

Frank Mendoza Hernández

SISTEMAS OPERATIVOS

Guía de estudio



UNED

UNIVERSIDAD ESTATAL A DISTANCIA

Institución Benemérita de la Educación y la Cultura



Producción académica
y asesoría metodológica

Mario Marín Romero

Diagramación

Mario Marín Romero

Encargada de cátedra

Karol Castro Chaves

Esta guía de estudio fue confeccionada en la UNED, en el año 2013, para ser utilizada en la asignatura “Sistemas Operativos”, código 881, que se imparte en el programa de Ingeniería Informática.

Universidad Estatal a Distancia

Vicerrectoría Académica

Escuela de Ciencias Exactas y Naturales



PRESENTACIÓN

Salvo algunas excepciones nuestro diario vivir gira en torno a las computadoras, trámites bancarios, hospitalarios, legales, semáforos, itinerarios de velo, los vehículos, aviones, maquinaria, etc. En el hogar prácticamente los todos los artefactos eléctricos poseen microprocesadores que nos alivian la carga pero que nos hacen compu-dependientes.

Si va a un banco y no hay sistema porque se “cayó” todo se paraliza ya los cajeros no funcionan, no se pueden pagar recibos y menos sacar dinero de los cajeros. Todo esto obliga a que los diseñadores de computadores y desarrolladores de sistemas se esmeren porque los sistemas (hardware y software) sean eficientes y eficaces, que las caídas de sistema sean mínimas, que las comunicaciones se mantengan constantes y con niveles altos de respuesta, esto es lo que exige la dinámica mundial en todas sus operaciones, pero no siempre fue así.

Las primeras computadoras eran máquinas inmensas operadas desde una consola. Solo se conocía el hardware. El programador escribía sus programas y, luego, desde la consola, la computadora era controlada para que se ejecutara ese programa.

Posteriormente, con el transcurrir de los años, se crearon software y hardware adicionales. Por ejemplo, se diseñaron ensambladores, cargadores y ligadores, que vinieron a ayudar en las tareas de programación. También se diseñaron bibliotecas de funciones comunes. Como su nombre lo indica, estas bibliotecas servían para copiar una función en un nuevo programa; los programadores no tenían que crear esas funciones.

Además, cada dispositivo de entrada / salida traía consigo sus propias características. Por lo tanto, era necesario tener cuidado con su programación, que para cada dispositivo era diferente. Además, para cada dispositivo debía desarrollarse una subrutina especial, llamada manejador de dispositivo. Por ejemplo, una tarea sencilla como leer un carácter de un lector de cinta de papel, podría requerir complicadas secuencias de operaciones específicas para dicho dispositivo.

Un tiempo después, se crearon los compiladores y otros leguajes, que brindaron ayuda en las tareas de programación, pero con ello el funcionamiento del computador se volvió más complejo.

Un sistema operativo (SO) es un programa que actúa como intermediario entre el usuario y el hardware de un computador. En sí, es un programa de computadora, pero muy especial, quizá el más complejo e importante en una computadora.

El sistema operativo tiene como objetivo principal lograr que el sistema de computación se use de manera cómoda, y que el hardware del computador se emplee eficientemente.

El sistema operativo despierta a la computadora y hace que reconozca a la unidad de procesamiento central (CPU), la memoria, el teclado, el sistema de video y las unidades de disco. Además, facilita la comunicación de los usuarios con la computadora y sirve de plataforma para que se corran los programas de aplicación.

El sistema operativo posee cuatro tareas principales, que son proporcionar una interfaz de línea de comandos o una interfaz gráfica al usuario, administrar los dispositivos de hardware de la computadora, administrar y mantener los sistemas de archivos de disco y apoyar a otros programas.

Esta guía de estudio está orientada a cubrir los principales aspectos de cada tema, con miras a contribuir efectivamente en su aprendizaje. Por ello, se constituye en un material complementario que, en ningún caso, sustituye al libro de texto. Por lo anterior, es responsabilidad del estudiante estudiar a fondo los diferentes tópicos, y aclarar dudas o inquietudes en las tutorías presenciales que brinda este curso.

Comprende ocho temas de gran importancia para la formación académica del estudiante. El orden en que se presentan estos temas está de acuerdo con su nivel de dificultad. Por lo tanto, el estudiante debe seguir al detalle esta guía para lograr un buen entendimiento de la materia.



CONTENIDOS

PRESENTACIÓN	iii
OBJETIVOS.....	x
 TEMA 1. CONCEPTOS BÁSICOS	 1
Objetivos de aprendizaje.....	2
Guía de lectura.....	2
Conceptos clave	3
Aspectos sobresalientes	3
1.1. ¿Qué es un sistema operativo?	3
1.2. Historia de los sistemas operativos	6
1.3. Revisión del hardware de una computadora	6
1.4. Tipos de sistemas operativos	8
1.5. Conceptos de los sistemas operativos	9
1.6. Llamadas al Sistema	9
1.7 Estructura de un sistema operativo	10
Ejercicios de autoevaluación.....	10

TEMA 2. PROCESOS E HILOS.....	11
Objetivos de aprendizaje.....	12
Guía de lectura.....	12
Conceptos clave	13
Aspectos sobresalientes.....	13
2.1. Procesos	13
2.2. Hilos.....	16
2.3. Comunicación entre procesos	17
2.4. Planificación	19
2.5 Problemas clásicos de comunicación entre procesos (IPC).....	22
Ejercicios de autoevaluación.....	22
 TEMA 3. ADMINISTRACIÓN DE MEMORIA.....	 23
Objetivos de aprendizaje.....	24
Guía de lectura.....	24
Conceptos clave	25
Aspectos sobresalientes.....	25
3.1. Sin abstracción de memoria	26
3.2. Una abstracción de memoria: espacio de direcciones	27
3.3. Memoria virtual	28
3.4. Algoritmos de reemplazo de páginas	28
3.5. Cuestiones de diseño para los sistemas de paginación.....	29
3.6. Cuestión de implementación	29
3.7. Segmentación	30
Ejercicios de autoevaluación.....	30

TEMA 4. SISTEMAS DE ARCHIVOS.....	31
Objetivos de aprendizaje.....	32
Guía de lectura.....	32
Conceptos clave	33
Aspectos sobresalientes	33
4.1. Archivos	34
4.2. Directorios.....	35
4.3. Implementación de Sistemas de Archivos.....	36
4.4. Administración y Optimización de Sistemas de Archivos.....	37
Ejercicios de autoevaluación.....	38
 TEMA 5. ENTRADA / SALIDA	 39
Objetivos de aprendizaje.....	40
Guía de lectura.....	40
Conceptos clave	41
Aspectos sobresalientes	41
5.1. Principios de hardware de E/S	42
5.2. Fundamentos de software.....	43
5.3. Capas de software de E/S.....	44
5.4. Discos.....	45
5.5. Relojes.....	47
5.6. Interfaces de usuario, clientes delgados y administración de energía	48
Ejercicios de autoevaluación.....	48

TEMA 6. INTERBLOQUEOS.....	49
Objetivos de aprendizaje.....	50
Guía de lectura.....	50
Conceptos clave	51
Aspectos sobresalientes.....	51
6.1. Recursos.....	52
6.2. Introducción a los interbloques	53
6.3. Detección y recuperación de un interbloqueo	54
6.4. Cómo evitar interbloques.....	55
6.5. Como prevenir interbloques	55
Ejercicios de autoevaluación.....	56
 TEMA 7. SEGURIDAD.....	 57
Objetivos de aprendizaje.....	58
Guía de lectura.....	58
Conceptos clave	59
Aspectos sobresalientes.....	59
7.1. El entorno de la seguridad	59
7.2. Fundamentos de la criptografía.....	60
7.3. Mecanismos de protección.....	63
7.4. Ataques desde el interior	64
Ejercicios de autoevaluación.....	64

TEMA 8. DISEÑO DE SISTEMAS OPERATIVOS	65
Objetivos de aprendizaje.....	66
Guía de lectura.....	66
Conceptos clave	67
Aspectos sobresalientes	67
8.1. La naturaleza del problema de diseño.....	67
8.2. Diseño de interfases.....	70
8.3. Implementación	72
8.4. Rendimiento.....	73
Ejercicios de autoevaluación.....	74
 RESPUESTAS A LOS EJERCICIOS DE AUTOEVALUACIÓN.....	75
GLOSARIO	91
LISTA DE REFERENCIAS.....	95



OBJETIVOS

Objetivo general

- ✓ Brindar al estudiante los elementos y tareas fundamentales que conforman e intervienen en un sistema operativo para éste sea eficiente, eficaz y amigable con el usuario.

Objetivos específicos

- Introducir al estudiante en los conceptos, problemática y técnicas de solución a los problemas de diseño en los sistemas operativos y procesos.
- Enfrentar al estudiante con la resolución de problemas de programación que requieran de la aplicación práctica de los conocimientos adquiridos sobre sistemas operativos.
- Propiciar la capacidad de análisis del estudiante, mediante el planteamiento de problemas específicos de los sistemas operativos, en los que aplique sus conocimientos al respecto.
- Describir el sistema de archivos en los sistemas operativos.
- Desarrollar los sistemas de administración de memoria en los sistemas operativos.
- Analizar los procesos de: entradas, salidas, bloqueos y sistemas operativos distribuidos.
- Desarrollar el problema del bloqueo en el sistema operativo.
- Relacionar el sistema operativo con el hardware.

CONCEPTOS BÁSICOS

1

Sumario

- *¿Qué es un sistema operativo?*
- *Historia de los sistemas operativos*
- *Revisión del hardware de una computadora*
- *Todo tipo de sistemas operativos*
- *Conceptos de los sistemas operativos*
- *Llamadas al Sistema*
- *Estructura de un sistema operativo*



Objetivos de aprendizaje

Al finalizar el estudio de este tema, entre otras habilidades, usted será capaz de:

- ✓ Explicar los componentes básicos de un sistema de cómputo.
- ✓ Explicar la evolución que han tenido los sistemas operativos a través del desarrollo informático.
- ✓ Identificar los diferentes sistemas operativos que existen.
- ✓ Explicar los conceptos de hardware.
- ✓ Explicar los conceptos básicos de los sistemas operativos y las llamadas al sistema.
- ✓ Explicar la estructura de un sistema operativo.

Guía de lectura

Antes de realizar continuar con la lectura de esta guía, estudie detalladamente los siguientes apartados en el libro de texto:

- a. ¿Qué es un sistema operativo?, página 3.
- b. Historia de los sistemas operativos, página 7.
- c. Revisión del Hardware de computadora, página 19.
 - i. Procesadores, página 19.
 - ii. Memoria, página 23.
- d. Tipos de sistemas operativos, página 33.
- e. Conceptos de los sistemas operativos, página 37.
- f. Llamadas al sistema, página 49.
- g. Estructura de un sistema operativo, página 62.



Conceptos clave

Al finalizar el estudio de este capítulo usted deberá tener claros los siguientes conceptos:

- ✓ Sistema operativo
- ✓ Llamadas al sistema
- ✓ Kernel
- ✓ Modo Kernel
- ✓ Modo usuario
- ✓ Tipos de Memoria
- ✓ Memoria caché
- ✓ Buses
- ✓ Shell
- ✓ Dispositivos de Entrada y Salida
- ✓ Procesadores
- ✓ Estructura de un Sistema Operativo

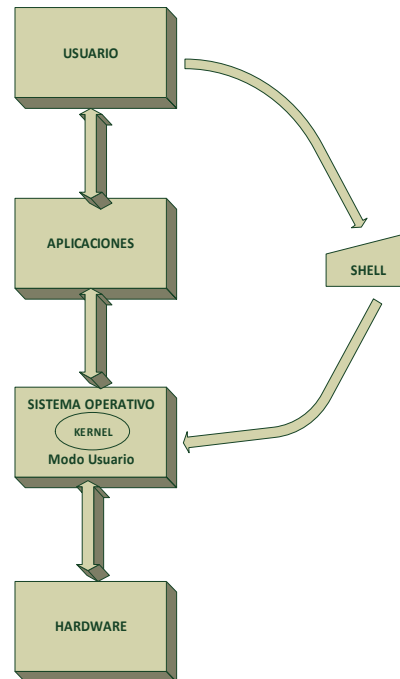
Aspectos sobresalientes

1.1. ¿Qué es un sistema operativo?

El sistema operativo (SO) es el encargado de administrar, de manera eficiente y eficaz, los recursos, tanto de software como de hardware, que estarán a disposición del usuario de manera directa (Shell) o a través de una aplicación. La figura 1.1 que se muestra a continuación, le permitirá tener una visión gráfica y rápida del lugar que ocupa el SO y del papel que juega en un equipo de cómputo.

Figura 1.1

UBICACIÓN DEL SISTEMA OPERATIVO EN UN SISTEMA DE CÓMPUTO



Conforme avance en la lectura del libro de texto se irá profundizando en diversos temas que le ayudarán a comprender mejor el papel que juega el SO en el uso constante de los recursos informáticos. El primer capítulo del libro, por ser introductorio, brinda un vistazo general de cada uno de los capítulos restantes que conforman el libro. En esta Guía de Estudio se hará referencia solamente a los temas de los capítulos que son objeto de estudio en el curso Sistemas Operativos (capítulos 1, 2, 3, 4, 5, 6, 8, 9 y 13).

Como puede observar en la figura 1, el SO se encuentra entre el hardware y el software de aplicaciones, o sea, es quien coordina ambos conceptos para dar una respuesta satisfactoria al usuario. Las aplicaciones de software son todos aquellos programas o aplicaciones que le permiten al usuario realizar una función, por ejemplo: procesadores de texto, hojas de cálculo, programas especializados de ingeniería y de diseño gráfico, programas desarrollados por programadores para una tarea específica, tales como: cálculos contables, manejo de inventarios, planillas, cuentas bancarias, administración de pólizas, entre otros.

El SO, en palabras simples, es el director de orquesta; es quien relaciona los programas de aplicaciones con el hardware que esa aplicación va a requerir para llevar a cabo la tarea.



El SO, que es un software, tiene dos formas de ejecutarse: en modo **Kernel** (o **Supervisor**) o en modo **usuario**. El Kernel es el corazón del SO y el usuario no tiene acceso a él. En el modo Kernel el sistema tiene acceso irrestricto a todos los recursos de hardware y decide cuál instrucción debe ejecutar, esto sin la intervención del usuario. En el modo de usuario solamente se ponen a disposición del operador un limitado número de instrucciones de máquina. Tanenbaum (2009) afirma que “las instrucciones que afectan el control de la máquina o que se encargan de la E/S (entrada/salida) están prohibidas para los programas en modo usuario” (p. 2).

Tanenbaum (2009) afirma que “los sistemas operativos realizan dos funciones básicas que no están relacionadas: proporcionar a los programadores de aplicaciones (y a los programas de aplicaciones, naturalmente) un conjunto abstracto de recursos simples, en vez de complejos conjuntos de hardware; y administrar estos recursos de hardware” (p.3).

Lo que esta afirmación quiere decir es que el SO puede verse como una **máquina extendida** y como un **administrador de recursos**. Antes para poder crear un subdirectorio o carpeta, copiar, mover o borrar archivos, formatear un disco y otras tareas básicas, se debía escribir largas y tediosas hileras de palabras propias del SO para que éste realizara la labor; esto se hacía mediante un lenguaje de programación llamado “Ensamblador”. Luego vinieron los SHELL que permitían, con menos palabras o caracteres, decirle al SO lo que se quería. Por último vinieron las interfaces gráficas, las cuales con un solo *clic* le pueden indicar al SO qué queremos hacer. Estas nuevas facilidades que se le dan al usuario para ejecutar instrucciones del SO es lo que se llama máquina extendida.

El SO, como un administrador de recursos, decide cuáles recursos utilizará y en qué momento. Hace uso de *buffers*, que son áreas de memoria de almacenamiento temporal para lectura o escritura de datos.

El SO administra el espacio en la memoria principal (RAM) y decide, no solo cuál proceso o programa ejecuta, sino también cuál instrucción de ese programa.

El SO puede decidir si empieza a ejecutar otro programa, mientras se están leyendo datos del disco duro para otro programa. Esto se conoce como pseudoparalelismo, pues el usuario tiene la sensación que se están ejecutando dos o más programas de manera simultánea, cuando en realidad lo que está sucediendo es que en los “tiempos muertos” de un programa, el SO aprovecha el tiempo para ejecutar otro programa. En otras palabras, el SO puede mantener varios programas en memoria principal a la vez, pero sólo ejecuta uno y



en los tiempos en que el programa debe leer más datos o escribir datos en disco, lo cual es una tarea que demanda gran cantidad de tiempo, el SO aprovecha y ejecuta otro programa que tiene esperando en la memoria.

1.2. Historia de los sistemas operativos

A través de la historia hemos tenido computadores que trabajaban de manera aislada sin comunicarse con otros computadores, pero a partir de la cuarta generación, iniciando los años 80, se dio un “bum” con los computadores personales, las redes de computadoras, las interfaces gráficas en donde aparecieron las versiones de Windows y Apple. Esta historia aún no acaba y vemos como los sistemas operativos están presentes en cuanto aparato electrónico tengamos, mejor ejemplo de ello, los teléfonos móviles o celulares y las “tablets”. Un vistazo de la historia de las computadoras la puede leer en las páginas 7 a la 18 del libro de texto.

1.3. Revisión del hardware de una computadora

Lo más cercano a nosotros y que podemos tomar como base para entender mejor los componentes de una computadora es una computadora personal. En ella encontramos: un monitor, un teclado, un *mouse*, una impresora, quizá un *scanner*, memorias secundarias como disco duro, lectoras de CD y DVD, memorias *flash* (llaves maya), memoria principal, otros dispositivos de entrada como teclados musicales, cámaras de vídeo o fotográficas, etc. Todos estos componentes están conectados entre sí a través de cables llamados buses, los cuales se llaman así porque por allí transitan los datos y las instrucciones que van de un dispositivo a otro. Esta estructura es la parte de la computadora llamada hardware. Para comprender mejor esta afirmación observe nuevamente la ubicación del Hardware en la ilustración 1.1.

1.3.1. Procesadores

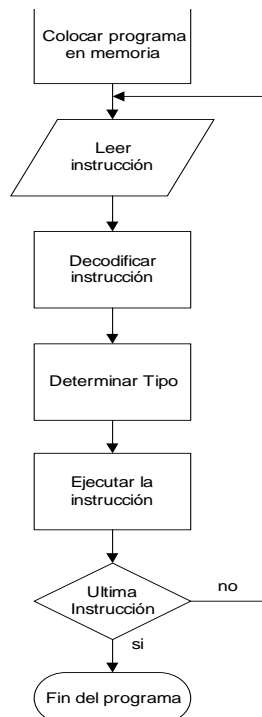
El “cerebro” de la computadora es la Unidad Central de Proceso (CPU o UCP en español). El ciclo que ejecuta la CPU con las instrucciones de un programa es el siguiente:

- a. Se obtiene una instrucción de la memoria.
- b. Se decodifica la instrucción.
- c. Se determina su tipo y operando.
- d. Se ejecuta la instrucción.
- e. Vuelve al punto a.

Este ciclo se repite desde la primera hasta la última instrucción de un programa que ha sido alojado en la memoria principal, como lo puede observar en la figura 1.2.

Figura 1.2

Ciclo del Procesador





Para que el CPU pueda desempeñarse adecuadamente requiere de una serie de registros, llamados también variables. Es importante que usted tenga claro que las variables del SO y las variables propias de un programa no son lo mismo.

Según Tanenbaum (2009), las tres variables más comunes del SO son:

1. Contador de Programa (*program counter*)
2. Apuntador de pila (*stack pointer*)
3. PSW

Un concepto, al que debe poner atención es el concepto de “llamadas al sistema (*system call*)”; este concepto frecuentemente es utilizado por los SO.

1.3.2. Memoria

La memoria es el segundo componente más importante de un computador. Hay diferentes tipos de memoria y diferentes formas de administrarla. Es importante que comprenda que el CPU utiliza diferentes tipos de memoria para realizar sus operaciones y para ejecutar instrucciones; según la operación que se ejecuta así será el tipo de memoria que se utilice.

En la figura 1.9 del libro de texto (página 23) puede observar cómo la memoria puede verse como una jerarquía de capas. Las capas superiores son las de mayor rapidez de acceso aunque también son las más pequeñas en capacidad. Estos conceptos debe estudiarlos detenidamente en el libro de texto, ya que son de suma importancia.

1.4. Tipos de sistemas operativos

A continuación se detallan una serie de tipos de sistemas operativos a los cuales usted debe prestar atención:

1. Sistemas operativos de Mainframe
2. Sistemas operativos de servidor
3. Sistemas operativos multiprocesador
4. Sistemas operativos de computador personal
5. Sistemas operativos de tiempo real
6. Sistemas operativos integrados
7. Los sistemas operativos de tarjeta inteligente.



1.5. Conceptos de los sistemas operativos

Los sistemas operativos, como hemos visto, contienen una serie de conceptos básicos y abstracciones. Entre ellos tenemos los siguientes:

1. Procesos o programas en ejecución
2. Espacio de direcciones
3. Sistema de Archivos
4. Dispositivos de Entrada y Salida.
5. Protección de los datos
6. Shell

Para ampliar estos conceptos refiérase a las páginas de la 37 a la 49 del libro de texto.

1.6. Llamadas al Sistema

Las llamadas al sistema son interrupciones que se hacen en modo de usuario; es una interfaz entre el usuario y el sistema operativo. Por ejemplo, cuando usted está programando y da una orden de lectura de datos a un archivo específico, seguramente usted le ha dicho al programa en qué subdirectorios se encuentra su archivo desde el punto de vista conceptual, pero desde el punto de vista físico usted no sabe en qué parte del disco duro están almacenados los datos, en qué pista o en qué cara del disco duro están. Usted da la orden de leer el archivo, esto es modo usuario, el SO toma esa orden (modo kernel) y busca el archivo y trae a la memoria la cantidad de datos necesarios y los pone a disposición del procesador para ser procesados. Puede ser que el archivo esté almacenado en pedazos en diferentes partes del disco pero usted lo ve como un solo archivo; asimismo, cuando usted le da la orden de grabar los datos ya procesados, el SO almacena los datos en el disco pero usted no sabe en qué parte del disco los grabó y si lo grabó en una sola pista o en diferentes pistas.

Puede ser que cuando un proceso se está ejecutando se haga una llamada al sistema desde otro proceso, entonces, el sistema evalúa si la llamada tiene prioridad y si la tiene le da una pausa al proceso para ejecutar el otro, una vez que ha finalizado el segundo proceso o éste entra en pausa, el SO aprovecha y continúa con el primer proceso que se estaba ejecutando.



Existen diferentes tipos de llamadas al sistema:

1. Llamadas al sistema para administración de procesos
2. Llamadas al sistema para administración de archivos
3. Llamada al sistema para administración de directorios
4. Diversas llamadas al sistema.

Para ampliar los conceptos anteriores, puede referirse a la sección 1.6 del libro de texto (páginas de la 49 a la 61).

1.7 Estructura de un sistema operativo

Dentro de la estructura de un sistema operativo, se encuentran:

1. El sistema monolítico
2. El sistema en capas
3. Microkernels
4. Modelo Cliente Servidor
5. Máquinas Virtuales
6. Exokernels

Para ampliar los conceptos anteriormente descritos, puede referirse a la sección 1.7 del libro de texto (páginas de la 62 a la 72).

Ejercicios de autoevaluación

A continuación, se presentan ejercicios que tienen como objetivo que usted pueda evaluar su aprendizaje sobre cada uno de los puntos contenidos en el desarrollo del capítulo 1.

Para el capítulo 1, realice los ejercicios 1, 2, 14, 17, 22 y 24, que se encuentran en las páginas de la 79 a la 82 del libro de texto.

PROCESOS E HILOS

2

Sumario

- *Procesos*
- *Hilos*
- *Comunicación entre procesos*
- *Planificación*
- *Problemas clásicos de comunicación entre procesos (IPC)*



Objetivos de aprendizaje

Al finalizar el estudio de este tema, entre otras habilidades, usted será capaz de:

- ✓ Explicar qué es un proceso y qué es un hilo.
- ✓ Explicar la diferencia entre un proceso y un hilo.
- ✓ Explicar cómo se realiza la comunicación entre los procesos y los hilos.
- ✓ Identificar los problemas más comunes que se dan en la comunicación de los procesos.
- ✓ Analizar los diferentes algoritmos que se han diseñado para eliminar o disminuir los problemas de comunicación entre procesos.
- ✓ Explicar qué es la planificación de los procesos.

Guía de lectura

Antes de realizar continuar con la lectura de esta guía, estudie detalladamente los siguientes apartados en el libro de texto:

- a. Concepto de procesos, página 83.
- b. Concepto de Hilo, página 95.
- c. Estudio y análisis de la comunicación entre procesos, página 117.
- d. Estudio y análisis sobre los algoritmos de planificación de procesos, página 145.



Conceptos clave

Al finalizar el estudio de este capítulo usted deberá tener claros los siguientes conceptos:

- ✓ Proceso
- ✓ Hilo
- ✓ Multiprogramación
- ✓ Estados de un proceso
- ✓ Demonios
- ✓ Región crítica
- ✓ Condiciones de competencia
- ✓ Exclusión mutua
- ✓ Planificación

Aspectos sobresalientes

2.1. Procesos

Los programas están compuestos por procesos (uno o más) que se ejecutan de manera “pseudoparalela” contando cada uno con su propio espacio en memoria, su propio calendario de ejecución y su propia pila. Recuerde que usted estudió esta percepción de pseudoparalelismo de los procesos en el capítulo 1. Solamente los computadores que tienen más de un procesador trabajan de manera paralela aunque compartan la misma memoria física.

Algunos procesos pueden a su vez, subdividirse en procesos más pequeños llamados hilos (antes se les denominaba subprocesos), éstos, a diferencia de los procesos, sí comparten el mismo espacio de direcciones y pueden implementarse, tanto en el espacio de usuario como en el espacio dedicado al kernel.



2.1.1. Modelo de Procesos

Un proceso puede tomarse como un programa en ejecución o como una parte de un programa en ejecución. Si un programa tiene pocas líneas de programación, es posible que el sistema operativo lo tome como un solo proceso, pero si el programa tiene cientos de líneas de programación, el SO deberá subdividirlo en porciones más manejables, lo que podrá tomarse también como procesos.

2.1.2. Creación de un proceso

Según sea la circunstancia, existen diferentes formas de crear un proceso; sin embargo, independientemente de donde venga la petición de hacerlo, todos son creados por el SO.

Tanenbaum (2009) menciona cuatro sucesos principales que pueden generar su creación:

1. “Inicialización del sistema
2. Ejecución de una llamada al sistema para crear procesos
3. Solicitud de un usuario para crear un proceso
4. Inicio de un trabajo por lotes” (p.86)

Cuando se arranca un SO, por lo general, se crean varios procesos. Algunos son de primer plano, es decir, procesos que interactúan con usuarios y trabajan para ellos. Otros son de segundo plano, es decir, que no están asociados con un usuario en particular, sino que tienen una función específica.

Los procesos que permanecen en segundo plano para encargarse de alguna actividad preestablecida, se llaman demonios. Los hay de impresión y de lectura.

Cada vez que un proceso genera un documento para imprimir éste se manda a una cola de impresión. En este caso, el “demonio de impresión” es un proceso que está monitoreando constantemente si existen archivos por imprimir. Cuando descubre que hay un archivo en la cola de impresión, lo envía a la impresora a la cual está destinado. Si la impresora está siendo utilizada en otro trabajo de impresión deja el trabajo en espera. Si hay más trabajos por imprimir que corresponden a otra impresora (un sistema puede tener varias impresoras asociadas) revisa si a la impresora a la que están destinados están libres. Cuando encuentra una impresora disponible, lo envía a imprimir y borra el trabajo de la cola de impresión.



Para el pago de las cuotas mensuales de un préstamo hipotecario con un banco estatal, no recurren a la deducción automática del salario, sino que rebajan de la cuenta del deudor, el monto correspondiente a la cuota cada mes. En este caso, posiblemente el banco debe tener una aplicación (“demonio de lectura”) que todos los días recorre las cuentas de todos los deudores que deben pagar en esa fecha, con el fin de ver si existe dinero suficiente que cubra la cuota. Si lo hay, inmediatamente rebaja de la cuenta el monto de la cuota.

2.1.3. Terminación de procesos

Tarde o temprano un proceso en ejecución terminará. Tanenbaum (2009) menciona que, por lo general, la terminación de un proceso se debe a una de las siguientes condiciones: “terminación normal (voluntaria), terminación por error (voluntaria), error fatal (involuntaria) o terminado por otro proceso (involuntario)” (p.88).

A estas cuatro se podría agregar una más: apagando el equipo (acción con la cual se daría fin a todos los procesos). El problema sería que algunos procesos finalizarían normalmente, pero otros, al interrumpirse abruptamente, podrían generar errores en los datos. Una forma similar y la que se tiene que prever es la falta de fluido eléctrico de manera imprevista.

2.1.4. Jerarquía de procesos

Para explicar el tema de la jerarquía de procesos, Tanenbaum lo presenta de la siguiente manera: “En algunos sistemas, cuando un proceso crea otro, el proceso padre y el hijo mantienen cierta asociación. El proceso hijo puede, a su vez, crear más procesos y formar una **jerarquía de procesos**. Por supuesto, en este caso, un proceso padre no puede finalizar hasta que todos los procesos hijos hayan finalizado y le haya devuelto el control” (p.89).

2.1.5. Estados de un proceso

Según Tanenbaum (2009), un proceso puede encontrarse en tres estados posibles:

1. En ejecución. Usando la CPU en ese instante.
2. Listo. Puede ejecutarse; detenido temporalmente para permitir que se ejecute otro.
3. Bloqueado. No puede ejecutarse mientras no ocurra cierto suceso externo.



Tanenbaum (2009) también afirma que hay cuatro transiciones posibles entre estos tres estados, como puede observarlo en la figura 2-2 de la página 90 del libro de texto.

El siguiente ejemplo ilustra el estado “listo”: Un procesador sólo puede ejecutar un proceso a la vez, pero el proceso en ejecución tiene tiempos “muertos” cuando hay tareas de Lectura/Escritura. En esos lapsos es cuando el procesador busca cuál proceso está listo y lo ejecuta. Cuando el primer proceso “regresa” devuelve el segundo proceso a estado “listo” y prosigue con el primero.

El estado “bloqueado” se da cuando un proceso no puede ejecutarse porque no tienen todos los recursos disponibles para su ejecución. Puede ser que haya estado en ejecución pero al llegar a cierta parte requiere de un recurso que está asignado a otro proceso, entonces el proceso pasa de estar en estado “ejecución” a estado “bloqueado” mientras espera que se libere el recurso que requiere. Una vez que se le asigna el recurso pasa a estado “listo” a la espera que el procesador le asigne tiempo de procesamiento.

En la figura 2-2 del libro de texto (página 90) puede observar que un proceso no pasa de estado “bloqueado” a estar en “ejecución”; tampoco pasa de “listo” a “bloqueado”. Esto último significa que una vez que se le asignan los recursos al procesador, no los suelta hasta que entren en ejecución. Amplíe en las páginas de la 83 a la 95 del libro de texto.

2.2. Hilos

En páginas anteriores de esta guía, se ha mencionado el concepto “pseudoparalelismo”, o sea, la percepción del usuario de que el computador está ejecutando varios procesos de manera simultánea. Al respecto, se aclaró que no es así, sino que los tiempos “muertos” de un proceso son aprovechados por el procesador para ejecutar otro.

El concepto de hilos que menciona Tanenbaum (2009) se puede tomar, para efectos de una mejor comprensión del tema, como miniprocesos; es decir, los pequeños procesos en los que el sistema operativo subdivide un proceso, los que se comportan bajo el mismo concepto de pseudoparalelismo.

Una de las diferencias entre los procesos y los miniprocesos (hilos) es que cada uno de los procesos ocupa su propio espacio de direcciones para ejecutarse, mientras que todos los hilos o miniprocesos de un proceso comparten el mismo espacio de direcciones.



Existen varias ventajas en descomponer un proceso en hilos o miniprocesos, entre ellas:

- Evitar bloqueos, pues al estar subdividido el proceso, la ejecución es más sencilla.
- Los hilos son más fáciles de iniciar y terminar.
- Los hilos son más rápidos en la ejecución, así que el tiempo total de ejecutar todos los hilos de un proceso es menor a que si se ejecutara el proceso sin subdividirse.
- Si el computador tiene varios procesadores puede efectuarse el verdadero paralelismo a nivel de hilos.

Los hilos se pueden ejecutar en espacio de usuario, en espacio de kernel o en forma híbrida entre ambos. Mientras los hilos se ejecutan en el espacio de usuario, el SO no se entera de lo que está ocurriendo; esto le permite dedicarse a otras tareas de procesamiento. Al igual que los procesos, los hilos requieren ser calendarizados o planificados, aún así, con esta similitud, los hilos se ejecutan de manera más eficiente. Existen hilos que se ejecutan en el espacio del kernel, lo que permite que el SO tenga su control completo.


La forma híbrida básicamente consiste en ejecutar los hilos en el espacio del usuario, pero se pasa el control al modo kernel cuando se requieren algunas tareas especializadas, como crear nuevos hilos o destruir algún hilo. Una vez finalizada la tarea, se regresa el control al modo de usuario.

Amplíe sobre hilos o microprocesos en las páginas de la 95 a la 117 del libro de texto.

2.3. Comunicación entre procesos

La comunicación entre procesos es sumamente importante. Si solamente tuviéramos un proceso en ejecución no habría ningún problema, pues todos los recursos de la máquina estarían a su disposición, pero cuando son decenas o cientos de procesos compitiendo entre sí por recursos, espacio en disco, espacio en memoria principal, tiempo de procesador, etc.; es cuando se requiere urgentemente de un verdadero director de orquesta.

Para aclarar el tema, imagine a tres buzos a varios metros bajo el nivel del mar realizando un trabajo específico, pero con un solo tanque de oxígeno y una boquilla. Los tres podrían competir por ver, quién utiliza la boquilla y en la pelea podría ser que uno o dos de ellos se ahoguen. Para evitar esta tragedia, antes de bajar, el jefe les hace un plan para aprovechar el único recurso, y finalizar el trabajo. Una forma podría ser que uno de ellos absorba el oxígeno y una vez que lo haga, lo pase al siguiente. Seguramente mientras el segundo buzo



esté absorbiendo el oxígeno, el primero lo estará exhalando y aguanta la respiración por unos pocos segundos, mientras el tercero hace la misma rutina y lo pasa nuevamente al primero y así, sucesivamente. Con un plan así de sencillo, los tres procesos (buzos) obtendrán el recurso (oxígeno) de manera ordenada hasta completar cada uno su trabajo. Conforme cada buzo va terminando su tarea emerge del agua y el recurso (oxígeno) será compartido por menos procesos y así, sucesivamente hasta que el último finalice.

La lucha de los procesos por obtener recursos es un tema que sucede segundo a segundo en el computador. Es responsabilidad del SO administrar adecuadamente todos los recursos para poder satisfacer a todos los procesos en sus requerimientos de tiempo de procesador, memorias y hardware, con el objetivo de que cada uno finalice su tarea de manera exitosa.

En los siguientes párrafos se hará referencia a tres conceptos que están estrechamente ligados al tema: condiciones de competencia, región crítica y exclusión mutua.

La condición de competencia se da cuando diferentes procesos requieren utilizar los mismos recursos de manera simultánea, sea para operaciones de lectura o escritura de datos. Por ejemplo, si los empleados de una empresa comparten una red local y cuentan con un repositorio en el que hay un documento (recurso) que puede ser utilizado por varias personas (procesos) y cada quien puede leerlo, modificarlo y posteriormente grabarlo, es posible que, sin darse cuenta, dos funcionarios accedan al mismo archivo de manera simultánea. En este momento entran en competencia por ese recurso. ¿Qué hace el sistema operativo? Aquí entra a participar el segundo concepto mencionado, sección crítica.

Se denomina **sección crítica**, en programación concurrente, a “la porción de código de un programa de computador en la que se accede a un recurso compartido (estructura de datos o dispositivo) que no debe ser accedido por más de un hilo en ejecución. La sección crítica por lo general termina en un tiempo determinado y el hilo, proceso o tarea sólo tendrá que esperar un período determinado de tiempo para entrar (...). Sólo un proceso puede estar en una sección crítica a la vez”. (Pita, 2007)

Siguiendo con el ejemplo, el programa que usted está utilizando para acceder el documento entra en su sección crítica, lo que es esa porción del programa que le permite realizar una labor de lectura y escritura, pero que no es exclusiva de esa aplicación, pues otras, o la misma invocada por otro usuario, pueden hacer lo mismo con el mismo archivo.

Cuando se da esta situación, se corre el riesgo de compartir memoria, archivos o cualquier otra cosa, entonces es necesario encontrar una manera de impedir que dos o más procesos



lean o escriban los datos compartidos al mismo tiempo. Aquí aparece el tercer concepto mencionado, la exclusión mutua, la que se refiere a algún mecanismo para asegurarnos de que si un proceso está utilizando una variable compartida o un archivo compartido, los demás procesos no podrán hacer lo mismo hasta que éste sea liberado. Por ejemplo, en un servidor que tiene un archivo de datos en Excel, que es actualizado por varios funcionarios, si uno de los funcionarios solicita acceso al archivo, el SO verifica si ya hay alguien ocupándolo; si el archivo está libre se lo asigna, pero si el archivo está en uso, el SO le enviará un mensaje en el que le avisará que el archivo está en uso y le asignará acceso, pero sólo en modo de lectura, es decir, que no puede modificar datos. Cuando el primer usuario libere el archivo, entonces el otro funcionario podrá acceder a él en forma de lectura y escritura, siempre que otro usuario no se le adelante.

En las páginas de la 120 a la 144 de su libro de texto, verá una serie de algoritmos ideados para que el SO administre la interacción entre procesos y la competencia por recursos.

2.4. Planificación

Al inicio de este capítulo se mencionó el concepto de planificación, refiriéndose a la manera en que el SO programa la ejecución de un proceso. Se dijo que si un computador solamente tiene un procesador y ejecuta solamente un proceso, no habría mayores problemas, el proceso tendría todos los recursos a su disposición. Pero este no es el caso, el tiempo del procesador hay que aprovecharlo al máximo. Como esta es la realidad y hay gran cantidad de requerimientos de procesamiento, el SO debe planificar la ejecución de cada proceso y de sus respectivos hilos. Para evitar contratiempos y bloqueos innecesarios se debe planificar, es decir calendarizar cada uno de los procesos.

¿Cuándo planificar?

Existen diversas situaciones en las que es necesario planificar:

Según Tanenbaum (2009), se debe planificar en las siguientes situaciones:

- Cuando se crea un proceso, pues hay que decidir si se ejecutará el proceso padre o el hijo. El planificador está en su derecho de escoger al padre o al hijo para ejecutar, la razón de esta situación es porque ambos están en condición de listo.



- Cuando un proceso termina, ya que se debe tomar una decisión de planificación. Se deberá escoger otro del conjunto de procesos listos. Si ningún proceso está listo, lo normal es que se ejecute un proceso inactivo suministrado por el sistema.
- Cuando un proceso se bloquea por E/S, un semáforo o algún otro motivo, y es preciso escoger otro proceso que se ejecute. En ocasiones, el motivo del bloqueo puede afectar la decisión.
- Cuando ocurre una interrupción de E/S tal vez haya que tomar una decisión de planificación.

Aquí hay dos nuevos conceptos: planificación apropiativa y planificación no apropiativa.

Un algoritmo de planificación **no apropiativo** (o no referente) escoge el proceso que se ejecutará y luego simplemente le permite ejecutarse hasta que se bloquee o hasta que sea cedido por la CPU de manera voluntaria.

Un algoritmo de planificación **apropiativo** (o preferente) escoge un proceso y le permite ejecutarse durante un tiempo establecido. Si al término de ese tiempo, el proceso continúa en ejecución, se le suspende y el planificador escoge otro proceso para que se ejecute (si existe alguno disponible).

En el primero, es el CPU quien decide en qué momento cambia a otro proceso; en el segundo, el tiempo (lapso de ejecución) es el que obliga al CPU a cambiar de proceso.

Categorías de algoritmos de planificación

Tanenbaum (2009) menciona tres categorías que vale la pena distinguir:

1. **Por lotes:** en esta categoría no existen usuarios esperando, ante sus terminales, a que el sistema responda con rapidez. Por ello, suelen ser aceptables los algoritmos **no apropiativos** o los **apropiativos** con intervalos de tiempo largos para cada proceso.
2. **Interactivo:** aquí, la expropiación es indispensable para evitar que un proceso acapare la CPU y niegue servicio a otros.
3. **Tiempo real:** en los sistemas con restricciones en tiempo real, a veces no es necesaria la expropiación, porque los procesos saben que tal vez no se ejecuten durante mucho tiempo y, por lo general, realizan su trabajo y se bloquean rápidamente.

Las metas de los algoritmos de planificación

A la hora de diseñar el algoritmo de un programa, es importante tener claro cuál es el objetivo de éste y hacia qué tipo de procesamiento va dirigido. Algunos objetivos dependen



del entorno (lotes, interactivo, tiempo real), pero también existen objetivos que son deseables en todos los casos.

En la figura 2-39, de la página 150 del libro de texto, se puede observar una lista de algunos objetivos de acuerdo al tipo de procesamiento que se espera.

2.4.1. Planificación en sistemas de procesamiento por lotes

Para la planificación en sistemas por lotes, existen varios métodos, entre ellos:

- Primero en llegar, primero en ser atendido
- Trabajo más corto primero
- Tiempo restante más corto a continuación

2.4.2. Planificación en sistemas interactivos

Existen varios algoritmos de planificación de procesos, entre ellos:

- Planificación por turno circular (Round-Robin). Observe la figura 2-41 de la página 155 de su libro de texto para comprender este algoritmo de planificación.
- Planificación por prioridades
- Múltiples colas
- Proceso más corto a continuación
- Planificación garantizada
- Planificación por sorteo
- Planificación por porción equitativa

2.4.3. Planificación en sistemas de tiempo real

Los **sistemas en tiempo real** se clasifican en general como en tiempo real estricto. Esto implica que existen plazos absolutos que deberán cumplirse, pase lo que pase, y en tiempo real no estricto, en los que pueden tolerarse incumplimientos ocasionales, aunque indeseables, de los pasos.

En ambos casos, el comportamiento en tiempo real se logra dividiendo el programa en varios procesos, cuyo comportamiento es predecible y se conoce con antelación. Por lo



general, tales procesos son cortos y pueden terminar su trabajo en mucho menos de un segundo.

Para ampliar sobre los temas, refiérase a las páginas de la 145 a la 163 del libro de texto.

2.5 Problemas clásicos de comunicación entre procesos (IPC)

Cuando se habló de comunicación entre procesos se expusieron los algoritmos que, en teoría, son los óptimos para que todo funcione de manera adecuada y sin problemas entre sí, pero lo interesante de todo esto es, que si analizamos detenidamente cada uno de los algoritmos podríamos encontrar una serie de situaciones en las que no todo resultaría como lo hemos planeado. Quizá no hemos analizado cuánto tiempo podría pasar un proceso en su estado de bloqueado o en estado listo.

En esta sección se exponen dos problemas que, si analizamos todo lo anteriormente expuesto, podrían aparecer en cualquier momento. Estos dos ejemplos están claramente ilustrados en las páginas de la 164 a la 168 del libro de texto.

Ejercicios de autoevaluación

A continuación, se presentan ejercicios que tienen como objetivo que usted pueda evaluar su aprendizaje sobre cada uno de los puntos contenidos en el desarrollo del capítulo 2.

Para el capítulo 2, realice los ejercicios 1, 3, 14, 22, 27, que se encuentran en las páginas 170, 171 y 172 del libro de texto.

ADMINISTRACIÓN DE MEMORIA

3

Sumario

- *Sin abstracción de memoria*
- *Una abstracción de memoria: espacio de direcciones*
- *Memoria virtual*
- *Algoritmo de reemplazo de páginas*
- *Cuestiones de diseño para los sistemas de paginación*
- *Cuestiones de implementación*
- *Segmentación*



Objetivos de aprendizaje

Al finalizar el estudio de este tema, entre otras habilidades, usted será capaz de:

- ✓ Explicar el concepto de administración de memoria básica.
- ✓ Explicar los procesos de intercambio.
- ✓ Explicar el concepto de memoria virtual.
- ✓ Comprender los algoritmos de reemplazo de páginas.
- ✓ Comprender el modelado de algoritmos de reemplazo de páginas.
- ✓ Comprender aspectos de diseño de los sistemas con paginación.
- ✓ Comprender aspectos de implementación.
- ✓ Explicar el concepto de segmentación.

Guía de lectura

Antes de realizar continuar con la lectura de esta guía, estudie detalladamente los siguientes apartados en el libro de texto:

- a. Sin abstracción de Memoria, página 176.
- b. Una abstracción de memoria, página 179.
- c. Memoria virtual, página 188.
- d. Algoritmos para reemplazo de páginas, página 201.
- e. Cuestiones de diseño para los sistemas de paginación, página 216.
- f. Cuestiones de implementación, página 234.
- g. Segmentación, página 234.
- h. Llamadas al sistema, página 49.
- i. Estructura de un sistema operativo, página 62.



Conceptos clave

Al finalizar el estudio de este capítulo usted deberá tener claros los siguientes conceptos:

- ✓ Jerarquía de memoria
- ✓ Administración de memoria
- ✓ Registro PSW
- ✓ Espacio de direcciones
- ✓ Intercambio
- ✓ Administración de memoria libre
- ✓ Administración de memoria con listas ligadas
- ✓ Memoria virtual
- ✓ Paginación
- ✓ Algoritmos para reemplazo de páginas
- ✓ Manejo de fallos de página
- ✓ Segmentación

Aspectos sobresalientes

En el capítulo 1 nos referimos superficialmente a los diferentes tipos de memoria: la principal o RAM (Memoria de Acceso Aleatorio), las secundarias (como disco duro, cintas, CD's, DVD's, memorias flash) y un tipo de memoria alojado en la principal llamado caché.

Lo anterior podemos verlo como una jerarquía de memoria: principal, secundarias y caché; cada una con sus propias características, por ejemplo, la caché es una memoria de acceso rápido, volátil, pequeña, mientras que la principal es rápida, de mediana capacidad y volátil; las secundarias son de gran capacidad, lentas y no volátiles. En este capítulo se hará énfasis en dos tipos de memoria: la principal y la caché.

La memoria principal es aquella que utiliza el procesador para realizar todas sus operaciones, aritméticas y lógicas.



El SO tiene muchas tareas que realizar, entre ellas administrar los diferentes tipos de memoria. La parte del SO que administra su jerarquía se llama administrador de memoria, su responsabilidad es mantener informado al SO sobre qué partes de la memoria están en uso y cuáles no, además, asignar memoria a los procesos cuando la necesitan y liberarla cuando termina, y administrar los intercambios entre la memoria principal y el disco cuando la primera es demasiado pequeña para contener todos los procesos.

Los programas de administración pueden dividirse en dos clases: los que traen y llevan procesos entre la memoria principal y el disco durante la ejecución (intercambio y paginación), y los que no lo hacen. El intercambio y la paginación son, en gran medida, mecanismos artificiales, obligados por la falta de suficiente memoria principal para contener todos los programas a la vez. Recordemos que la memoria principal es relativamente pequeña, 4GB u 8GB en RAM (las hay más grandes) comparada con la capacidad del disco duro que anda en capacidades mayores a los 300GB.

Existen programas de aplicación muy grandes. Un programa con todos sus módulos puede medir más de 10 GB. Si usted tiene en su PC una memoria de 4 GB, es imposible que su programa pueda subir todos los módulos, eso sin contar el espacio para los datos y otros programas de control. Lo que el sistema operativo hace, llamado intercambio, es subir únicamente la parte del programa que usted va a utilizar; cuando requiera utilizar otra parte quita de la memoria la que tenía y sube la nueva. Esta dinámica la mantiene durante toda la ejecución. Si usted no utiliza algunos módulos, no hay razón para subir esa parte del programa a la memoria. En algunos programas desarrolladores más viejos el propio programador subdividía el programa mediante la creación de “overlays” o módulos.

Existen dos conceptos nuevos a los que debe poner atención antes de continuar con la lectura: intercambio y paginación.

3.1. Sin abstracción de memoria

La monoprogramación, sin intercambio ni paginación, es el esquema de administración de memoria más sencillo posible. Consiste en ejecutar únicamente un programa a la vez, repartiendo la memoria entre ese programa y el SO. Si el sistema está organizado de esta manera, únicamente puede ejecutarse un proceso a la vez. Tan pronto como el usuario teclea un comando, el SO copia el programa solicitado del disco a la memoria y lo ejecuta. Cuando el proceso termina, el SO exhibe un indicador de comandos y aguarda un nuevo comando.



3.2. Una abstracción de memoria: espacio de direcciones

La otra cara de la moneda es la multiprogramación, usada por la gran mayoría de los sistemas modernos; a través de ella se permite la ejecución de varios procesos al mismo tiempo (recordemos el concepto de pseudoparalelismo), con esto se eleva el aprovechamiento de la CPU. La memoria puede partitionarse y a cada una de esas particiones se le asigna un espacio de direcciones en donde se ubicará un proceso, dependiendo del tamaño de la memoria, así podrá partitionarse.

El espacio de direcciones usa dos variables llamadas registro base y registro límite, el primero especifica en dónde se inicia el espacio de direcciones y el segundo su longitud.

Como se mencionó al principio, la memoria principal no es lo suficientemente grande para albergar todos los procesos a la vez, pero sí es muy importante que todos los procesos se vayan ejecutando de alguna manera utilizando el principio de pseudoparalelismo.

Pueden usarse dos enfoques generales para la administración de memoria, dependiendo (en parte) del hardware disponible. La estrategia más sencilla, llamada intercambio (swapping), consiste en traer a la memoria un proceso entero, ejecutarlo durante un rato y volver a guardarlo en disco. La otra estrategia, llamada memoria virtual, permite que los programas se ejecuten aunque solo una parte de ellos estén en la memoria principal. (Tanenbaum, 2009)

Otros dos conceptos para estudiar de manera detenida en el libro de texto: Intercambio y Memoria Virtual (páginas 181 a la 201).

Cuando el intercambio crea múltiples huecos en la memoria, es posible combinar tales huecos en uno sólo más grande, desplazando todos los procesos hacia abajo hasta donde sea posible. Esta técnica se llama compactación de memoria.

Si la memoria se asigna en forma dinámica, el SO debe administrarla. En términos generales, existen dos formas de llevar el control del uso de la memoria: mapa de bits y listas ligadas. Para el segundo concepto se han diseñado varios algoritmos para administrar la memoria. Ambos conceptos están muy bien explicados en el libro de texto y merecen ser analizados por el estudiante. (Ver páginas 185 a la 187)

(Para ampliar sobre el tema de administración de memoria con mapas de bits y con listas enlazadas, consulte la sección 3.2, páginas de la 179 a la 187 del libro de texto.)



3.3. Memoria virtual

La idea básica del método conocido como memoria virtual es que el tamaño combinado del programa, sus datos y su pila podrían exceder la cantidad de memoria física que se le puede asignar. El sistema mantiene en la memoria principal, las partes del programa que se están usando en ese momento, y el resto en el disco. La mayoría de los sistemas con memoria virtual utilizan la técnica llamada paginación.

En cualquier computadora existe un conjunto de direcciones de memoria que los programas pueden producir. Las direcciones pueden generarse empleando indización, registros base, registros de segmentos y otros métodos. Estas direcciones generadas por el programa se denominan direcciones virtuales y constituyen el espacio de direcciones virtual.

Cuando se usa memoria virtual, las direcciones virtuales no se envían de manera directa al bus de memoria, sino a una unidad de administración (MMU; memory management unit) que establece una correspondencia entre las direcciones virtuales y físicas de la memoria.

El espacio de direcciones virtuales se divide en unidades llamadas páginas. Las unidades correspondientes en la memoria física se denominan marcos de página. El número de página se utiliza como índice para consultar la tabla de páginas y así, obtener el número del marco de página que corresponde a la página virtual.

El propósito de la tabla de páginas es establecer una correspondencia entre las páginas virtuales y los marcos de página. A pesar de lo sencillo de esta descripción, hay que resolver dos problemas importantes: una, la tabla de páginas puede ser extremadamente grande y la segunda, la transformación (correspondencia) debe ser rápida.

(Para ampliar el tema correspondiente a la memoria virtual y sus subtemas refiérase a la sección 3.3, páginas 188 a la 201 del libro de texto.)

3.4. Algoritmos de reemplazo de páginas

Para el reemplazo de páginas existen varios algoritmos que, igualmente, están descritos detalladamente en el libro de texto y que aquí los nombramos.

1. En el algoritmo óptimo de reemplazo de páginas.
2. En el algoritmo de reemplazo de páginas no usadas recientemente (NRU).



3. Primero en entrar, primero en salir (FIFO; first-in, first-out).
4. Segunda oportunidad.
5. Reemplazo de páginas tipo reloj.
6. Reemplazo de página menos recientemente usada (LRU).
7. Reemplazo de páginas de conjunto de trabajo y
8. Reemplazo de páginas WSClock

(Refiérase a las secciones 3.4, páginas de la 201 a la 215 del libro de texto.)

3.5. Cuestiones de diseño para los sistemas de paginación

En esta sección se profundiza en el tema de paginación, básicamente se enfatiza en temas en los que el estudiante debe profundizar y los diseñadores consideran para poder obtener un buen rendimiento de un sistema de paginación. Estos temas son:

1. Políticas de asignación local contra las de asignación global
2. Control de carga
3. Tamaño de página
4. Espacios separados de instrucciones y de datos
5. Páginas compartidas
6. Bibliotecas compartidas
7. Archivos asociados
8. Política de limpieza
9. Interfaz de memoria virtual

Para ahondar en estos temas refiérase a las páginas 216 a la 227 del libro de texto.

3.6. Cuestión de implementación

El SO tiene participación activa en lo relacionado con el ciclo de la paginación, que consiste en crear un proceso, ejecutarlo, atender un fallo de página y finalizarlo.

El concepto de fallo de página toma relevancia, pues es cuando el SO debe recurrir al hardware, en este caso el disco duro, para volver a restablecer la página que se cayó.

Para ahondar en el tema de los fallos de página refiérase a las secciones 3.6.2, 3.6.3, 3.6.4, 3.6.5 y 3.6.6 del libro de texto, páginas de la 227 a la 334.



3.7. Segmentación

La memoria virtual que hemos tratado hasta ahora es unidimensional, porque las direcciones virtuales van desde cero hasta alguna dirección máxima, y son consecutivas. En el caso de varios problemas, podría ser mucho mejor tener dos o más espacios de direcciones distintos que sólo uno.

Un segmento es proporcionar a la máquina varios espacios de direcciones independientes por completo. Dado que cada segmento constituye un espacio de direcciones distinto, los segmentos pueden crecer o decrecer en forma independiente, sin afectarse unos a otros.

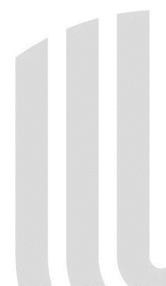
Un segmento podría contener un procedimiento, un arreglo, una pila o una colección de variables escalares, pero por lo regular no contiene una mezcla de tipos distintos. (Vea una comparación entre paginación y la segmentación en la figura 4-37 de la página 252)

La implementación de la segmentación difiere de la paginación en un aspecto fundamental: las páginas son de tamaño fijo y los segmentos no. (Para ampliar sobre segmentación refiérase a la sección 4.8 Segmentación, páginas de la 234 a la 247 del libro de texto.)

Ejercicios de autoevaluación

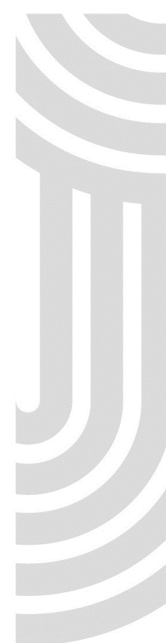
A continuación, se presentan ejercicios que tienen como objetivo evaluar cada uno de los puntos contenidos en el desarrollo del capítulo.

Para el capítulo 3, realice los siguientes ejercicios que se encuentran localizados en el libro de texto en las páginas 248 a la 254.



4

SISTEMAS DE ARCHIVOS



Sumario

- *Archivos*
- *Directorios*
- *Implementación de sistemas de archivos*
- *Administración y optimización de sistemas de archivos*



Objetivos de aprendizaje

Al finalizar el estudio de este tema, entre otras habilidades, usted será capaz de:

- ✓ Explicar lo referente a los archivos de E/S.
- ✓ Explicar lo referente a directorios.
- ✓ Explicar lo concerniente a la implementación de sistemas de archivos.
- ✓ Conocer ejemplos de sistemas de archivos.

Guía de lectura

Antes de realizar continuar con la lectura de esta guía, estudie detalladamente los siguientes apartados en el libro de texto:

- a. Conceptos sobre archivos, página 257.
- b. Conceptos acerca de Directorio, página 268.
- c. Implementación de sistemas de archivos, página 273.
- d. Administración y optimización de sistemas de archivos, página 292.



Conceptos clave

Al finalizar el estudio de este capítulo usted deberá tener claros los siguientes conceptos:

- ✓ Archivos
- ✓ Directorios
- ✓ Implementación de sistemas de archivos
- ✓ Ejemplos de sistemas de archivos

Aspectos sobresalientes

Una de las tareas más importantes que tiene un computador es la poder almacenar grandes cantidades de información y poder recuperarla en el momento preciso. Si bien mucha de esta información se almacena en la memoria principal, que es donde se ejecutan las aplicaciones y se procesa la información, lo importante es poder almacenarla y poder recuperarla.

Para poder realizar esta tarea se requiere de dispositivos de almacenamiento llamados también memorias secundarias que son permanentes.

Según Tanenbaum (2009) “tenemos tres requisitos indispensables para el almacenamiento de información a largo plazo:

1. Debe tener la posibilidad de almacenar una cantidad muy grande de información.
2. La información debe sobrevivir a la terminación del proceso que la usa.
3. Debe existir la capacidad de que múltiples procesos accedan a la información de manera concurrente.”

Almacenar la información en medios externos, tales como: discos duro, discos externos, memoria flash, cd's, y otros medios externos, es la solución a los requisitos mencionados anteriormente.



Otra tarea fundamental del sistema operativo es la de administrar la forma en que se nombran, se recupera, se almacena la información en los dispositivos externos.

Si bien procesar información a través de un computador es tarea de todos los días, por lo único que nos preocupamos es el medio en el que almacenaremos la información, el resto le toca al sistema operativo.

4.1. Archivos

Cuando un proceso crea un archivo, le asigna un nombre. Cuando el proceso termina, el archivo sigue existiendo y otros programas pueden tener acceso a él, utilizando su nombre.

Generalmente los nombres de los archivos, sea de datos o de programas, en los equipos mainframe están compuestos por una serie de caracteres de entre 6 a 8 caracteres dependiendo del software de desarrollo, no así a nivel de PC en que los nombres de archivos pueden ser de hasta 30 caracteres.

Estos nombres están compuestos de dos partes separadas por un punto, la parte a la derecha del punto se llama extensión e indica el tipo de archivo que es. En la figura 4.1., en la página 258 del libro de texto, podrá ver una gran cantidad de extensiones y el tipo de archivo que identifican.

Los archivos pueden organizarse internamente de varias maneras. En la figura 4-2 de la página 259, del libro de texto se ilustran las tres posibilidades más comunes.

Una forma puede verse como una sucesión de bytes, lo que ofrece el máximo de flexibilidad. La otra forma es organizarse como una sucesión de registros y la tercera forma es como un árbol.

Es importante que el estudiante tenga presente que los archivos van más allá de simples archivos de texto, hojas de cálculo y otros similares. El libro hace especial énfasis en los archivos de datos, esos que se utilizan para almacenar información de personas, características de productos, estadísticas y que son utilizados por programas de aplicación para atender clientes, cobra recibos, diferentes tipos de pagos, etc.



Los archivos para efectos de almacenamiento y manejo de datos pueden clasificarse en dos tipos: de acceso secuencial y de acceso aleatorio. Es importante que el estudiante identifique cuáles son las características de cada tipo de organización y analice sus respectivas ventajas y desventajas.

Todo archivo tiene una serie de atributos que le pueden ser útiles al usuario, tiene un nombre y datos. Además, todos los sistemas operativos asocian otra información a cada archivo, como la fecha y la hora en que se creó y su tamaño.

La figura 4-4 de la página 263 del libro de texto, presenta una tabla en la que se describen los atributos que puede tener un archivo.

En la sección 4.1.6 del libro de texto, páginas 264 y 265 se muestran una serie de primitivas (palabras propias del SO) que utilizan los sistemas operativos para la manipulación de archivos.

4.2. Directorios

Para llevar el control de los archivos, los sistemas de archivos suelen tener directorios o carpetas. Esto es algo que comúnmente hacemos todos los días desde nuestros pc, algo similar ocurre en los macrocomputadores. A diferencia que en los PC, en los grandes computadores se utiliza el concepto de bibliotecas (Library) para identificar las carpetas.

Analice las ventajas de tener la posibilidad de crear un número arbitrario de subdirectorios contra el hecho de mantener todos los archivos en una sola carpeta o directorio o biblioteca. Además, qué mecanismos debería implementar, según el libro de texto, para tener acceso a los archivos. El libro de texto menciona dos sistemas, el de un solo nivel y el jerárquico.

Cuando se utiliza el sistema jerárquico de almacenamiento se tiene la ventaja de que podemos tener archivos con el mismo nombre almacenados en diferentes subdirectorios. Para poder llegar a ellos se requiere de una ruta la que le indica al sistema operativo cuál es realmente el archivo (aunque se llamen igual) que requerimos para trabajar. Para ampliar este concepto, estudie la sección 4.2.3, página 269 del libro de texto.



4.3. Implementación de Sistemas de Archivos

Un tema que debe tener suma atención en el estudio de este capítulo es lo referente a la implementación de sistemas de archivos (sección 4.3, página 273). En esta sección encontraremos conceptos como: Máster Boot Record (MBR), también llamado sector 0 (cero).

Es importante tener en cuenta que los discos se dividen en bloques y además, los archivos no siempre (casi nunca) se almacenan de manera continua, así que es importante saber en qué bloques está almacenado un archivo. Para la implementación de los archivos Tanenbaum (2009) menciona algunos métodos, entre ellos está el de asignación continua, otro método es el de lista enlazada, un tercer método es el de lista enlazada utilizando una tabla de memoria, también encontramos el método de nodos-i. El detalle de estos métodos puede estudiarlos en las páginas 274 a la 279 del libro de texto.

Así como se requiere una implementación de archivos, y como estos se almacenan en directorios y subdirectorios, también se requiere de una implementación de directorios.

Tanenbaum (2009) dice que la función principal del sistema de directorios es establecer una correspondencia entre el nombre del archivo ASCII y la información necesaria para localizar los datos.

Un aspecto estrechamente relacionado es dónde deben guardarse los atributos. Todo sistema de archivos mantiene atributos de los archivos, quién es su dueño y fecha de creación y última fecha de modificación. Esta información complementaria al archivo en sí, debe almacenarse en algún lado. Una posibilidad obvia es guardarlos directamente en la entrada de directorio. Muchos sistemas hacen precisamente eso.

En algunas instalaciones, hablando de centros de cómputo multiusuario, a los usuarios se les asignan cuotas de espacio para evitar que pocos acaparen demasiado espacio en disco. La idea consiste en que el administrador del sistema asigne a cada usuario una porción máxima de archivos y bloques, y que el sistema operativo cuide que los usuarios no excedan la cuota.

Aunque el sistema de archivos no puede ofrecer protección contra la destrucción física del equipo y los medios, sí puede ayudar a proteger la información.



Un concepto que se maneja de manera frecuente a la hora de buscar archivos en el disco, es el uso de la memoria caché. La memoria caché guarda un registro de los archivos más frecuentemente usados y su localización, así puede ir directamente al lugar en que se encuentra almacenado sin necesidad de realizar una búsqueda en todas las particiones del disco.

En este punto es importante que el estudiante investigue y analice del propio libro de texto cuáles son algunos de los aspectos de protección del sistema de archivos, como por ejemplo los respaldos, analice su objetivo.

Siguiendo con el tema de la implementación o administración de archivos, el libro de texto menciona cuatro formas: una de ellas es la de archivos compartidos (sección 4.3.4), archivos estructurados por registro (sección 4.3.5), sistemas de archivos por bitácora (4.3.6) y los sistemas de archivos virtuales (4.3.7). El estudiante debe estudiar cada uno de estas secciones para ahondar en el tema.

4.4. Administración y Optimización de Sistemas de Archivos

La información en los discos es dinámica, los archivos crecen y decrecen si se agregan o se borran registros de un archivo. Otra situación que se da es que también se pueden agregar o borrar archivos. Cuando un archivo se borra, las particiones quedan disponibles, a estas particiones libres se les llama huecos. Cuando se quiere grabar un nuevo archivo, el sistema operativo buscará la forma de almacenar la información y, como mencionamos anteriormente, no siempre las particiones quedan una al lado de la otra, todo esto hace que el sistema operativo tenga que “ingeniárselas” para almacenar la información en las diferentes particiones para que a la hora de accederla, no haya problemas en su recuperación.

Estas particiones en que se divide el disco también se les llama bloques y el sistema operativo debe llevar un registro de qué bloques están libres para su reutilización.

Cuando un archivo es almacenado en diferentes bloques no continuos entre sí, la búsqueda de un dato se hace más lenta debido a que la cabeza lectora debe “saltar” de un lado para otro del disco para armar todo el archivo. Para hacer más eficiente la lectura y el acceso al archivo, se ejecuta un algoritmo que lo que hace es colocar todas las partes del archivo en



bloques continuos, a este proceso se le llama desfragmentación. La otra técnica para optimizar el acceso al archivo es el uso de la memoria caché.

Para ampliar sobre estos temas, el estudiante debe estudiar la sección 4.4, páginas 292 a la 312 del libro.

Ejercicios de autoevaluación

A continuación, se presentan ejercicios que tienen como objetivo evaluar cada uno de los puntos contenidos en el desarrollo del capítulo.

Para el capítulo 4, realice los siguientes ejercicios que se encuentran localizados en el libro de texto en las páginas 325-328: 2, 3, 4, 7, 14.

ENTRADA / SALIDA

5

Sumario

- *Principios de hardware*
- *Fundamentos de software*
- *Capas de software*
- *Discos*
- *Relojes*
- *Interfases de usuario*
- *Cientes delgados*
- *Administración de energía*



Objetivos de aprendizaje

Al finalizar el estudio de este tema, entre otras habilidades, usted será capaz de:

- ✓ Explicar los principios del hardware de E/S.
- ✓ Explicar los principios de software de E/S.
- ✓ Explicar el concepto de capas del software de E/S.
- ✓ Explicar los conceptos de Disco, relojes, teclados y pantallas como dispositivos de E/S.

Guía de lectura

Antes de realizar continuar con la lectura de esta guía, estudie detalladamente los siguientes apartados en el libro de texto:

- a. Principios de hardware de E/S, página 329.
- b. Fundamentos del software de E/S, página 343.
- c. Capas del software de E/S, página 348.
- d. Discos, página 360.
- e. Relojes, página 388
- f. Interfaces de usuario: teclado, ratón, monitor, página 394.
- g. Clientes delgados, página 415.
- h. Administración de energía, página 417.



Conceptos clave

Al finalizar el estudio de este capítulo usted deberá tener claros los siguientes conceptos:

- ✓ Dispositivos
- ✓ Controladores
- ✓ Acceso directo a memoria
- ✓ Uso de búffer
- ✓ Demonio
- ✓ Algoritmos de programación del brazo del disco
- ✓ Almacenamiento estable
- ✓ Pulsos de reloj

Aspectos sobresalientes

En el capítulo anterior hicimos énfasis en la implementación de archivos, de cómo estos se crean, se modifican y se eliminan, mencionamos el disco como el dispositivo principal de almacenamiento. No obstante, existe una gran cantidad de dispositivos que permiten la entrada y salida de información. Los más comunes son el teclado, el monitor, el “mouse”, los discos, las memorias flash, las impresoras, los scanners, etc.

Una de las principales funciones de un sistema operativo es controlar todos los dispositivos de entrada y salida de la computadora. Debe enviar comandos a los dispositivos, atrapar interrupciones y manejar errores. También, debe proporcionar una interfaz sencilla y fácil de usar entre los dispositivos y el resto del sistema.

El propósito de este capítulo es ocuparnos de la programación de los dispositivos de E/S, examinaremos algunos principios del hardware de E/S y estudiaremos el software de E/S general.



5.1. Principios de hardware de E/S

Los dispositivos de E/S pueden dividirse en dos categorías: **dispositivos de bloque** y **dispositivos de caracteres**. (Tanenbaum, 2009)

Un ejemplo de un dispositivo de bloque son los discos. Como vimos en el capítulo anterior, los discos se dividen en bloques de tamaño fijo para almacenar información.

Una impresora es un ejemplo de un dispositivo de carácter.

Mayor detalle sobre cada tipo de dispositivo se encuentra en la página 330 del libro de texto.

Los dispositivos de E/S son unidades compuestas por un componente mecánico y uno electrónico. El mecánico es el dispositivo en sí (disco duro, impresora, teclado, monitor, etc) y el componente electrónico es el software que lo controla, éste se denomina **controladora** o **adaptador de dispositivo**.

Cada controladora tiene unos cuantos registros que le sirven para comunicarse con la CPU. Para comunicar la CPU con los registros de control y los búfferes de datos de los dispositivos hay dos alternativas. La primera, a cada registro de control se le asigna un número de **puerto de E/S**; la segunda, consiste en establecer una correspondencia entre todos los registros de control y el espacio de memoria. A cada registro de control se le asigna una dirección de memoria única a la que no se asigna memoria. Este sistema se denomina **E/S con correspondencia en memoria**.

Tenga o no E/S con correspondencia en memoria, la CPU necesita direccionar las controladoras de dispositivos para intercambiar datos con ellas. La CPU puede solicitar datos a una controladora de E/S, byte por byte, pero ello obliga a la CPU a perder tiempo. Por esta razón, es común utilizar un esquema distinto, llamado **acceso directo a memoria (DMA, direct memory access)**.

Cuando un dispositivo de E/S termina el trabajo que se le encomendó, causa una interrupción. Esto se hace aplicando una señal a una línea de bus que se le haya asignado.



Una interrupción que deja a la máquina en un estado bien definido, se denomina **interrupción precisa**. Una interrupción de este tipo tiene cuatro propiedades:

1. El contador de programa (CP) se guarda en un lugar conocido.
2. Todas las instrucciones previas a aquella, a la que apunta el CP, ya se ejecutaron por completo.
3. No se ha ejecutado ninguna instrucción posterior a aquella a la que apunta el CP.
4. Se conoce el estado de ejecución de la instrucción a la que apunta el CP.

Una interrupción que no cumple con estos requisitos es una interrupción imprecisa.

Para profundizar en estos temas, refiérase a la sección 5.1 del libro de texto, páginas de la 329 a la 342.

5.2. Fundamentos de software

Un concepto clave en el diseño del software para E/S es el de **independencia del dispositivo**. Esto significa que debe ser posible escribir programas capaces de acceder a cualquier dispositivo de E/S, sin tener que especificar por adelantado de qué dispositivo se trata.

Algo estrechamente relacionado con la independencia del dispositivo es la meta de **nombres uniformes**. El nombre de un archivo o dispositivo deberá ser simplemente una cadena o un entero y no depender en absoluto del dispositivo.

Otro aspecto importante del software de E/S, es el **manejo de errores**. En general, los errores deben manejarse tan cerca del hardware como sea posible. Si la controladora descubre un error de lectura, deberá tratar de corregirlo ella misma si puede. Si no puede, el controlador de dispositivo deberá corregirlo, tal vez repitiendo el intento de leer el bloque.

Otro aspecto clave es la diferencia entre las transferencias **síncronas** (por bloqueo) y **asíncronas** (controladas por interrupciones). Casi toda la E/S física es síncrona: la CPU inicia la transferencia y se pone a hacer alguna otra cosa hasta que llega la interrupción.

Otra cosa que el software de E/S debe manejar es el **uso de búfferes**. Es común que los datos provenientes de un dispositivo no pueda almacenarse en forma directa en su destino final.

Existen tres formas fundamentalmente distintas de efectuar la E/S. La forma más sencilla deja que la CPU haga todo el trabajo. Este método se denomina **E/S programada**. La



segunda forma es la **E/S controlada por interrupciones**. Cuando la impresora ha impreso un carácter y está preparada para aceptar el siguiente, genera una interrupción, lo que detiene el proceso actual y guarda su estado. Luego se ejecuta el servicio de interrupciones. Una desventaja obvia es que se presenta una interrupción con cada carácter. La **E/S con DMA**. Aquí la idea consiste en dejar que la controladora DMA alimente los caracteres a la impresora uno por uno, sin molestar a la CPU. Básicamente, el DMA es E/S programada, solo que la controladora DMA es la que realiza todo el trabajo, no la CPU.

Para profundizar en estos temas, refiérase a la sección 5.2 del libro de texto, páginas de la 343 a la 347.

5.3. Capas de software de E/S

Aunque la E/S programada a veces es útil, en la mayoría de las operaciones de E/S las interrupciones son inevitables, por molestas que sean. Lo mejor es ocultarlas en las profundidades del SO, de modo que la parte de este último que tiene conocimiento de ellas sea lo más reducido posible. La mejor manera de ocultar las interrupciones es hacer que el controlador que inicia una operación de E/S se bloquee hasta que la E/S haya terminado y se presente la interrupción.

Cada dispositivo de E/S conectado a una computadora necesita código específico que sirva para controlar ese dispositivo. Este código, llamado **controlador de dispositivo** (device driver), por lo general es escrito por el fabricante del dispositivo y se proporciona junto con el hardware.

Para acceder al hardware del dispositivo, es decir, a los registros de la controladora, por lo general es necesario que el controlador forme parte del kernel del SO, al menos en las arquitecturas actuales.

Los sistemas operativos por lo regular clasifican los controladores en unas cuantas categorías. Las más comunes son los **dispositivos de bloques** como discos, que contienen varios bloques de datos susceptibles de direccionarse en forma independiente, y los **dispositivos de caracteres**, tales como: teclados, impresoras, que generan o aceptan un flujo de caracteres.



La función básica del software independiente del dispositivo es realizar las operaciones de E/S que son comunes a todos los dispositivos y presentar una interfaz uniforme al software de usuario.

Si la interfaz con los discos, impresoras, teclados, etc., es muy distinta para cada caso en particular, entonces cada vez que llegue un nuevo dispositivo será preciso modificar el sistema operativo para ese dispositivo. Esto no es muy recomendable.

(Para una mayor comprensión del tema, vea la figura 5-13 del libro de texto, página 353.)

Aunque casi todo el software de E/S está dentro del sistema operativo, una porción pequeña consiste en bibliotecas enlazadas con programas de usuario, e incluso en programas enteros que se ejecutan fuera del kernel. Los procedimientos de biblioteca generalmente emiten llamadas al sistema, entre las que se encuentran las de E/S.

No todo el software de E/S en el nivel de usuario consiste en procedimientos de biblioteca. Otra categoría importante es el sistema de spooling. El **spooling** es una forma de manejar dispositivos dedicados en un sistema con multiprogramación.

Para profundizar en estos temas, refiérase a la sección 5.3 el libro de texto, páginas de a 348 a la 360.

5.4. Discos

Ahora comenzaremos a estudiar algunos dispositivos reales de E/S. Los primeros son los discos.

Los discos son de diversos tipos. Los más comunes son los discos magnéticos (discos duros y disquetes). Estos se caracterizan por el hecho de que las lecturas y escrituras son igual de rápidas, lo que los hace ideales como memoria secundaria (paginación, sistemas de archivos, etc.).

Los discos magnéticos se organizan en cilindros, cada uno de los que tiene tantas pistas como haya cabezas apiladas en forma vertical. Las pistas se dividen en sectores.

Una característica del dispositivo que tiene implicaciones importantes para el controlador de disco es la posibilidad de que una controladora desplace las cabezas lectoras de dos o más unidades de disco, al mismo tiempo. Esto se denomina **desplazamiento de cabeza traslapado**. Mientras la controladora y el software están esperando que el desplazamiento



de la cabeza termine en una unidad, la controladora puede iniciar un desplazamiento de cabeza en otra unidad. Muchas controladoras también, pueden leer o escribir en una unidad mientras desplazan la cabeza en otra u otras unidades, pero una controladora de disquete no puede leer o escribir en dos unidades al mismo tiempo.

Un disco duro consiste en una pila de platos de aluminio. Para que pueda usarse el disco, cada plato debe recibir un **formato de bajo nivel** efectuado por el software. El formato consiste en una serie de pistas concéntricas, cada una de las que contiene cierto número de sectores, con espacios cortos entre ellos.

El tiempo que se toma para leer o escribir en un disco está determinado por tres factores:

1. Tiempo de desplazamiento (seek time), o sea, el tiempo que tarda el movimiento del brazo hasta el cilindro correcto.
2. Retraso rotacional, ó sea, el tiempo que tarda el sector correcto en girar hasta colocarse bajo la cabeza.
3. Tiempo real de transferencia de datos.

Los defectos de fabricación introducen sectores defectuosos. Existen dos enfoques generales para manejar los bloques defectuosos: ocuparse de ellos en la controladora o hacerlo en el sistema operativo. Con el primer enfoque, antes de que el disco salga de la fábrica se le prueba y se escribe en el disco una lista de sectores defectuosos.

En el segundo enfoque, si el sistema operativo se está encargando del ajuste de correspondencia, deberá asegurarse de que no haya sectores defectuosos en los archivos y tampoco en la lista o mapa de bits de sectores libres. Una forma de hacerlo es crear un archivo secreto integrado por todos los sectores defectuosos. Si este archivo no se incorpora al sistema de archivos, los usuarios no podrán leerlo accidentalmente o, lo que es peor, liberar su espacio.

En algunas aplicaciones es indispensable que los datos nunca se pierdan ni se corrompan, aunque se presenten errores de disco o de CPU. De manera ideal, un disco deberá trabajar todo el tiempo sin errores. Lo malo es que dicha meta es inasequible. Lo que sí es factible es tener un subsistema de disco con la siguiente propiedad: cuando se le ordena escribir algo, se escriben correctamente los datos o no se escribe nada, dejando los datos existentes intactos. Un sistema así se denomina **almacenamiento estable** y se implementa en software.

El almacenamiento estable utiliza un par de discos idénticos en el que los bloques correspondientes colaboran para formar un bloque sin errores. En ausencia de errores, los

bloques correspondientes en ambas unidades son iguales. Es posible leer cualquiera de ellos y obtener el mismo resultado. Para lograr esta meta, se definen tres operaciones: **escrituras estables, lecturas estables y recuperación después de caídas**.

Para profundizar en estos temas, refiérase a la sección 5.4 del libro de texto, páginas de la 360 a la 388.

5.5. Relojes

Los **relojes** (también llamados **temporizadores**) son indispensables para el funcionamiento de cualquier sistema multiprogramado por diversas razones. El software de reloj puede adoptar la forma de un controlador de dispositivo, aunque un reloj no es un dispositivo de bloques, ni un dispositivo de caracteres.

Por lo general, en las computadoras se usan dos tipos de reloj, ambos muy diferentes a los que usan las personas. Los más sencillos están conectados a la línea de alimentación eléctrica y causan una interrupción en cada ciclo de voltaje. Ahora éstos son poco comunes.

El otro tipo de reloj se construye con tres componentes: un cristal oscilador, un contador y un registro de retención. Este genera una señal periódica de gran precisión. Lo único que hace el hardware de reloj es generar interrupciones a intervalos conocidos, todo lo demás relacionado con el tiempo debe efectuarse en el software, con el controlador de reloj. Sus tareas son:

1. Mantener la hora del día.
2. Evitar que se ejecuten los procesos durante más tiempo del debido.
3. Contabilizar el consumo de CPU.
4. Procesar la llamada al sistema alarma emitida por procesos de usuario.
5. Proporcionar temporizadores de vigilancia a ciertas partes del sistema.
6. Realizar perfiles, supervisión y recolección de datos estadísticos.

Casi todas las computadoras tienen un segundo reloj programable que puede ajustarse de modo que cause interrupciones con la frecuencia que las necesite un programa. Este temporizador es adicional al temporizador principal del sistema.

Los **temporizadores de software** evitan interrupciones. Cada vez que el kernel se ejecuta por algún motivo, justo antes de volver al modo de usuario examina el reloj de tiempo real para ver si ha expirado un temporizador de software.



Para profundizar en estos temas, refiérase a la sección 5.5 del libro de texto, páginas de la 388 a la 394.

5.6. Interfaces de usuario, clientes delgados y administración de energía

Toda computadora de uso general tiene al menos un teclado y una pantalla que sirve para comunicarse con ella.

La tarea básica del controlador de teclado consiste en obtener entradas del teclado y pasarlas a los programas de usuario cuando estos lean de la terminal.

Las salidas son más sencillas que las entradas. En general, la computadora envía caracteres a la terminal, donde se exhiben.

Para profundizar en estos temas, refiérase a la sección 5.6 del libro de texto, páginas de la 394 a la 515.

Ejercicios de autoevaluación

A continuación, se presentan ejercicios que tienen como objetivo evaluar cada uno de los puntos contenidos en el desarrollo del capítulo.

Para el capítulo 5, realice los siguientes ejercicios que se encuentran localizados en el libro de texto en las páginas 427 a la 432: 1, 2, 3, 9, 14, 21, 23.

INTERBLOQUEOS

6

Sumario

- *Recursos*
- *Introducción a los interbloqueos*
- *El algoritmo del avestruz*
- *Detección y recuperación de un interbloqueo*
- *Cómo evitar interbloqueos*
- *Cómo prevenir interbloqueos*



Objetivos de aprendizaje

Al finalizar el estudio de este tema, entre otras habilidades, usted será capaz de:

- ✓ Explicar el concepto de recursos expropiables y no expropiables.
- ✓ Identificar las situaciones que pueden dar origen a un proceso irreversible.
- ✓ Identificar y analizar algunos algoritmos que se han diseñado para evitar los bloqueos irreversibles.
- ✓ Identificar y conocer los métodos que se han diseñado para la detección de este tipo de bloques.
- ✓ Identificar y analizar los algoritmos que se han diseñado para la recuperación posterior de los procesos bloqueados.
- ✓ Identificar y analizar algunos de las maneras en que se pueden prevenir los bloqueos irreversibles.

Guía de lectura

Antes de realizar continuar con la lectura de esta guía, estudie detalladamente los siguientes apartados en el libro de texto:

- ✓ Recursos, página 434.
- ✓ Introducción a los Interbloqueos, página 437.
- ✓ El algoritmo del avestruz, página 441.
- ✓ Detección y recuperación de un interbloqueo, página 443.
- ✓ Cómo evitar interbloqueos, página 448.
- ✓ Cómo prevenir interbloqueos, página 454.



Conceptos clave

Al finalizar el estudio de este capítulo usted deberá tener claros los siguientes conceptos:

- ✓ Recurso Apropiativo
- ✓ Recurso no apropiativo
- ✓ Vector de recursos existentes
- ✓ Vector de recursos disponibles

Aspectos sobresalientes

Los sistemas de cómputo abundan en recursos que sólo pueden ser utilizados por un proceso a la vez. Si dos procesos escriben de manera simultánea en la impresora, el resultado es basura. Por ello, todos los sistemas operativos tienen la prerrogativa de otorgar a un proceso acceso exclusivo a ciertos recursos.

En muchas aplicaciones, un proceso necesita acceso exclusivo no a un recurso, sino a varios. Cuando un proceso retiene un recurso por tiempo indefinido y otro recurso lo solicita, y en vez de liberarlo, solicita otro recurso y se mantienen bloqueados ambos procesos, a esta situación se le llama interbloqueo. (Tanenbaum, 2009). También pueden presentarse bloqueos entre máquinas.

El propósito de este capítulo es conocer los diferentes tipos de bloqueos irreversibles que se pueden dar (orígenes y características) y presentar una serie de algoritmos que se han diseñado para su prevención, detección y solución.

En Costa Rica todavía quedan en algunos puentes a través de los que solamente se puede transitar en una sola vía a la vez, por eso encontramos que en uno de sus puntos de acceso encontremos un rótulo que dice “CEDA”, esto quiere decir (para el que no lo sabe) que si dos vehículos convergen en el puente en forma opuesta y simultánea, el vehículo que tiene el ceda debe esperar a que el otro vehículo pase para luego pasar él. Si este rótulo no estuviera, cualquiera de los dos podría aducir que llegó un milisegundo antes y por eso



tiene prioridad de paso, esto ocasionaría caos vial, si vienen otros vehículos atrás. Ambos vehículos se bloquean el uno al otro. Esto se puede llamar **interbloqueo**.

Si trasladamos este ejemplo en nuestros términos informáticos, podemos decir que cada vehículo es un proceso y el puente es un recurso.

Adentrándonos en este capítulo, el asunto podría ir un poco más allá. Un proceso informático no solamente hace uso de un recurso a la vez, puede usar la memoria principal, puede usar el disco duro para leer o grabar datos, puede usar unidades de cinta impresoras a la vez. Pero si una vez utilizado el proceso el recurso no se libera, otros procesos pueden quedarse indefinidamente en espera de continuar.

6.1. Recursos

Como hemos visto en capítulos anteriores, el SO es el encargado de administrar los procesos y los recursos que se asignan a cada uno de ellos. Hay recursos que pueden ser usados de manera alterna entre varios procesos, otros se asignan a un proceso y no se liberan hasta que se haya finalizado completamente. Por ejemplo, el disco duro puede ser utilizado por varios procesos a la vez (recordemos lo de pseudoparalelismo) pero una impresora no, antes de imprimirse un nuevo documento, primero hay que esperar a que se finalice con el que se está imprimiendo.

Por ello, todos los sistemas operativos tienen la prerrogativa de otorgar a un proceso (en forma temporal) acceso exclusivo a ciertos recursos; los bloqueos irreversibles son un problema que puede presentarse en cualquier SO.

Cuando un programador diseña un programa, que al final puede convertirse en uno o más procesos, no piensa en la administración de recursos, simplemente pide y pide recursos y si el programador no tiene el cuidado, un proceso puede apropiarse de un sin número de recursos y liberarlos hasta que finalice, mientras tanto habrá una cola de otros procesos esperando turno.

En un párrafo anterior se mencionó la palabra recurso, un **recurso** puede ser un dispositivo de hardware o información. Un **recurso** es cualquier cosa que solo un proceso puede usar en un instante dado.

Es importante tener claro los siguientes conceptos:



Recurso expropiable: es aquel se le puede quitar al proceso que lo tiene sin causar daños. Ejemplo de ellos es la memoria principal.

Recurso no expropiable: no puede quitársele a su actual dueño sin hacer que el cómputo falle. Ejemplos de ellos un quemador de cd, una impresora.

El ciclo natural del uso de un recurso está constituido por tres pasos: solicitar el recurso, usar el recurso y liberar el recurso.

Para la administración de los recursos por parte del SO se han diseñado varios algoritmos con el fin de hacer el mejor uso de ellos y evitar los interbloqueos. Algunos de ellos son los semáforos y los mutexes

Para profundizar en el tema, refiérase a la sección 6.1 del libro de texto, páginas 434 a la 437.

6.2. Introducción a los interbloqueos

Hay un concepto que es más grave aún, se llama **bloqueo irreversible**, ocurre si cada proceso del conjunto está esperando un suceso que solo otro proceso del conjunto puede causar.

Dos pintores llegan a una casa, solamente hay una brocha y un tarro de pintura, el primer pintor llega y toma la brocha y se va para la pila a lavarla, mientras está en esa tarea, el otro pintor toma el tarro de pintura y se va para otra habitación pensando que allí estaría la brocha. El que tiene el tarro de pintura no puede empezar esperando que el otro le lleve la brocha y el que tiene la brocha no puede empezar esperando que el otro le lleve la pintura. En síntesis, ninguno podrá iniciar hasta que el otro libere el recurso. (Obviamente en la vida real las personas conversan y se ponen de acuerdo y remedian rápidamente el problema).

Para que se dé un bloqueo irreversible, tienen que darse cuatro condiciones:

- Condición de exclusión mutua: cada recurso está asignado a exactamente un proceso o está disponible.
- Condición de retención y espera: los procesos que tienen recursos previamente otorgados pueden solicitar otros recursos.
- Condición de no expropiación: los recursos ya otorgados no pueden arrebatarse al proceso que los tiene; este debe liberarlos en forma explícita.



- Condición de espera circular: debe haber una cadena circular de dos o más procesos, cada uno de los cuales está esperando un recurso que está en manos del siguiente miembro de la cadena.

(El **modelado de bloqueos irreversibles** puede visualizarse en las figuras 6-3 y 6-4, páginas 439 y 440, respectivamente, del libro de texto.)

6.3. Detección y recuperación de un interbloqueo

Para prevenir los bloques irreversibles se pueden seguir las siguientes estrategias:

- Simplemente ignorar el problema: quizá si nos olvidamos de él, él se olvidará de nosotros (analizar la sección 6.3, página 441 del libro de texto).
- Detección y recuperación: dejar que se presenten bloqueos irreversibles, detectarlos y tomar medidas (analizar la sección 6.4, página 442 del libro de texto).
- Evitación dinámica: mediante una asignación cuidadosa de recursos (analizar la sección 6.5, página 448 del libro de texto)
- Prevención, anulando en forma estructural una de las cuatro condiciones necesaria para que haya un bloqueo irreversible (analizar la sección 6.6, página 454 del libro de texto).

La mayoría de los sistemas operativos pueden padecer bloqueos irreversibles que ni siquiera se detectan. Una estrategia razonable para el programa que emitió la llamada sería esperar un tiempo aleatorio e intentarlo, si no se obtiene el recurso nuevamente esperar un tiempo aleatorio y volver a intentarlo y así hasta que en una de tantas el proceso se apropie del recurso esperado (**algoritmo del avestruz**).

Existen varios métodos para la detección de bloqueos irreversibles, los que se mencionan a continuación:

- **Detección de bloqueos irreversibles con un recurso de cada tipo.**

(Analizar figura 6-5 y estudiar el algoritmo sugerido en la página 443 del libro de texto.)

- **Detección de bloqueos irreversibles con múltiples recursos de cada tipo.**

Para **recuperarse de un bloqueo irreversible**, hay varias maneras que a continuación se mencionan:



- **Recuperación mediante expropiación.**
- **Recuperación mediante reversión.**
- **Recuperación de eliminación de procesos.**

(Para ahondar más en el tema, refiérase a las páginas 442 a la 448 del libro de texto.)

6.4. Cómo evitar interbloqueos

Para evitar los **procesos irreversibles**: los principales algoritmos se basan en el concepto de **estados seguros**. Es importante que el estudiante analice los conceptos de estado seguro y estado inseguro.

El algoritmo del banquero y del banqueo para múltiples recursos, ayudan a evitar los bloqueos irreversibles.

(Para ahondar más en el tema, refiérase a las páginas 448 a la 454 del libro de texto.)

6.5. Como prevenir interbloqueos

Si se quiere atacar la condición de exclusión mutua se tienen que evitar asignar un recurso si no es absolutamente necesario, y tratar de asegurarse de que el número de procesos que podrían solicitar el recurso, sea lo más pequeño posible.

Para atacar la condición de retener y esperar, se debe evitar que los procesos que tienen recursos esperen cuando tratan de adquirir más recursos podremos evitar los bloqueos irreversibles.

Si uno o más recursos están ocupados, no se asignará nada y el proceso tendrá que esperar.

(Para atacar la condición de no expropiación, estudie la sección 6.6.3, página 455 del libro de texto.)

Para atacar la condición de espera circular existen varias formas: una sería simplemente tener una regla en el sentido de que cualquier proceso tiene derecho tan solo a un recurso en todo momento. Si necesita un segundo recurso, deberá liberar el primero.



Otra forma de evitar la espera circular es con una numeración global de todos los recursos. Los procesos pueden solicitar recursos cuando los necesiten, pero todas las solicitudes deben efectuarse en orden numérico. (Ver figura 3-14, página 457 del libro de texto.)

Los bloqueos de dos fases: en muchos sistemas de bases de datos, una operación que se presenta con frecuencia es solicitar bloqueos para varios registros y luego actualizar todos esos registros. Si se están ejecutando varios procesos al mismo tiempo, el peligro de un proceso irreversible es muy real.

El bloqueo irreversible que no son por recursos: puede suceder que dos procesos se bloqueen mutuamente esperando que el otro haga algo

La inanición se puede entender como una espera pasiva de un recurso, no hay prisa, si el recurso está disponible lo uso, si no, el proceso entra en un letargo hasta que el recurso le sea asignado. Esta política, aunque al parecer es razonable, puede hacer que algunos procesos nunca sean atendidos, aunque no hayan caído en un bloqueo irreversible.

La inanición puede evitarse utilizando una política de asignación de recursos tipo primero que llega, primero que se atiende. Así, el proceso que ha esperado más tiempo será el que reciba servicio a continuación.

(Para ampliar mejor los conocimientos acerca de los temas aquí expuestos, refiérase a las secciones 6.5, 6.6 y 6.7, páginas 448 a la 461 del libro de texto.)

Ejercicios de autoevaluación

A continuación, se presentan ejercicios que tienen como objetivo evaluar cada uno de los puntos contenidos en el desarrollo del capítulo.

Para el capítulo 6, realice los siguientes ejercicios que se encuentran localizados en el libro de texto en las páginas 463 a la 466.

Sumario

- *El entorno de la seguridad*
- *Fundamentos de la criptografía*
- *Mecanismos de protección*
- *Autenticación*
- *Ataques desde el interior*
- *Como explotar los errores en el código*
- *Malware*
- *Defensas*





Objetivos de aprendizaje

Al finalizar el estudio de este tema, entre otras habilidades, usted será capaz de:

- ✓ Explicar lo referente al entorno de la seguridad.
- ✓ Explicar los aspectos básicos de criptografía.
- ✓ Explicar lo concerniente a la autenticación de usuarios.
- ✓ Explicar lo referente a los ataques desde dentro del sistema.
- ✓ Explicar lo referente a los ataques desde afuera del sistema.
- ✓ Explicar lo referente a mecanismos de protección.

Guía de lectura

Antes de realizar continuar con la lectura de esta guía, estudie detalladamente los siguientes apartados en el libro de texto:

- a. El entorno de la seguridad, página 613.
- b. Fundamentos de la criptografía, página 616.
- c. Mecanismos de protección, página 622.
- d. Autenticación, página 641.
- e. Ataques desde el interior, página 656.
- f. Como explotar los errores en el código, página 659.
- g. Malware, página 667.
- h. Defensas, página 692.



Conceptos clave

Al finalizar el estudio de este capítulo usted deberá tener claros los siguientes conceptos:

- ✓ Dispositivos
- ✓ Controladores
- ✓ Acceso Directo a Memoria
- ✓ Uso de búfer
- ✓ Demonio
- ✓ Algoritmos de programación del brazo del disco
- ✓ Almacenamiento estable
- ✓ Pulsos de reloj

Aspectos sobresalientes

Muchas compañías poseen información valiosa que resguardan celosamente. Dicha información puede ser técnica, comercial, financiera, legal, entre muchas otras posibilidades.

A medida que aumenta la fracción de esta información que se almacena en sistemas de computación, crece en importancia la necesidad de protegerla. Por ello, la protección de esta información contra un uso no autorizado es una tarea fundamental para todos los sistemas operativos.

7.1. El entorno de la seguridad

Utilizaremos el término **seguridad** para referirnos al problema general, y el término **mecanismos de protección** para referirnos a los mecanismos específicos del sistema operativo que sirven para salvaguardar la información en la computadora.



Desde una perspectiva de seguridad, los sistemas de computación tienen tres metas generales, con sus respectivas amenazas (como se muestra en la figura 9-1, en la página 584 del libro de texto).

La primera, la **confiabilidad de los datos**, tiene que ver con mantener en secreto los datos. La segunda meta, la **integridad de los datos**, implica que los usuarios no autorizados no podrán modificar ningún dato sin permiso del dueño. La tercera meta, **disponibilidad del sistema**, implica que nadie podrá alterar el sistema de modo que no pueda usarse.

Otro aspecto del problema de la seguridad es la **privacidad**: proteger a las personas contra el mal uso de su información personal.

A cualquier persona que se mete en datos que no le incumbe, se le denomina **intruso**. Estos actúan de dos maneras. Pasivamente, sólo quiere leer archivos que no está autorizado. Los intrusos activos tienen peores intenciones: quieren efectuar cambios no autorizados a los datos.

Entre las categorías más comunes de intrusos, están las siguientes:

1. Curiosear casual por parte de los usuarios no técnicos
2. Husmeo por parte de personal interno
3. Intentos decididos por hacer dinero
4. Espionaje comercial o militar

Entre las causas más comunes de la pérdida accidental de datos, están:

1. Actos fortuitos
2. Errores de hardware o software
3. Errores humanos

7.2. Fundamentos de la criptografía

El propósito de la **criptografía** es tomar un mensaje o un archivo, llamado **texto simple**, y convertirlo en **texto cifrado**, de tal manera que solo las personas autorizadas puedan convertirlo otra vez en texto simple. Esta táctica se conoce como **seguridad de ocultamiento**, y solo los aficionados en materia de seguridad la utilizan.



Muchos sistemas criptográficos tienen la propiedad de que, dada la clave de cifrado, es fácil deducirla, y viceversa. Tales sistemas se llaman **criptografía de clave secreta** o **criptografía de clave simétrica**.

La **criptografía de clave pública** tiene la propiedad de que se usan claves distintas para descifrar y cifrar. Si se escoge bien la clave de cifrado, es casi imposible descubrir a partir de ella, la clave de descifrado correspondiente.

Cuando un usuario inicia una sesión en una computadora, lo normal es que el sistema operativo quiera determinar quién es el usuario. Este proceso se denomina **autenticación de usuarios**.

La forma de autenticación más utilizada consiste en pedir al usuario que teclee un nombre de inicio de sesión y una contraseña. La protección por contraseña es fácil de entender y de implementar.

Casi todos los crackers se introducen al sistema objetivo estableciendo una conexión con éste y probando muchas combinaciones (inicio de sesión, contraseña) hasta hallar una que funcione.

El programa que invocan los usuarios para instalar o cambiar su contraseña también puede prevenir al usuario de que se está escogiendo una mala contraseña. Entre las cosas por las que podría emitir una advertencia están:

1. Las contraseñas deben tener como mínimo siete caracteres.
2. Deben contener letras tanto mayúsculas como minúsculas.
3. Deben contener al menos un dígito o carácter especial.
4. No deben ser palabras del diccionario, nombres de personas, etc.

Un segundo método para autenticar usuarios consiste en examinar algún objeto físico que poseen en lugar de algo que saben. El objeto físico empleado a menudo es una tarjeta de plástico que se inserta en un lector enlazado con la terminal o la computadora. Por lo regular, el usuario no solo debe insertar la tarjeta sino también teclear una contraseña, para evitar que alguien use una tarjeta extraviada o robada.

Un tercer método de autenticación mide características físicas del usuario que son difíciles de falsificar. Esto se conoce como **biométrica**. Por ejemplo, un lector de huellas dactilares o de patrón de voz en la terminal, podría verificar la identidad del usuario.



(Para ahondar más en este tema, estudie las secciones 9-1, 9-2 y 9-3 del libro de texto, páginas de la 583 a la 606.)

Una vez que un cracker ha iniciado sesión en una computadora, puede comenzar a causar daños.

Un añejo ataque desde dentro es el **caballo de Troya**. Es un programa al parecer inocente, que contiene código para realizar una función inesperada e indeseable. Dicha función podría consistir en modificar, borrar o cifrar los archivos del usuario, por ejemplo.

Algo con cierta relación con los caballos de Troya es la **falsificación de inicio de sesión**, que funciona como sigue: se utiliza una pantalla similar a la de inicio de sesión, el usuario teclea su contraseña e identificación. Al ser digitadas, se guardan en un archivo en donde se recolectan estos datos y podrán ser usados posteriormente por un intruso.

Otro ataque desde el interior es la **bomba lógica**. Este dispositivo es un fragmento de código escrito por uno de los programadores de una compañía e insertado de manera subrepticia en el sistema operativo de producción. Mientras este programa reciba la contraseña del programador diariamente no pasará nada. Pero si el programador es despedido, el programa, al no recibir la contraseña, en cualquier momento puede “estallar”.

Otra brecha de seguridad de origen interno es la trampa. Este problema se presenta cuando un programador del sistema inserta código que permite pasar por alto alguna verificación normal. Por ejemplo, podría incluir una sección de código que se salte la verificación de usuario para una contraseña dada que sólo el programador sabe.

El proceder normal para probar la seguridad de un sistema consiste en contratar a un grupo de expertos, conocido como **equipos tigre** o **equipos de penetración**, que tratarán de introducirse en el sistema.

(Para conocer más en este tema estudie la página 613 del libro de texto.).

(Para ahondar más en estos temas, estudie la sección 9-4 del libro de texto, páginas de la 606 a la 614.)



7.3. Mecanismos de protección

En el caso de máquinas conectadas a Internet o a otra red, existe una creciente amenaza externa. Una computadora en una red puede ser atacada desde una computadora distante a través de la red. En casi todos los casos, semejante ataque consiste en transmitir cierto código por la red a la máquina objetivo, donde se ejecutará y causará daños.

Un **virus** es un programa que puede reproducirse anexando su código a otro programa, de forma análoga a como se producen los virus biológicos. Veamos cómo funcionan los virus. Virgilio escribe su virus, tal vez en lenguaje ensamblador, y luego lo inserta con cuidado en su propia máquina utilizando una herramienta llamada **colocador**. Luego se distribuye ese programa infectado, quizá publicándolo en un tablero de boletines o en una colección de software gratuito por Internet.

El **gusano de Internet** se dio cuando un estudiante descubrió dos errores de programación en Berkeley UNIX, que permitían obtener acceso no autorizado a máquinas por toda la Internet. Trabajando solo, escribió un programa que se reproducía a sí mismo, un gusano que aprovechaba esos errores y se reproducía en segundos dentro de cada máquina a la que podía obtener acceso.

En algunos sistemas, la protección se implementa con un programa llamado **monitor de referencias**. Cada vez que se intenta tener acceso a un recurso que podría estar protegido, el sistema pide al monitor de referencias verificar antes si está permitido. El monitor de referencias consulta entonces sus tablas de políticas y toma una decisión.

Un **dominio** es un conjunto de pares (objeto, derechos). Cada par especifica un objeto y algún subconjunto de operaciones que pueden ejecutarse con él. Un **derecho** en este contexto implica permiso para realizar una de las operaciones.

Es común que un dominio corresponda a un solo usuario, e indique lo que el usuario puede y no puede hacer con él.

(Para ahondar más en este tema, estudie las secciones 9-5 y 9-6 del libro de texto, páginas de la 617 a la 653.)



7.4. Ataques desde el interior

La razón por la que no se están construyendo sistemas seguros es más complicada, pero se reduce a dos cuestiones fundamentales. Primera, los sistemas actuales no son seguros pero los usuarios no están dispuestos a deshacerse de ellos.

El segundo problema es más sutil. La única forma de construir un sistema seguro es que sea sencillo. Las funciones son enemigas de la seguridad. Los diseñadores creen que los usuarios quieren más funciones. Más funciones implican más complejidad, más código, más errores de programación y más fallas de seguridad.

En el mundo de la seguridad se habla a menudo de **sistemas de confianza** más que de sistemas seguros. Estos son sistemas que han planteado formalmente requisitos de seguridad y cumplen con ellos. En el corazón de todo sistema de confianza está una **base de computo de confianza**, esta consiste en el hardware y el software necesarios para hacer cumplir todas las reglas en materia de seguridad.

Si la base de cómputo de confianza opera según las especificaciones, la seguridad del sistema no podrá ser violada, sin importar que lo demás esté mal.

Casi todos los sistemas operativos permiten a usuarios individuales determinar quién puede leer y escribir sus archivos y otros objetos. Esta política se denomina **control de acceso a discreción**.

(Para profundizar en este tema, estudie las secciones 9-7 del libro de texto, páginas de la 653 a la 666.)

Ejercicios de autoevaluación

A continuación, se presentan ejercicios que tienen como objetivo evaluar cada uno de los puntos contenidos en el desarrollo del capítulo.

Para el capítulo 9, realice los siguientes ejercicios que se encuentran localizados en el libro de texto en las páginas 667, 668, 669 y 670: 9, 14, 15, 9, 14, 15, 17, 27.

DISEÑO DE SISTEMAS OPERATIVOS

8

Sumario

- *La naturaleza del problema de diseño*
- *Diseño de interfaces*
- *Implementación*
- *Rendimiento*
- *Administración de proyectos*
- *Tendencias en el diseño de sistemas operativos*



Objetivos de aprendizaje

Al finalizar el estudio de este tema, entre otras habilidades, usted será capaz de:

- ✓ Explicar la naturaleza del problema de diseño.
- ✓ Explicar lo referente a diseño de interfaces.
- ✓ Explicar los pasos de la implementación.
- ✓ Explicar lo referente al desempeño.
- ✓ Conocer acerca de la administración de proyectos.

Guía de lectura

Antes de realizar continuar con la lectura de esta guía, estudie detalladamente los siguientes apartados en el libro de texto:

- ✓ La naturaleza del problema de diseño, página 960.
- ✓ Diseño de interfaces, página p63.
- ✓ Implementación, página 971.
- ✓ Rendimiento, página 987.
- ✓ Administración de proyectos, página 994.
- ✓ Tendencias en el diseño de sistemas operativos, página 998.



Conceptos clave

Al finalizar el estudio de este capítulo usted deberá tener claros los siguientes conceptos:

- ✓ Dispositivos
- ✓ Controladores
- ✓ Acceso Directo a Memoria
- ✓ Uso de búfer
- ✓ Demonio
- ✓ Algoritmos de programación del brazo del disco
- ✓ Almacenamiento estable
- ✓ Pulsos de reloj

Aspectos sobresalientes

Una vez que usted ha realizado las lecturas anteriores, analice cuidadosamente los comentarios siguientes, tomados del libro de texto, ya que con ellos se pretende enfatizar o ampliar algunos contenidos importantes del tema. Existen varios aspectos importantes que debemos señalar, antes de entrar al desarrollo de los subtemas.

8.1. La naturaleza del problema de diseño

Para poder diseñar con éxito un sistema operativo, los diseñadores deben tener una idea clara de lo que quieren. La falta de una meta dificulta sobremanera la toma de decisiones subsiguientes.



En el caso de los sistemas operativos de propósito general, existen cuatro objetivos principales:

1. Definir abstracciones.
2. Proporcionar operaciones primitivas.
3. Garantizar el aislamiento.
4. Administrar el hardware.

La tarea más difícil es definir las abstracciones correctas. Algunas de ellas, como los procesos y archivos, se han usado desde hace tanto tiempo que podrían parecer obvias. Otras, como los subprocesos, son más recientes y menos maduras.

Cada una de las abstracciones puede ilustrarse en forma de estructuras de datos concretas. Los usuarios pueden crear procesos, archivos, semáforos, etc. Las operaciones primitivas manipulan estas estructuras de datos.

Puesto que puede haber múltiples usuarios en sesión al mismo tiempo en una computadora, el sistema operativo debe proporcionar mecanismos para mantenerlos separados. Un usuario no debe interferir con otro.

Una meta clave del diseño de sistemas es asegurarse de que cada usuario sólo pueda ejecutar operaciones autorizadas con datos autorizados.

El sistema operativo tiene que administrar el hardware. En particular, tiene que ocuparse de todos los chips de bajo nivel, como las controladoras de interrupciones y las controladoras de bus. También tiene que proporcionar un marco en el que las controladoras de dispositivos puedan controlar los dispositivos de E/S más grandes, tales como: discos, impresoras y la pantalla.

Mencionemos ocho de las cuestiones que hacen que el diseño de un sistema operativo sea mucho más difícil que el de un programa de aplicación:

1. Los sistemas operativos se han convertido en programas extremadamente grandes. Ninguna persona puede sentarse frente a una PC y producir un sistema operativo serio en unos cuantos meses.



2. Los sistemas operativos tienen que manejar concurrencia. Existen múltiples usuarios y múltiples dispositivos de E/S, todos los que están activos a la vez. Administrar concurrencias es, por naturaleza, mucho más difícil que administrar una sola actividad secuencial.
3. Los sistemas operativos tienen que enfrentar a usuarios hostiles en potencia. El sistema operativo necesita tomar medidas para impedir que esos usuarios se comporten de manera indebida.
4. A pesar de que no todos los usuarios confían en los demás, muchos sí quieren compartir parte de su información y recursos con otros usuarios selectos. El sistema operativo tiene que posibilitar esto, pero de tal manera que los usuarios malintencionados no puedan interferir.
5. Los sistemas operativos tienen una vida larga. Por ello, los diseñadores tienen que tomar en cuenta la manera en que el hardware y las aplicaciones van a cambiar en el futuro y cómo deben prepararse para tales cambios.
6. Los diseñadores de sistemas operativos en realidad no tienen una idea muy clara de cómo van a usar sus sistemas, por lo que necesitan incorporar un alto grado de generalidad en ellos.
7. Los sistemas operativos modernos suelen diseñarse de modo que sean portables, lo que significa que tienen que funcionar en múltiples plataformas de hardware. También tienen que reconocer cientos o incluso miles de dispositivos de E/S, los que se diseñan en forma independiente.
8. Y, por último, la necesidad frecuente de ser compatible con algún sistema operativo anterior. Ese sistema pudo haber tenido restricciones en cuanto a longitud de palabras, nombre de archivos u otros aspectos que los diseñadores ahora consideran obsoletos, pero de los que no pueden liberarse.

(Para ahondar más en este tema, estudie la sección 13.1 del libro de texto, páginas de la 960 a la 962)



8.2. Diseño de interfases

Un sistema operativo presta un conjunto de servicios, en su mayoría tipos de datos (por ejemplo, archivos) y operaciones que se realizan con ellos. Juntos constituyen la interfaz con los usuarios.

Un adecuado **diseño de interfaces** debe seguir ciertos principios. Las interfaces tienen que ser sencillas, completas y susceptibles de implementarse de manera eficiente.

Principio 1: Sencillez. Una interfaz sencilla es más fácil de entender e implementar sin errores

Principio 2: Integridad. La interfaz debe permitir hacer todo lo que los usuarios necesitan hacer, es decir, tiene que ser completa o íntegra.

Principio 3: Eficiencia. Si una característica o llamada al sistema no puede implementarse de forma eficiente, quizá no vale la pena tenerla. También debe ser intuitivamente obvio para el programador cuánto cuesta una llamada al sistema.

Una vez establecidas las metas, puede comenzar el diseño. Un buen punto de partida es pensar en cómo van a ver el sistema los clientes. Una de las cuestiones más importantes es cómo hacer que todas las características del sistema sean un conjunto consistente, y presenten lo que se conoce como **coherencia arquitectónica**.

Es este sentido, es importante distinguir dos tipos de **clientes de los sistemas operativos**. Por un lado están los **usuarios**, que interactúan con programas de aplicación; por el otro lado están los **programadores**, que los escriben. Los usuarios tratan sobre todo con la GUI. Los programadores tratan principalmente con la interfaz de llamadas al sistema.

Tanto para la interfaz en el nivel GUI como para la interfaz de llamadas al sistema, el aspecto más importante es tener un buen **paradigma** que proporcione una forma de ver la interfaz. Por ejemplo, el paradigma WIMP utiliza acciones de apuntar y hacer clic, apuntar y hacer doble clic, arrastrar, entre otras, para conferir una coherencia arquitectónica a la totalidad.



Sea cual sea el paradigma que se escoja, es importante que lo usen todos los programas de aplicación. Por ello, los diseñadores de sistemas deben proporcionar a los creadores de aplicaciones, bibliotecas y conjuntos de herramientas con los que puedan acceder a procedimientos que produzcan el aspecto y el manejo uniformes.

Existen dos paradigmas de ejecución que se usan en forma amplia: el algoritmo y el controlado por sucesos.

El **paradigma algorítmico** se basa en la idea de que un programa se inicia para realizar alguna función que conoce con antelación o que obtiene de sus parámetros. La lógica básica viene incorporada en el código, y el programa emite llamadas al sistema cada cierto tiempo para obtener entradas del usuario, solicitar servicios del sistema operativo, etc.

El otro paradigma de ejecución es el **paradigma controlado por sucesos**. Aquí el programa realiza algún tipo de preparación inicial, digamos exhibir cierta pantalla, y luego espera a que el sistema operativo le notifique del primer suceso. Muchas veces el suceso es la pulsación de una tecla o un movimiento del ratón. Este diseño es muy útil para los programas muy interactivos.

Otro paradigma igualmente importante es el **paradigma de datos**. Se refiere a datos en cinta, archivos, objetos o documentos. En los cuatro casos, la intención es unificar los datos, dispositivos y demás recursos para que sea más fácil manejarlos. Todo sistema operativo debe tener un paradigma de datos que lo unifique.

El sistema operativo debería ofrecer el menor número posible de llamadas al sistema, y cada una deberá ser lo más sencilla posible.

En algunos casos, podría parecer que las llamadas al sistema necesitan ciertas variantes, pero muchas veces es más recomendable tener una sola llamada al sistema que maneje el caso general, con diferentes procedimientos de biblioteca para ocultar ese hecho a los programadores.

Si el hardware cuenta con un mecanismo muy eficiente para hacer algo, debe exponerse a la vista de los programadores de forma sencilla, y no enterrarlo en alguna otra abstracción. El propósito de las abstracciones es ocultar propiedades indeseables, no ocultar las deseables.

(Para ahondar más en este tema, estudie la sección 13.2 del libro de texto, páginas de la 963 a la 968).



8.3. Implementación

Dejando a un lado las interfaces y las llamadas al sistema, examinemos ahora cómo puede **implementarse un sistema operativo**.

Un método razonable que se ha establecido bien con el paso de los años es el **sistema de capas**. En el caso de un sistema nuevo, los diseñadores que decidan seguir esta ruta primero deberán escoger con mucho cuidado las capas y definir la funcionalidad de cada una de ellas. (Estudie lo referente a este tema en las páginas 971 y 974 del libro de texto.)

Aunque la estructura de capas tiene partidarios entre los diseñadores del sistema, también existe una fracción que adopta una perspectiva diametralmente opuesta. Su punto de vista se basa en el **argumento de extremo a extremo**. Este concepto dice que si el programa de usuario tiene que hacer algo él mismo, es un desperdicio hacerlo también en una capa más baja.

Un término medio entre hacer que el sistema operativo se encargue de todo o que no haga nada, es que el SO haga un poquito. Este diseño, llamado **cliente-servidor**, lleva a un microkernel, con una buena parte del sistema operativo ejecutándose como procesos servidores en el nivel de usuario. Este es el modelo más modular y flexible de todos. Lo fundamental en flexibilidad, es que cada controlador de dispositivo se ejecute también como proceso de usuario, protegido en forma plena contra el kernel y otros controladores. El problema principal con este método, y con los microkernels en general, es la merma en desempeño causada por todas las conmutaciones de contexto adicionales.

En los sistemas cliente-servidor se buscaba aprovechar el kernel al máximo. El enfoque opuesto consiste en colocar más módulos en el kernel, pero de forma protegida.

Otra cuestión pertinente aquí es la de los subprocesos de sistema, sea cual sea el modelo de estructuración escogido. A veces conviene permitir la existencia de subprocesos de kernel independientes de cualquier proceso de usuario. Estos procesos pueden ejecutarse en segundo plano. (Para ahondar más en este tema, estudie la sección 13.3 del libro de texto, páginas de la 971 a la 981).



8.4. Rendimiento

En condiciones iguales, un sistema operativo rápido es mejor que uno lento. Sin embargo, un sistema operativo rápido y poco confiable no es tan bueno como uno lento pero confiable.

Tal vez lo más importante que los diseñadores de sistemas pudieran hacer para mejorar el desempeño sería mostrarse mucho más selectivos en cuanto a añadir nuevas funciones y características.

Un método general para mejorar el desempeño es sacrificar tiempo a cambio de espacio. Al efectuar una optimización importante, conviene buscar algoritmos que ganen velocidad ocupando más memoria o, por otro lado, que ahorren memoria escasa realizando más cálculos.

Una técnica muy conocida para mejorar el desempeño es el uso de **cachés**. La estrategia general consiste en efectuar todo el trabajo la primera vez y luego guardar el resultado en un caché. En los siguientes intentos, primero se verifica el caché. Si el resultado está ahí, se usa; si no, se vuelve a realizar todo el trabajo.

Las **entradas de caché** siempre son correctas. Una búsqueda en caché podría fallar, pero si encuentra una entrada, se garantiza que es correcta y puede usarse sin más. En algunos sistemas conviene tener una **tabla de sugerencias**. Estas son sugerencias en cuanto a la solución, pero no se garantiza que sean correctas.

Los procesos y programas no actúan al azar: exhiben un alto grado de localidad en el tiempo y en el espacio, y existen varias formas de aprovechar esta información para mejorar el desempeño. El **principio de localidad** también es válido para los archivos. Una vez que un proceso ha seleccionado su directorio de trabajo, es probable que muchas de sus referencias futuras sean a archivos que están en ese directorio.

Otra área en la que la localidad desempeña un papel es la **calendarización de subprocesos en multiprocesadores**. Una forma de calendarizar subprocesos en un multiprocesador es tratar de ejecutar cada subproceso en la última CPU que utilizó, con la esperanza de que algunos de sus bloques de memoria todavía estén en el caché de memoria.

En muchos casos es recomendable distinguir entre el caso más común y el peor caso posible y tratarlos de distinta manera. Con frecuencia el código para los dos casos es muy diferente.



Es importante lograr que el caso común sea rápido. El peor caso, si no se presenta a menudo, solo tiene que manejarse en forma correcta.

(Para ahondar más en este tema, estudie la sección 13.4 del libro de texto, páginas de la 987 a la 993).

Ejercicios de autoevaluación

A continuación, se presentan ejercicios que tienen como objetivo evaluar cada uno de los puntos contenidos en el desarrollo del capítulo.

Para el capítulo 13, realice los siguientes ejercicios que se encuentran localizados en el libro de texto en las páginas 1003 a la 1006.



RESPUESTAS A LOS EJERCICIOS DE AUTOEVALUACIÓN

Tema 1

Ejercicio 1

¿Qué es multiprogramación?

La multiprogramación se dio a partir de la tercera generación y consiste en dividir la memoria en varias partes, con un trabajo distinto en cada partición. Mientras un trabajo estaba esperando a que terminara la E/S, otro podría estar usando la CPU.

Ejercicio 2

¿Qué es spooling? ¿Cree que las computadoras personales avanzadas lo usarán como función estándar en el futuro?

El spooling consiste en cargar un trabajo nuevo del disco en la partición recién desocupada para ejecutarlo. Esta característica se utiliza tanto para entrada de datos como en la salida.

El avance de la informática ha demostrado que el tiempo del CPU es cada vez más valioso y la oportunidad con que se realicen los trabajos toma mayor relevancia. Las computadoras personales son muy potentes y es importante aprovechar al máximo este recurso. Ya el spooling se utiliza en las salidas (lista de impresión), así que es posible utilizar un sistema similar para la entrada de datos.

**Ejercicio 14**

¿Cuál es la diferencia clave entre una interrupción de sistema (TRAP) y una interrupción normal?

La diferencia básica está en que una instrucción TRAP cambia del modo de usuario al modo kernel e inicia el SO. Cuando un sistema de usuario quiere obtener servicios del SO, se provoca una llamada al sistema. Las interrupciones normales las hace el hardware, generalmente cuando se presenta una situación excepcional, como una división por cero. En ambos casos, el control lo asume el SO.

Ejercicio 17

¿Cuál es el propósito de una llamada al sistema en un sistema operativo?

Básicamente, el propósito de una llamada al sistema es solicitarle al SO recursos o una tarea específica que el proceso por sí mismo no puede realizar, por ejemplo, abrir o cerrar archivos, lectura o escritura de datos de un medio de almacenamiento, tiempo de procesador, entre otros.

Ejercicio 22

¿Qué diferencia fundamental existe entre un archivo especial de bloques y un archivo especial de caracteres?

Los primeros sirven para modelar dispositivos que constan de una colección de bloques direccionales en forma aleatoria, como los discos. Los segundos se usan para modelar impresoras, módems y otros dispositivos que aceptan o producen flujos de caracteres.

Ejercicio 24

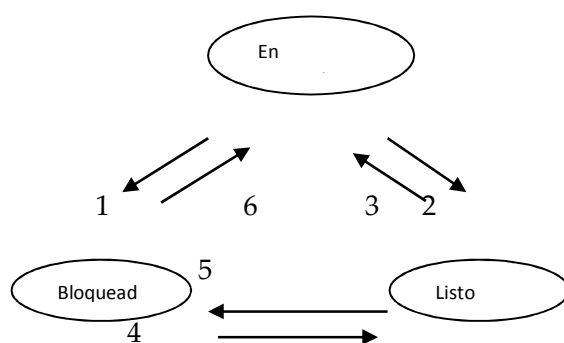
El modelo cliente-servidor es muy utilizado en sistemas distribuidos. ¿Puede utilizarse en un sistema de una sola computadora?

El modelo cliente-servidor es eso, un modelo que puede ser adaptado a diferentes plataformas, entre ellas a los sistemas distribuidos, aunque no sea propio de ellos. El concepto básico es: un proceso usuario (cliente) envía la solicitud a un proceso servidor, que realiza el trabajo y devuelve la respuesta. En resumen, sí puede ser adaptado a una sola computadora, solamente que no es lógica esta instalación en una computadora que ya hace el servicio de manera natural.

Tema 2

Ejercicio 1

En la figura 2-2, página 90, se muestran tres estados de procesos. En teoría, si existen tres estados, podría haber seis transiciones, dos por cada estado. No obstante, sólo se muestran cuatro transiciones. En la siguiente figura haremos unos cambios ¿Existen circunstancias en las que podría darse alguna de las transiciones faltantes, o ambas?



Es posible que se diera la situación apuntada arriba. Sin embargo, el diseño del sistema operativo sería más complicado. Imagine que tengamos simultáneamente las transiciones 3 y 6 (obviando la transición 4). Sería sumamente complicado para el sistema operativo determinar a cuál proceso le da prioridad, al que viene de desbloquearse o al que viene del estado “listo”.

Si hubiera una transición 5, un proceso podría quedarse “dormido” entre los procesos “listo” y “bloqueado”. Las transiciones 5 y 6 se han eliminado para evitar mayores problemas a los que de por sí, ya debe atender el sistema operativo.

Ejercicio 3

En todas las computadoras actuales, al menos una parte de los manejadores de interrupciones se escribe en lenguaje ensamblador. ¿Por qué?

Porque se requiere de un lenguaje de bajo nivel (más cerca del sistema operativo) que haya sido diseñado especialmente para este propósito (comunicarse directamente con el sistema operativo).

**Ejercicio 14**

¿Cuál es la mayor ventaja de implementar hilos en espacio de usuario? ¿Cuál es la mayor desventaja?

La ventaja está en que el kernel tiene mayor disposición para ejecutar sus procesos en su propio espacio, en otras palabras, el kernel está utilizando un espacio “prestado” así puede atender cantidad de procesos. La desventaja estriba en que el usuario, en un momento dado podría echar mano a ese espacio y bloquear algún proceso al requerir recursos.

Ejercicio 22

Quando se está desarrollando una computadora, lo común es simularla con un programa que ejecuta una instrucción a la vez. Incluso los multiprocesadores se simulan en forma estrictamente secuencial. ¿Es posible que se presente una condición de competencia cuando no hay sucesos simultáneos, como en la simulación?

Sí, y para eso se crean estos ambientes de simulación, para garantizar que la instrucción anterior, una vez finalizada, haya liberado el o los recursos que estaba utilizando. Si no lo hizo, la siguiente instrucción trataría de acceder un recurso no liberado y así, se daría la condición de competencia.

Ejercicio 27

Si un sistema sólo tiene dos procesos. ¿Tiene sentido usar una barrera para sincronizarlos? ¿Por qué sí o por qué no?

No tiene sentido, pues hay métodos más sencillos, diseñados especialmente para dos procesos, para administrar las competencias. Las barreras fueron diseñadas para grupos de procesos.



Tema 3

Ejercicio 2

Un sistema de intercambio elimina huecos mediante la compactación. Suponiendo una distribución aleatoria de muchos huecos y muchos segmentos de datos y un tiempo de lectura o escritura en una palabra de memoria de 32 bits de nseg, ¿aproximadamente cuánto tiempo se requiere para compactar 128 MB? Para simplificar, suponga que la palabra 0 es parte de un hueco y que la palabra más alta en la memoria contiene datos válidos.

*Primero, convertimos los 128 MB a bits. $128 * 1024 = 131072 \text{ bytes} * 8 = 1.048.576 \text{ bits}$*

1 MB es igual a 1024 bytes y un byte = a 8 bits.

Segundo, leer o grabar 32 bits tarda 10 nseg.

Tercero, obtenemos grupos de bits $1.048.576/32 = 32738 \text{ Grupos de 32 bits}$

Para leer un grupo y grabarlo se tardean 20 nsegs

Cuarto, Obtenemos el tiempo aproximado de leer y grabar 32738 grupos de 32 bits.

*$32.738 * 20 = 655360 \text{ nsegs}$*

Quinto, 1 nseg (nanosegundo) es la millonésima parte de un segundo. (10 a la -9)

Para tener una noción más exacta del tiempo de compactación dividimos el total de nanosegundos entre 1 millón.

$655360/1000000 = 0,66 \text{ (redondeado)}$

En otras palabras para compactar 128 MB se tarda alrededor de menos de 1 segundo.

Ejercicio 4

Considere un sistema de intercambio en el que la memoria contiene los siguientes huecos en orden según su posición en la memoria: 10 KB, 4 KB, 20 KB, 18 KB, 7 KB, 9 KB, 12 KB y 15 KB. Determine cuál hueco se usará si se reciben solicitudes sucesivas pidiendo:

a) 12 KB



b) 10 KB

c) 9 KB

¿Si se usa primer ajuste? Repita el problema usando mejor ajuste, peor ajuste y siguiente ajuste.

Primer ajuste: El hueco de 20 KB, el hueco de 10 KB y el hueco de 18 KB.

Mejor ajuste: El hueco de 12 KB, el hueco de 10 KB y el hueco de 9 KB.

Peor ajuste: El hueco de 20 KB, el hueco de 18 KB y el hueco de 15 KB.

Siguiente ajuste: El hueco de 20 KB, el hueco de 18 KB y el hueco de 9 KB.

Ejercicio 23

Considere la sucesión de páginas de la figura 3-15b. Suponga que los bits R para las páginas B a A son 11011011, respectivamente. ¿Qué página desalojará el algoritmo de segunda oportunidad?

B C D E F G H A

1 1 0 1 1 0 1 1

El algoritmo desalojará la página D, puesto que es la página que va por su segunda oportunidad, de desalojar otra página seguiría la página G. Debe recordarse que las páginas marcadas en el bit R con un 0 significan que ya han sido tomadas en cuenta y que son candidatas a ser desalojadas.

Ejercicio 32

¿Puede una página estar en dos conjuntos de trabajo al mismo tiempo? Explique.

Es posible siempre y cuando en ambas esté de sólo lectura. Puede ser que en una esté de solo lectura y en la otra de lectura y escritura (y borrado) pero no es recomendable desde el punto de vista de confiabilidad de los datos, pues en un momento dado se puede brindar información que está siendo actualizada simultáneamente, lo cual puede provocar errores en la integridad de la información.



Tema 4

Ejercicio 2

En la figura 4-4, uno de los atributos es la longitud de registro. ¿Por qué podría interesarle este dato al sistema operativo?

Recordemos que el sistema operativo también administra la memoria, o sea, es el encargado de asignar segmentos de memoria a los procesos. Si un atributo es la longitud del registro, el SO no tendrá primero que leer el registro para saber su longitud y luego buscar un segmento donde quepan varios registros sino que, con sólo saber este atributo puede realizar una operación, según el número de registros que desee subir a memoria, y buscar el espacio en memoria adecuado. Esto le ahorraría tiempo al SO.

Ejercicio 3

¿Es absolutamente indispensable la llamada al sistema OPEN en UNIX? ¿cuáles serían las consecuencias de no tenerla?

No es indispensable la llamada al sistema OPEN, pues el sistema operativo, al ver que se utilizará un archivo en ese momento, puede realizar las tareas que hace OPEN. Sin embargo, las consecuencias que se pueden dar son: si un programa en ejecución quiere abrir un archivo de datos y no hay espacio en la memoria, el programa puede quedarse por mucho rato esperando espacio. Otra es que el archivo esté siendo utilizado por otra aplicación y no pueda ejecutarse hasta tanto la aplicación que lo tiene ocupado lo libere. Con la llamada OPEN, el programa se garantiza que tendrá el archivo a su disposición. (Página 387 del libro de texto.)

Ejercicio 4

Los sistemas que soportan archivos secuenciales siempre tienen una operación para “rebobinar” archivos. ¿Los sistemas que manejan archivos de acceso aleatorio también la necesitan?

No. La instrucción “rebobinar” se aplica especialmente a archivos que se almacenan en cinta. En cambio los archivos aleatorios se almacenan en disco generalmente.



Ejercicio 5

Algunos sistemas operativos proporcionan una llamada al sistema *rename* para asignar un nuevo nombre a un archivo. ¿Hay alguna diferencia entre usar esta llamada al sistema para cambiar el nombre de un archivo y simplemente copiar el archivo en un archivo nuevo con el nuevo nombre y después borrar el archivo viejo?

Con ambos procedimientos se consigue el fin perseguido: cambiar el nombre del archivo. Sin embargo, habrá que tomar en cuenta que en un momento dado tendremos dos archivos similares, del mismo tamaño y con la misma información. Esto ocupará espacio en disco que puede necesitarse en un momento dado.

Ejercicio 14

¿Cómo implementa MS-DOS el acceso aleatorio a archivos?

Utiliza lo que se denomina la FAT (File Allocation Table). El MS-DOS lleva el control de los bloques de archivos con una tabla de asignación de archivos (FAT) en la memoria principal. La entrada de directorio contiene el número del primer bloque del archivo. Este número se utiliza como índice para consultar la FAT. (Páginas 403 y 440 del libro de texto.)

Tema 5

Ejercicio 1

Los avances en la tecnología de chips han hecho posible colocar una controladora entera, incluida toda la lógica de acceso al bus, en un chip de bajo costo. ¿Cómo afecta eso el modelo de la figura 1-5?

Lo afecta de manera positiva. Lo anterior debido a que el chip, al tener incorporada la lógica de acceso al bus de datos, evita que la controladora tenga que recurrir al kernel liberando al SO de operaciones tal vez rutinarias y permitiéndole que se dedique a tareas más importantes. Entre menos interrupciones haya, el SO será más eficiente. (Página 274 del libro de texto.)



Ejercicio 3

La figura 5-3b muestra una forma de tener E/S con correspondencia en memoria aún se usan buses separados para la memoria y los dispositivos de E/S, probando primero el bus de memoria y, si este falla, probando el de E/S. Un ingenioso estudiante de ciencias de la computación ha ideado una mejora: probar ambos buses en paralelo, a fin de acelerar el proceso de acceso a dispositivos de E/S. ¿Qué le parece la idea?

Esto sería muy bueno si realmente el bus de memoria tuviera fallas frecuentes. Si así fuera, no sería un SO confiable. Como habría un alto porcentaje de fallas la idea no es tan descabellada, más bien sería una manera de hacer más eficiente el tiempo de respuesta. Como las fallas son mínimas, no vale la pena tener dos buses pues de lo que se trata es de separar tareas, no de duplicar.

Ejercicio 13

¿Por qué los archivos de salida para la impresora se colocan generalmente en spooling en disco antes de imprimirse?

Aunque desde el punto de vista técnico sería más fácil permitir que comúnmente un proceso de usuario abra un archivo especial por caracteres correspondiente a la impresora, vamos a suponer que un proceso lo abre y luego no hace nada durante horas. Ningún otro proceso podría imprimir nada. En vez de eso, lo que se hace es crear un proceso especial, llamado demonio, y un directorio especial, llamado directorio de spool. Para imprimir un archivo, lo primero que hace un proceso es generar el archivo completo que se imprimirá y colocará en el directorio de spool. Corresponde al demonio administrar los archivos y es el único proceso autorizado para usar el archivo. Proteger el archivo especial contra el uso directo de los usuarios elimina el problema de que uno de ellos lo mantenga abierto durante un tiempo innecesariamente largo. (Páginas 298 y 299 del Libro de texto.)

Ejercicio 18

¿Cuáles son las ventajas y desventajas de los discos ópticos, en comparación con los discos magnéticos?

Los discos magnéticos (discos duros) son de mayor capacidad y pueden ser regrabados cuantas veces se requiera. No así los discos ópticos que solamente pueden grabarse solamente una vez y son de capacidad limitada. Los discos ópticos tienen la ventaja de que su información no puede ser borrada, es perdurable, obviamente depende de los cuidados que se le brinden al disco de manera física.



Ejercicio 19

Si una controladora escribe en la memoria los bytes que recibe del disco a la misma velocidad a la que los recibe, sin colocarlos en un búfer interno, ¿podría tener alguna utilidad la intercalación? Explique.

No sería necesaria la intercalación, pues la misma memoria haría las funciones de búfer. Lo único que habría que cuidar es que el número de bytes a copiar no sea mayor que la memoria disponible.

Tema 6

Ejercicio 2

Estudiantes que trabajan en PC individuales en un laboratorio de computación envían los archivos que desean imprimir a un servidor que hace spooling en su disco duro. ¿En qué condiciones podría presentarse un bloqueo irreversible, si el espacio en disco para el spooling de impresión es limitado? ¿Cómo puede evitarse el bloque irreversible?

Se presentaría si el servidor de archivos no es capaz de liberar el espacio de impresión cada vez que termine una tarea de impresión. En cierto momento, el spooler puede llenarse pero, si no se libere el espacio, el sistema operativo podría asumir que no hay espacio para más trabajos y quedarse esperando que se libere, cosa que no va a suceder. Lo anterior puede evitarse justamente mediante un algoritmo que indique al sistema operativo que se ha finalizado un trabajo de impresión y, por ende, ese espacio quedaría disponible para un nuevo trabajo.

Ejercicio 3

En la figura 6.1 los recursos se devuelven en orden inverso de su adquisición. ¿Sería igual de conveniente devolverlos en otro orden?

Lo conveniente, independientemente del orden en que se hayan adquirido los recursos es que conforme el proceso deje de utilizar un recurso lo libere inmediatamente pues si se sigue el orden de adquisición se podría retener un recurso por más tiempo del necesitado y retrasar otros procesos.



Ejercicio 17

Un sistema tiene dos procesos y tres recursos idénticos. Cada proceso necesita un máximo de dos recursos. ¿Es posible caer en un bloqueo irreversible? Explique su respuesta.

Sí es posible caer en un bloqueo irreversible. Veamos el siguiente ejemplo:

El proceso A tiene el recurso 1 y el proceso B tiene el recurso 2. Si el proceso A requiere del recurso 2 para completar su trabajo y el proceso B, simultáneamente, requiere del recurso 1 para ejecutar su trabajo, ambos se quedarían esperando que el otro proceso libere su recurso para poder tener acceso a él. Ambos quedarían en un bloqueo irreversible.

Ejercicio 20

Una computadora tiene seis unidades de cinta y n procesos compiten por ellas. Cada proceso podría necesitar dos unidades. ¿Con qué valores de n el sistema está a salvo de bloqueos irreversibles?

Con n igual a 1 o 2 o 3, siempre y cuando no sean las mismas unidades de cinta.

Ejercicio 27

Una manera de prevenir los interbloqueos consiste en eliminar la condición de contención y esperar. En el texto se propuso que, antes de pedir un recurso nuevo, los procesos deben liberar los que tengan (suponiendo que sea posible) (puntuación). Sin embargo, esto introduce el peligro de que un proceso obtenga el recurso nuevo pero pierda algunos de los que tenía antes porque procesos competidores se quedaron con ellos. Proponga una forma de mejorar el esquema.

Como tutor, mi propuesta particular es que un proceso (ejemplo A) libere sus recursos para que otro proceso (B) que tiene el recurso que él necesita finalice su tarea y lo libere, el proceso A entraría en una lista de prioridades del tipo FIFO y, cuando le toque su turno, los demás procesos deberán liberar sus recursos para que el finalice su labor, los procesos que liberaron recursos ingresarán en la lista de prioridades para que en su turno otros procesos liberen los recursos, y así sucesivamente.



Ejercicio 31

Leer enunciado en el libro de texto, página 465

Un algoritmo sería tener dos semáforos, uno a cada lado del río. El último babuino que quiera pasar el río enciende el semáforo, lo pone en rojo (el de su lado) para indicar que él es el último en pasar. Si llega otro no puede pasar hasta que alguien que venga del otro lado lo ponga en verde. En la otra orilla el semáforo está en rojo, de manera que nadie puede pasar hasta que el último babuino que viene lo ponga en verde. La historia se repite hacia el otro lado. Para que no se acapare la pasada solamente de un lado, se puede poner un indicador que no pueden pasar más de 10 babuinos por turno. En este caso, el número 10 será el encargado de poner en rojo el semáforo de su lado y el de poner en verde el semáforo del otro lado cuando llegue.

Tema 7

Ejercicio 9

Cite tres características que debe reunir un buen indicador biométrico para que pueda utilizarse en la autenticación de inicios de sesión.

- a. Debe tener suficiente variabilidad como para que el sistema pueda distinguir sin error entre muchas personas.*
- b. Debe ser una característica que no varíe mucho con el pasar del tiempo.*
- c. Debe ser una característica física del usuario difícil de calificar.*

(Páginas 603 y 604 del libro de texto.)

Ejercicio 14

Con la proliferación de los cafés Internet, las personas van a exigir que sea posible acudir a uno de ellos en cualquier lugar del mundo y trabajar desde él. Describa una forma de producir documentos firmados desde uno de esos cafés empleando una tarjeta inteligente (suponga que todas las computadoras están equipadas con lectores de tarjetas inteligentes). ¿Es seguro su esquema?



Una tarjeta inteligente contiene un chip que almacena información. En esta se puede almacenar la firma digital. El problema sería que la computadora a la que está conectada la lectora de la tarjeta guarde en memoria la firma digital. Esto podría ser contraproducente desde el punto de vista de seguridad.

Ejercicio 15

¿El ataque por caballo de Troya puede funcionar en un sistema protegido por capacidades?

Sí. Recordemos que el caballo de Troya es un programa al parecer inocente que contiene código para realizar una función inesperada e indeseable y una capacidad otorga al dueño del proceso ciertos derechos sobre cierto objeto.

Si alguien me envía un archivo público, y yo lo tomo y lo sumo a mi lista de objetos, ese objeto ya es de mi propiedad, y en cualquier momento podría aparecer el código indeseado en operación. Muy distinto es si alguien trata de tener acceso a un objeto de mi propiedad sin el permiso requerido, ahí no podría tener acceso a ese objeto. (Páginas 607 y 650 del libro de texto.)

Ejercicio 17

Cuando se elimina un archivo, sus bloques por lo general se devuelven a la lista libre, pero no se borran. ¿Cree que sería recomendable que el SO borre todos los bloques antes de liberarlos? Considere factores tanto de seguridad como de desempeño en su respuesta, y explique el efecto de cada una.

Desde el punto de vista de desempeño no sería óptimo, pues no intervienen en nada, y cuando un nuevo proceso quiera utilizarse simplemente la información nueva borra la anterior. Desde el punto de vista de seguridad, habría problemas si hubiera algún proceso que leyera bloques de memoria en desuso pero que no los borre. El proceso debería ser muy sofisticado, pues no siempre la información se guarda en un solo bloque.

Ejercicio 27

¿Cuál es la diferencia entre un virus y un gusano? ¿Cómo se reproduce cada uno?

Un virus es un programa que puede reproducirse anexando su código a otro programa. Un gusano consiste en dos programas: el autoarranque y el gusano propiamente dicho. El autoarranque se



compila y se ejecuta en el sistema atacado. Una vez que se está ejecutando se conecta con la máquina de procedencia, carga el gusano principal y se ejecuta. La diferencia está en que el virus se añade a otro programa, el gusano se propaga por sí mismo. (Páginas 617 y 635 del libro de texto.)

Tema 8

Ejercicio 2

En la figura 13-1 se muestran dos paradigmas, el algorítmico y el controlado por eventos. Indique qué paradigma puede ser más fácil de usar cada uno de los tipos de programas siguientes:

- a. Un compilador
 - b. Un programa para editar fotografías
 - c. Un programa de nómina
- a. Algorítmico
- b. Controlado por suceso
- c. Algorítmico

Ejercicio 6

Supongamos que se intercambian las capas 3 y 4 de la figura 13-2. ¿Qué implicaciones tendría eso para el diseño del usuario?

Lo que se busca con la capa 3 es tener todo el ambiente preparado, para que, al ejecutarse a capa 4, se tengan subprocesos correctos que se calendaricen en forma normal y se sincronicen empleando un mecanismo estándar. Si se intercambian estos procesos, no se tendrá el ambiente debidamente preparado, y el incremento de fallas en los procesos puede ser mayor. (Páginas 867 y 868 del libro de texto.)



Ejercicio 8

A menudo, los sistemas operativos realizan la asignación de nombres en dos niveles distintos: externos e internos. Indique cuáles son las diferencias hay entre estos nombres en cuanto a:

a. Longitud

El nombre externo le permite al usuario identificar más claramente un archivo, pues el nombre es más significativo. Internamente, el nombre debe ser más corto, para no utilizar mucho espacio en memoria.

b. Unicidad.

Externamente, el archivo es una cadena de caracteres. Internamente, es un código en una tabla de referencia o índice.

c. Jerarquías

Externamente, es importante que el usuario tenga una idea de dónde está ubicado el archivo que busca, o sea, en cuál subdirectorio. Lo anterior le puede ayudar a diferenciar entre dos archivos con el nombre igual o similar. Desde el punto de vista interno, se crea una correspondencia directa entre el nombre externo y una dirección de memoria, donde se encuentra el archivo. Internamente, no se manejan jerarquías de directorios y subdirectorios, sino un índice a una tabla.

Ejercicio 9

Una forma de manejar tablas cuyo tamaño no se conoce de antemano es hacerlas fijas. Pero cuando se llene una hay que remplazarla por una más grande, copiar las entradas anteriores a la nueva tabla y después liberar la tabla anterior. ¿Cuáles son las ventajas y desventajas hacer la tabla dos veces mayor que la original, en comparación con hacerla sólo 1.5 veces más grande?

Las ventajas son que, en una sola llamada al sistema, se puede duplicar el espacio de la tabla, esto siempre y cuando se cuente con suficiente espacio en memoria. Además, con un método así, la tabla siempre será continua, no enlazada.

La desventaja aquí es que se requiere de cierta administración del almacenamiento, y la dirección de la tabla ahora es una variable, no una constante.



Ejercicio 12

La indirección es una forma de hacer un algoritmo más flexible. ¿Acaso tiene desventajas? Y de ser así ¿cuáles son?

La desventaja que podría tener es que se deben crear tablas indizadas para direccionar cada uno de los dispositivos. Por ejemplo, en el caso del teclado, cada letra minúscula y mayúscula, así como los caracteres especiales y teclas de función, están en una tabla que las relaciona con su respectivo código ASCII. En otras palabras, al causar una interrupción por teclado (apretar una tecla), primero hay que consultar una tabla que es fija, o sea, no va directamente al código ASCII.

La desventaja sería espacio para la tabla y tiempo en la consulta de la tabla.

Ejercicio 25

Nombre algunas características de un sistema operativo convencional que no sean necesarios en un sistema embebido que se utilice dentro de un aparato doméstico.

Atención de diferentes usuarios: El aparato doméstico es de programación física, para lo que ya está todo programado.

Procesos de E/S diversos: El aparato doméstico tiene entradas específicas, no se da la posibilidad de hacer combinaciones de entradas. Un aparato doméstico tiene perillas o ingreso por teclado, o funciones específicas con nombres.

Los SO convencionales permiten estarse renovando, y rara vez se requiere de un cambio en el hardware. Si se quiere renovar un aparato doméstico por mejores funciones, debe adquirir uno nuevo y desechar el anterior.



GLOSARIO

Acceso aleatorio. Cuando los registros de un archivo se acceden de manera no secuencial sino de acuerdo a una cadena de caracteres dada. Se utiliza generalmente en las bases de datos.

Acceso secuencial. Cuando los registros de un archivo de datos se acceden de manera secuencial, o sea, uno tras otro.

Argumento de extremo a extremo. Este concepto dice que si el programa de usuario tiene que hacer algo él mismo, es un desperdicio hacerlo también en una capa más baja.

Atributo. Calidades de un archivo, tales como tamaño, si es texto (txt, doc) si es una hoja de cálculo (xls), si son datos (dat), etc. Fecha de creación, Última fecha de modificación.

Bloque. Unidad básica de la división de un disco. El disco divide en bloques en los cuales se almacenan fracciones de un archivo.

Buffer. Son memorias auxiliares que se ubican entre una memoria secundaria y la memoria principal y se utilizan especialmente para la lectura y escritura de datos. Las impresoras utilizan este tipo de memorias.

Buses. Es el medio a través del cual viajan los datos de un dispositivo a otro. También se conocen como buses de datos.

Compactación de memoria. Cuando el intercambio crea varios huecos en la memoria es posible combinarlos todos en uno grande desplazando los procesos lo más hacia abajo que sea posible.



Condiciones de competencia. Cuando dos o más procesos compiten por un mismo recurso.

Condición de exclusión mutua. Cada recurso está asignado a exactamente un proceso o está disponible.

Condición de retención y espera. Los procesos que tienen recursos previamente otorgados pueden solicitar otros recursos.

Condición de no expropiación. Los recursos ya otorgados no pueden arrebatarse al proceso que los tiene; este debe liberarlos en forma explícita.

Condición de espera circular. Debe haber una cadena circular de dos o más procesos, cada uno de los que está esperando un recurso que está en manos del siguiente miembro de la cadena.

Controlador de dispositivo. Software asociado al hardware que viene directamente del fabricante. El sistema operativo lo reconoce para que empiece a operar.

Dispositivos de bloques. Discos, que contienen varios bloques de datos susceptibles de direccionarse en forma independiente

Dispositivos de caracteres. Teclados, impresoras, que generan o aceptan un flujo de caracteres.

DMA. Acceso dinámico a memoria. Facilidad que tiene el computador de acceder a la memoria sin necesidad de solicitar o interrumpir al sistema Operativo.

Desfragmentación. Proceso mediante el cual el sistema operativo elimina los huecos entre archivos y graba los archivos en bloques continuos para optimizar el acceso.

Demonios. Programas que se ejecutan en segundo plano que tienen tareas específicas de monitoreo.

Demonio de paginación. Es un proceso que está inactivo pero que se “despierta” periódicamente para inspeccionar el estado de la memoria. Lo anterior para asegurar una provisión abundante de marcos de página libres.

Espacio de direcciones. Es el conjunto de direcciones que pueden que puede utilizar un proceso para direccionar la memoria.

Estados de un proceso. Son tres, listo, bloqueado y en ejecución. Un proceso puede estar en un solo estado a la vez.



Exclusión mutua. Cuando un proceso se encuentra en su región crítica impide que otro tenga acceso al mismo recurso.

Fallo suave. Ocurre cuando la página referenciada no está en el TLB sino en la memoria.

Fallo duro. Ocurre cuando la misma página no está en memoria ni tampoco en el TLB.

FAT. File Allocation Table. Tabla de localización de archivos utilizada en los sistemas MS-DOS

Fragmentación interna. Se da cuando varias páginas están llenas con código o datos, generalmente la última página no se llena al 100%, como ese sobrante no se puede usar o queda desperdiciado se le llama fragmentación interna.

Hilo. Miniproceso. Comparte el mismo espacio de direcciones de todos los miniprocesos del mismo proceso padre.

Inanición. Se puede entender como una espera pasiva de un recurso.

Intercambio. Consiste en llevar cada proceso completo a memoria, ejecutarlo durante cierto tiempo y después regresarlo al disco.

Kernel. Corazón del SO. También se le conoce como núcleo. El usuario no tiene acceso directo a él. Modo Kernel: Cuando una aplicación o tarea es administrada directamente por el kernel o núcleo del SO.

Llamadas al sistema. Solicitud que hace una aplicación en modo usuario al SO de una tarea que sólo éste puede realizar.

Memoria caché. Memoria virtual que se crea en una parte pequeña de la memoria principal. En ella se mantienen las aplicaciones de uso frecuente.

Memoria virtual. Cada programa tiene su propio espacio de direcciones, el cual se divide en trozos llamados páginas.

Modo usuario. Tareas del sistema operativo que están limitadas al usuario.

Multiprogramación. En los sistemas actuales, es la capacidad que tiene el procesador de realizar varias tareas dando la sensación de paralelismo al usuario.

Shell. Grupo de instrucciones disponibles al usuario para realizar ciertas tareas a manera de comando desde una terminal.



Paradigma algorítmico. Se basa en la idea de que un programa se inicia para realizar alguna función que conoce con antelación o que obtiene de sus parámetros.

Paradigma controlado por sucesos. Aquí el programa realiza algún tipo de preparación inicial, digamos exhibir cierta pantalla, y luego espera a que el sistema operativo le notifique del primer suceso

Paradigma de datos. Se refiere a datos en cinta, archivos, objetos o documentos.

Planificación. Responsabilidad que tiene el sistema operativo en organizar la ejecución de los procesos en estado listo.

Proceso. Programa en ejecución; incluye los valores que tienen el contador de programa, los registros y las variables.

Recurso expropiable. Es aquel se le puede quitar al proceso que lo tiene sin causar daños. Ejemplo de ellos es la memoria principal.

Recurso no expropiable. No puede quitársele a su actual dueño sin hacer que el cómputo falle. Ejemplos de ellos un quemador de cd, una impresora.

Región crítica. Porción de programa mediante la cual se tiene acceso a recursos compartidos.

Relojes. También llamados temporizadores, son indispensables para el funcionamiento de cualquier sistema multiprogramado.

Segmento. Espacios de direcciones completamente independientes que pueden crecer o disminuir dependiendo de las necesidades.

Sistema Operativo. Software que administra los recursos disponibles tanto a nivel macro (mainframe) como micro (PC).

Spooling. Es una forma de manejar dispositivos dedicados en un sistema con multiprogramación.



LISTA DE REFERENCIAS

Pita, M. (2007). *Sincronización entre procesos*. Tomado de
<<http://www.monografias.com/trabajos51/sincro-comunicacion/sincro-comunicacion2.shtml>>

Tanenbaum, A. (2009). *Sistemas Operativos Modernos*. México: Pearson.