un grupo de procesos administrados puede superar la asignación mínima. La capacidad adicional de un grupo de procesos administrados que no esté usando la asignación mínima se volverá a asignar para procesar los grupos que necesitan más recursos.

Reglas de administración

Cuando se crea una directiva de asignación de recursos con restricciones de CPU, también se puede elegir una regla de administración para aplicar. Estas reglas de administración son similares a las directivas de asignación de recursos, pero cuando se aplican a una única asignación de recursos que forma parte de una directiva de asignación de recursos, dividen la CPU asignada entre todos los procesos coincidentes para la asignación de recursos.

Las reglas de administración incluyen:

Estándar (Predeterminado)

El Administrador de recursos del sistema de Windows no intenta controlar la forma en que se divide la CPU asignada entre los procesos coincidentes. Cuando se selecciona esta regla de administración, se pueden subasignar recursos a procesos coincidentes mediante criterios coincidentes del proceso adicionales. Para obtener más información, vea la sección sobre subasignación más adelante en este tema.

Por ejemplo, es posible que un proceso coincidente consuma todo el ancho de banda asignado de la CPU. El Administrador de recursos del sistema de Windows no administra este consumo, de modo que puede afectar a un segundo proceso.

Igual por proceso

El ancho de banda disponible de la CPU se divide de forma homogénea entre los procesos coincidentes. Cuando se selecciona esta regla de administración, no se admite la subasignación.

Por ejemplo, si dos procesos coincidentes consumen el 100 por ciento del ancho de banda asignado de la CPU, el Administrador de recursos del sistema de Windows reducirá la prioridad del proceso que supera el 50 por ciento del uso de la CPU.

Igual por usuario

El ancho de banda de la CPU lo comparten de forma equitativa los grupos de procesos coincidentes que ejecuta un usuario individual. Cuando se selecciona esta regla de administración, no se admite la subasignación.

Por ejemplo, si dos usuarios ejecutan varias aplicaciones que consumen el 100 por ciento del ancho de banda asignado de la CPU, el Administrador de recursos del sistema de Windows reducirá la prioridad de los procesos que ejecuta el usuario que supera el 50 por ciento del uso de la CPU.

Igual por sesión

En un servidor Host de sesión de Escritorio remoto, el ancho de banda disponible de la CPU lo comparten de forma equitativa los procesos coincidentes que se ejecutan en cada sesión de Servicios de Escritorio remoto. Cuando se selecciona esta regla de administración, no se admite la subasignación.

Por ejemplo, si dos usuarios conectados a un servidor Host de sesión de Escritorio remoto consumen el 100% del ancho de banda asignado de la CPU, el Administrador de recursos del sistema de Windows reducirá la prioridad de los procesos que se ejecutan en la sesión de Servicios de Escritorio remoto que supera el 50% del uso de la CPU.

Subasignación

Las asignaciones de destino de porcentaje de la CPU se pueden dividir en subasignaciones. Una subasignación asigna recursos que se calculan como un porcentaje de los recursos asignados por la asignación de recursos principal. Esta subasignación compara un criterio coincidente del proceso diferente del de la asignación de recursos principal.

Las subasignaciones tienen precedencia sobre la directiva de asignación de recursos predeterminada. Para obtener más información, vea Subasignar recursos.

Administración predeterminada: Igual por proceso

La directiva predeterminada para la administración de recursos entre los procesos de un solo grupo de procesos es la directiva integrada Igual por proceso. Con esta directiva:

El ancho de banda disponible de la CPU se divide de forma equitativa entre los procesos que identifica el criterio coincidente del proceso.

La protección de procesos fuera de control está habilitada de forma predeterminada.

Iniciar el Administrador de recursos del sistema de Windows sin la configuración adicional aplica esta directiva a todos los procesos que se pueden administrar y que se ejecutan en un servidor administrado.

Esta directiva predeterminada se puede cambiar mediante la edición de las propiedades del Administrador de recursos del sistema de Windows. Debe habilitar la Directiva de asignación de recursos actual (si el Calendario está deshabilitado) o deshabilitar la Directiva predeterminada de calendario (si el Calendario está habilitado).

Afinidad de procesador

Además de especificar un destino del porcentaje de la CPU, los procesos coincidentes se pueden vincular con procesadores específicos en sistemas multiprocesador. Este método puede dividir eficazmente los recursos del servidor entre unos pocos procesos que coinciden con el criterio, pero se debe ser precavido cuando se usa la afinidad de procesador con un gran número de criterios coincidentes del proceso. El Administrador de recursos del sistema de Windows solo considerará el estado de ese único procesador cuando se calculan los recursos disponibles para un proceso con afinidad, de forma que los recursos del procesador se pueden sobreasignar cuando el sistema experimenta una carga intensiva.

En algunas ocasiones, el ancho de banda disponible de la CPU puede ser menor del esperado. Esto reducirá el ancho de banda asignado de la CPU para los procesos coincidentes y puede hacer que respondan más lentamente de lo esperado. Esto se puede producir cuando:

El número de procesadores con el que tiene afinidad el grupo de procesos es demasiado pequeño.

Los procesos sin afinidad usan un procesador que está restringido para otro grupo de procesos.

Los criterios coincidentes del proceso no pueden comprobar si hay un conflicto de asignación entre los procesos con afinidad.

4.6.- Job Scheduler (Despachador).

Despachador (Scheduler).

Su misión es asignar los procesadores centrales a los procesos. Es llamado cuando un proceso no puede seguir o puede emplearse mejor en otra parte. Se activa en las situaciones siguientes:

1. Tras una interrupción externa ha cambiado el estado de un proceso.

2. Después de que un extra código provoque imposibilidad de seguir la ejecución de un proceso.

3. Tras una señal de error se ha suspendido el proceso hasta que no se haya tratado este.

Estas son causas especiales de interrupción, esto es, todas ellas consisten en interrupciones que alteran el estado de algún proceso.

SCHEDULER

Componente del sistema operativo responsable de decidir quien hará uso de la CPU.

FUNCIONES

- El despachador examina la prioridad de los procesos.

- Controla los recursos de una computadora y los asigna entre los usuarios.

- Permite a los usuarios correr sus programas.

- Controla los dispositivos de periféricos conectados a la máquina.

-Cambio de contexto.

- Cambio a modo usuario.

Para determinar el proceso más adecuado para ser ejecutado se ordenan los procesos ejecutables de acuerdo con algún criterio de prioridad. Las prioridades de los procesos vienen dadas y por tanto no son misión del despachador.

Se utilizará una cola ordenada de modo que en cabeza de la cola este el proceso más adecuado, así la misión del despachador es la de ejecutar el primer proceso de la cola que no esté siendo ejecutado.

Se puede tener en lugar de una cola, más de una. Por ejemplo, tres. Una para aquellos procesos a los que se les permite dos segundos consecutivos de CPU. Otra para los que se les permiten 0,25 y otra para los que se les permite solo 0,02 segundos.

Cada cola se sirve con el criterio de "el primero en llegar es el primero en ser servido". Las colas con menor tiempo tienen mayor prioridad. Los procesos se colocan inicialmente en la cola de menor tiempo. Si consume todo el tiempo asignado se transfiere a la siguiente cola en prioridad, así sucesivamente.

Se consigue así que los procesos que consumen menos tiempo de procesador, reciban un proceso rápido, mientras que las tareas habituales uno más largo.

OBJETIVO PRINCIPAL DEL DESPACHADOR

Optimizar la eficiencia del sistema, de acuerdo con criterios considerados importantes para el ambiente del sistema

BIBLIOGRAFÍA BÁSICA Y COMPLEMENTARIA:

Sistemas operativos en red Núñez Rodríguez, M.ª de las Mercedes · Nieto Santos, Silvia

Sistemas Operativos En Red García Cervigon Hurtado, Alfonso / Alegre Ramos, María Del Pilar