Sistemas operativos

Autor: Katherine Roa Banquez

AREANDINA
Fundación Universitaria del Área Andina

MIEMBRO DE LA RED
ILUMNO

Sistemas operativos / Katherine Roa Banquez, / Bogotá D.C.,
Fundación Universitaria del Área Andina. 2017

978-958-5455-72-6

Catalogación en la fuente Fundación Universitaria del Área Andina (Bogotá).

Edición:
Fondo editorial Areandino
Fundación Universitaria del Área Andina
Calle 71 11-14, Bogotá D.C., Colombia
Tel.: (57-1) 7 42 19 64 ext. 1228
E-mail: publicaciones@areandina.edu.co
http://www.areandina.edu.co

Primera edición: noviembre de 2017

Corrección de estilo, diagramación y edición: Dirección Nacional de Operaciones virtuales
Diseño y compilación electrónica: Dirección Nacional de Investigación

Hecho en Colombia
Made in Colombia

Sistemas operativos

Autor: Katherine Roa Banquez

AREANDINA
Fundación Universitaria del Área Andina

MIEMBRO DE LA RED
ILUMNO

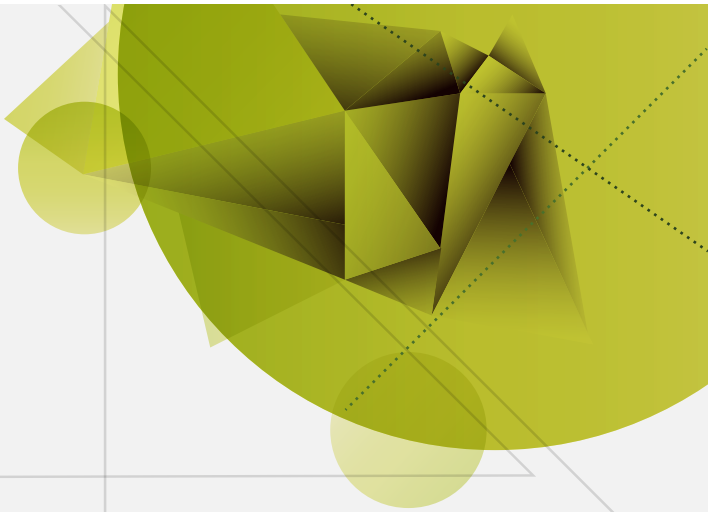
Índice

Índice

1

Unidad 1

Conceptos generales de los sistemas operativos



Sistemas operativos

Autor: Katherine Roa

AREANDINA
Fundación Universitaria del Área Andina

MIEMBRO DE LA RED
ILUMNO

Introducción

El capítulo inicia con la definición de un Sistema Operativo (SO) abordado desde varios autores, se presenta un SO como un intermediario entre un sistema de cómputo y el usuario, cuya tarea es suministrar un buen servicio a estos y alcanzar un uso eficiente de este.

La función principal de un Sistema Operativo es admitir la ejecución de los programas del usuario para asegurar su conveniencia y el uso eficiente de los recursos. En esta guía describiremos las funciones realizadas por el Sistema Operativo para atender las necesidades de un usuario.

Continuaremos conociendo los objetivos y funciones de los sistemas operativos al igual que su evolución a lo largo del tiempo y finalizaremos con la estructura de un Sistema Operativo o mejor conocido como métodos de estructuración.

U1

Metodología

Para un mayor entendimiento de este capítulo, se recomienda al estudiante realizar una lectura de la guía y generar un reporte crítico del mismo a modo personal, con el fin de afianzar los conocimientos adquiridos en esta cartilla.

Se sugiere adicionalmente, desarrollar las diferentes lecturas y actividades planteadas en la semana, con el fin de comprender mejor la temática trabajada.

U1

Desarrollo temático

Conceptos generales de los sistemas operativos

Definición de los sistemas operativos

Si queremos conocer el significado de un Sistema Operativo (SO), tal vez, podamos encontrar diferentes respuestas, teniendo en cuenta a la persona a la que le preguntemos. En este sentido Alegre (2010) expone que:

- Para un joven universitario, un Sistema Operativo es el software que permite el acceso al conocimiento disponible en internet.
- Para un programador, un Sistema Operativo es el software que permite el uso de un sistema de cómputo para el desarrollo de programas.
- Para un técnico de planta informatizada, el Sistema Operativo es el componente de un sistema de cómputo que controla la planta.
- Para una persona del común (que usa un paquete de aplicaciones), un Sistema Operativo es simplemente el software que hace posible que use el paquete.

En nuestro caso nos vamos a enfocar en la definición dada por Candela, S., García, C. Quesada, A. Santana, F. Santos, J. (2007), "… es un programa que actúa como intermediario entre el usuario y el hardware de un sistema de computo. El propósito de un Sistema Operativo es ofrecer un ambiente en el que el usuario pueda ejecutar programas de una forma cómoda y eficiente".

En consecuencia a lo anterior, el hardware de un computador no puede trabajar por sí solo, necesita de unos programas (software) que hacen que el sistema arranque y funcione, a este software que sirve de intermediario entre los usuarios y el hardware es lo que se conoce como Sistema Operativo.

Por lo tanto, un Sistema Operativo es quizás el componente más importante de una computadora, dado que cuando se carga el Sistema Operativo en la memoria y se ejecuta, este despierta el ordenador y hace que reconozca a la CPU, la memoria, las unidades de disco y cualquier dispositivo de entrada y salida, verificando su conectividad y eficiencia en sus procesos.

Un Sistema Operativo controla el uso de los recursos de un sistema de cómputo, tales como la CPU, memoria y los dispositivos de Entrada y Salida (E/S) a fin de satisfacer los requerimientos de los usuarios. Estos esperan conveniencia, calidad de servicio y seguridad al ejecutar sus programas, mientras que los administradores del sistema esperan un uso eficiente de los recursos de la computadora y un buen desempeño al ejecutar los programas de los usuarios.

La interacción del Sistema Operativo con el sistema de cómputo y los programas del usuario es un aspecto crucial de su operación. Las características de la arquitectura del sistema de cómputo se usan para realizar esta.

Un ambiente de cómputo se caracteriza por un sistema de cómputo, sus interfaces con otros sistemas, sus usuarios y la naturaleza de sus requisitos computacionales. Las metas de un Sistema Operativo se determinan por la noción de la utilización efectiva de un sistema de cómputo en el ambiente donde se usa, de modo que los sistemas operativos se clasifican sobre la base de los ambientes de cómputo donde se emplea.

Objetivos y funciones de los sistemas operativos

Se dice que un Sistema Operativo es manejado por eventos. Un evento es cualquier situación que requiere atención del Sistema Operativo. Por ejemplo, una petición del recurso por un programa del usuario o el fin de una operación de E/S. Cuando ocurre un evento, el control del CPU se pasa al Sistema Operativo. Este analiza el acontecimiento y realiza acciones apropiadas. Por ejemplo, cuando un programa pide recurso, el Sistema Operativo lo asigna si está disponible y cuando termina la operación E/S, informa al programa que solicitó la operación de E/S e inicia otra operación de E/S en el dispositivo, si alguno está pendiente. Después de atender el evento, el Sistema Operativo da las instrucciones a un programa de usuario para la ejecución en el CPU.

El funcionamiento manejado por eventos del Sistema Operativo mostrado en la Figura 1, es una visión lógica de su funcionamiento. En la visión física corresponde, el fin de una operación de E/S o una solicitud del recurso pedido por un programa causa una interrupción en el sistema de cómputo. El CPU se diseña para reconocer una interrupción y desviarse a sí mismo a la ejecución de una rutina de interrupción del proceso, que activa una rutina apropiada de manejo de los eventos.

Figura 1. Un Sistema Operativo en su ambiente
Fuente: Propia.

Un Sistema Operativo no solo debe asegurar el uso eficiente de un sistema de cómputo, sino que también debe proveer la conveniencia del usuario. La meta crucial de un Sistema Operativo es proveer la combinación que mejor se ajuste al ambiente de uso eficiente y de conveniencia del usuario, a lo cual se le llama utilización efectiva.

El uso eficiente del sistema es la mayor preocupación en un ambiente de computación que se centra en el sistema, tal como el procesamiento de datos no interactivo. Esto se obtiene a través del uso de buenas políticas de asignación de recursos.

La eficiencia de uso tiene dos aspectos, un Sistema Operativo consume algunos recursos de un sistema de cómputo durante su propia operación; por ejemplo, ocupa memoria y usa CPU. Este consumo de recursos

constituye una sobrecarga que reduce los recursos disponibles a los programas de los usuarios.

El otro aspecto del uso eficiente tiene que ver con el empleo de recursos que realizan los programas del usuario. Puede resultar una eficiencia baja debido a dos causas; si un Sistema Operativo le asigna exceso de recursos a los programas o si es incapaz de asignar recursos libres a los programas que los necesite. Lo primero conduce a un derroche de recursos, y el otro produce entorpecido de los recursos y afecta los progresos de los programas.

Encontramos un gran número de sistemas operativos en uso porque cada uno de ellos provee un gusto diferente de utilización efectiva; por un lado, podemos encontrar sistemas operativos que proveen un servicio rápido requerido por las aplicaciones de comandos y de control, por otro lado, tenemos sistemas operativos que hacen un uso eficiente de los recursos de la computadora para proveer computación de bajo costo.

Un Sistema Operativo implementa requisitos computacionales de sus usuarios con la ayuda de recursos del sistema de cómputo. Sus funciones cruciales son:

Programa: iniciación y terminación de los programas. Proporcionar métodos beneficiosos de modo que varios programas puedan trabajar con el objetivo común.

Recursos: asegurar la disponibilidad de los recursos del sistema y asignarlos a los programas.

Planificación: decidir cuándo y durante cuánto tiempo dedicará la CPU a un programa.

Protección: proteger los datos y los programas contra la interferencia de otros usuarios y de sus programas.

Evolución histórica

Imagen 1. Evolución de los sistemas operativos
Fuente: https://sistemasoperativosuvm.files.wordpress.com/2013/02/linea-del-tiempo.jpg

En la imagen 1. Se observa un resumen de la evolución de los sistemas operativos en cada una de las décadas, iniciando desde la década de los 40 con la creación de la primera generación de computadoras en el año 1940, estas computadoras funcionaban con válvulas o tubos al vacío, se accedía directamente a la consola por medio de una serie de micros interruptores que permitían introducir el programa en la memoria del computador; estas eran utilizadas en el ámbito científico y militar.

En el año 1951, las computadoras de esta generación, usaban transistores para procesar información, estas computadoras ya trabajaban con lenguajes de programación "de alto-nivel"; uno de los primeros lenguajes fueron COBOL y FORTRAN. Los sistemas operativos fueron bastante simples, con conceptos tales como el monitor residente, el proceso por lotes y el almacenamiento temporal.

En 1964, con esta generación se inició el uso de los circuitos integrados, este tipo de computadoras trabajan con terminales remotas, las cuales permiten acceder a la computadora central para realizar operaciones, también podían ser conectadas a Internet. Los sistemas operativos trabajaban técnicas de multiprogramación, multiprocesador, tiempo compartido, entre otros. Los lenguajes utilizados fueron el FORTRAN, ALGOL y COBOL.

En la década de los 70, en el año 1970, se da inicio UNIX, y con el aparecen los sistemas operativos multiusuario - multitarea, Multics (Multiplexed information and computing service), el cual implementó un único nivel de almacenamiento para el acceso a los datos.

En la década de los 80, en el año 1981, nace el Sistemas Operativos MS-DOS (Micro Soft Disk Operating System), creado por Microsoft para IBM PC. Desde esta fecha iniciaron con las diferentes generaciones, desde la versión PC DOS 1.0.

En el año 1984, nace el Sistema Operativo Mac OS cuya características principal era una GUI (Graphic User Interface), multitareas y mouse; a finales de la década, para el año 1987 surge el Sistema Operativo OS/2 de IBM que intentó reemplazar a DOS como Sistema Operativo de las computadoras personales, era un SO multitarea, reconocía múltiples aplicaciones ejecutándose a la vez, sin embargo solo podía mostrar una aplicación a la vez en la pantalla.

En el año 1990, nacen dos grandes Sistemas operativos, SunOS y BeOS, el primero, fue la versión del SO derivado de Unix y BSD desarrollado por Sun Microsystems para sus estaciones de trabajo y servidores, y el segundo, fue desarrollado por Be Incorporated, orientado principalmente a proveer alto rendimiento en aplicaciones multimedia.

En 1992, sale al mercado el Sistema Operativo Solaris de tipo Unix desarrollado por Sun Microsystems y actualmente por Oracle Corporation, en esta misma década en el año 1993 nace Windows NT, el cual pertenece a la familia de sistemas operativos producidos por Microsoft, de aquí en adelante lanzan las diferentes versiones de Windows: Windows 98 (1998), Windows 10 (Beta), Windows Server (2000),Windows XP (2001), Windows Vista (2007), Windows 7 (2009) y Windows 8.x (2012).

En el año 2001, nace el Sistema Operativo MAC OS X, el cual está basado en el entorno operativo Unix, este SO es desarrollado, comercializado y vendido por Apple Inc.

Durante los años 2005 – 2010, emerge OpenSolaris como Sistema Operativo libre, a partir de la versión privativa de Solaris de Sun Microsystems, a partir de esta generación han nacido otras versiones como IllumOS, OpenIndiana.

Métodos de estructuración

A continuación se presenta cinco diseños de métodos de estructuración:

- Sistemas monolíticos
- Sistemas en capas
- Máquinas virtuales
- Microkernels

Sistemas monolíticos

Es una de las estructuras más utilizadas, teniendo en cuenta que no requiere de una estructura, el Sistema Operativo funciona como una recopilación de procedimientos, donde cada una tiene una interfaz definida desde el punto de vista de parámetros y resultados, los cuales pueden hacer uso de otros procedimientos.

Otra característica fundamental del sistema monolítico, es que trabaja una bajo una estructura básica, para explicar este proceso, nos enfatizaremos en lo expuesto por el autor Tanenbaum (2003).

1. Un programa principal que invoca el procedimiento de servicio solicitado.
2. Un conjunto de procedimientos de servicio que ejecutan las llamadas al sistema.
3. Un conjunto de procedimientos utilitarios que apoyan a los procedimientos de servicio.

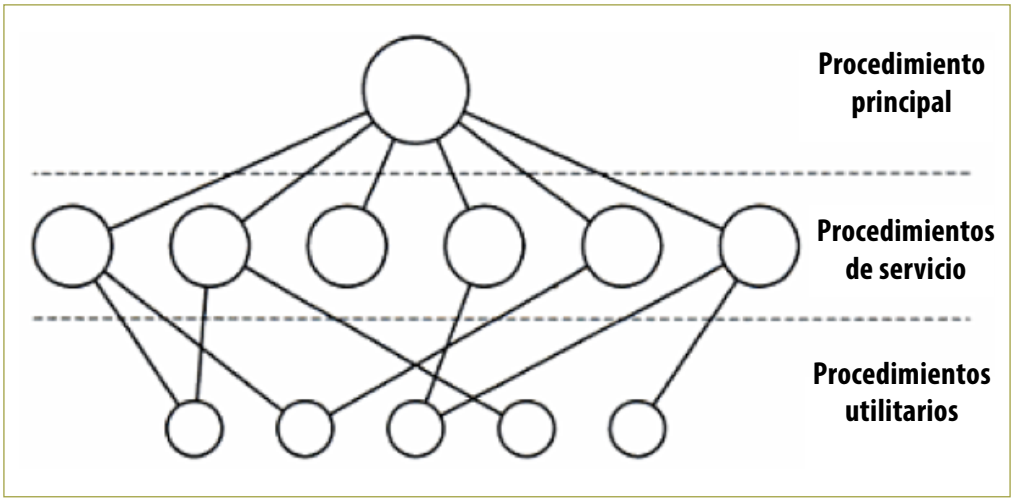


Imagen 2. Modelo de estructuración para un sistema monolítico
Fuente: Tanenbaum (2003).

Sistemas en capas

Este método consiste en estructurar el Sistema Operativo de forma jerárquica entre capas, cada una fijada por la que está abajo.

Como ejemplo de este método, vamos a referenciar el expuesto por el autor Tanenbaum (2003), el cual hace referencia al THE (Techische Hogescool Eindhoven), creado por E. W. Dijkstra y sus estudiantes en el año 1968 en los países bajos. Este sistema trabaja por lotes y cuenta con seis capas:

Capa 0: asignación del procesador y multiprogramación: es el encargado de asignar el procesador, conmutado entre procesos al presentarse interrupciones o expirar temporizadores. Esta capa es la que hace posible la multiprogramación de la CPU.

Capa 1: administración de memoria y tambor: es la encargad de repartir espacios a los procesos en la memoria principal y el tambor.

Capa 2: comunicador operador-proceso: es la encargada de la comunicación entre cada proceso y la consola del operador.

Capa 3: administración de entrada/salida: se encarga de administrar los dispositivos de entrada y salida.

Capa 4: programas de usuario: es la encargada de administrar los procesos, memoria, dispositivos E/S que se ejecutaban desde los programas.

Capa 5: el operador: es el proceso operador de THE.

Máquinas virtuales

Este tipo de método se basa en tiempo compartido que proporciona multiprogramación y una maquina extendida como una interfaz más cómoda. El monitor de la máquina virtual quien es el corazón del sistema, se ejecuta en el hardware y realiza la multiprogramación proporcionando no solamente una máquina virtual sino varias, las cuales son copias exactas del hardware de las maquinas reales.

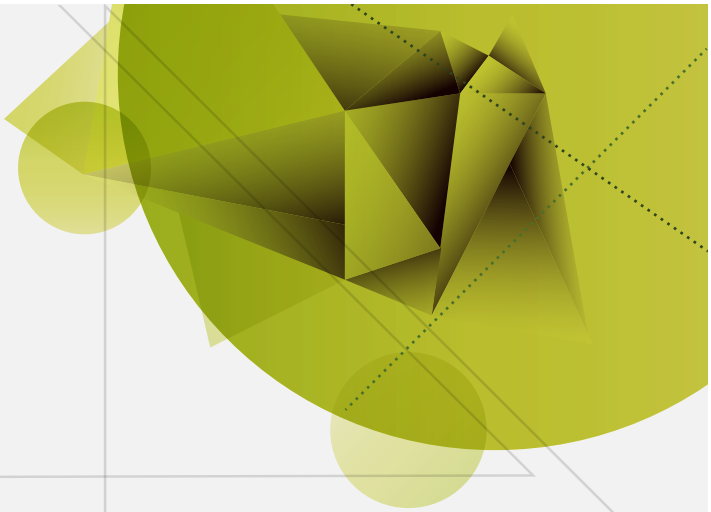
Microkernel

Este método consiste según Tanenbaum (2003) "…en asignar recursos a las máquinas virtuales y luego examinar cualquier intento de usarlos para garantizar que ninguna máquina use los recursos de otra. Cada máquina virtual en nivel de usuario puede ejecutar su propio Sistema Operativo".

1

Unidad 1

Fundamentos de los sistemas operativos



Sistemas operativos

Autor: Katherine Roa

AREANDINA
Fundación Universitaria del Área Andina

MIEMBRO DE LA RED
ILUMNO

Introducción

El Sistema Operativo que hace que el equipo funcione, sin él, el hardware no entraría en funcionamiento y no se podría ejecutar ningún programa. De allí que en este capítulo profundizaremos en el proceso de arranque del sistema desde frio y todos los procesos que realiza este hasta que se pueda ejecutar un programa.

Igualmente, estudiaremos los diferentes componentes de un Sistema Operativo, donde se presentarán las principales características y funcionamiento dentro del SO, y por último abordaremos los tipos de sistemas operativos.

U1

Metodología

Para un mayor entendimiento de este capítulo, se recomienda al estudiante realizar una lectura de la guía y generar un reporte crítico del mismo a modo personal, con el fin de afianzar los conocimientos adquiridos en esta cartilla.

Se sugiere adicionalmente, desarrollar las diferentes lecturas y actividades planteadas en la semana, con el fin de comprender mejor la temática trabajada.

U1 Desarrollo temático

Fundamentos de los sistemas operativos

Arranque del sistema

Cada vez que un Sistema Operativo es arrancado en frio (boot) una parte del SO debe ser trasferida a memoria principal, obteniendo el control del procesador y realizando labores de inicialización del sistema. Se trata de un procedimiento interno del núcleo que en algunos sistemas puede ser invocado por procedimiento de usuario.

La rutina de arranque comienza deshabilitando las interrupciones, pasando a continuación a iniciar el soporte físico; seguido se inicializan las estructuras de datos del sistema y los buzones del sistema (uno por cada interrupción), inicializando a nulo la cola de procesos preparados, retardos y las colas de procesos y mensaje de los buzones del sistema.

El proceso de iniciar el sistema desde el estado de apagado se denomina bootstrapping o booting.

La secuencia de arranque de un ordenador es un proceso que se lleva mediante una serie de pasos:

1. Desde la memoria no volátil o memoria flash se carga el programa BIOS (Basic Input/output System). Este programa comprueba el estado del hardware y determina su configuración, proceso que se denomina POST (Power On Self Test).
2. Si no hay errores (en el paso uno), carga en memoria un código llamado Bootstrap, que puede ser almacenado en un CD-DVD, USB, en un disco duro, en el MBR (Master Boot Record), ubicado en la partición de arranque, o también se puede acceder desde la red.
3. Si arrancamos desde el disco duro, el código en el MBR examina la tabla de particiones, identifica la partición activa, lee el sector de arranque de la partición, ejecuta el código almacenado en ese sector de arranque.
4. Una vez cargado el sistema operativo, este comienza con un test del sistema de archivos, crea las estructuras de datos internas necesarias para el funcionamiento del Sistema Operativo y comienza a arrancar los procesos del sistema.
5. Una vez arrancados los procesos del sistema, es momento en que el equipo ya está en funcionamiento y en espera de que el usuario lo utilice y empiece a ejecutar sus propios procesos

Componentes de un Sistema Operativo

Un Sistema Operativo está compuesto por:

- Gestión de procesos.

- Administración de memoria principal.
- Administración de ficheros.
- Gestión de los dispositivos de entrada/salida.

Gestión de procesos

Un proceso es un programa que ha iniciado su ejecución, cada uno de estos tiene proporcionado un espacio de direcciones en la memoria. Cada proceso está asociado con algún conjunto de registros, incluido el contador de programa, el apuntador de pila y otros registros de hardware, así como la demás información para ejecutar un programa. El ciclo de vida de un proceso nace cuando se inicia su ejecución y muere en el momento en que finaliza su ejecución o es cancelado.

Cuando un programa se ejecuta, este ingresa a un sistema de cola, y el planificador de trabajo del Sistema Operativo va seleccionando de esta el programa a ejecutarse, este tipo de proceso es conocido como procesamiento por cola serie o "por lotes" (batch). En la figura 3 se presenta el procedimiento de un proceso.

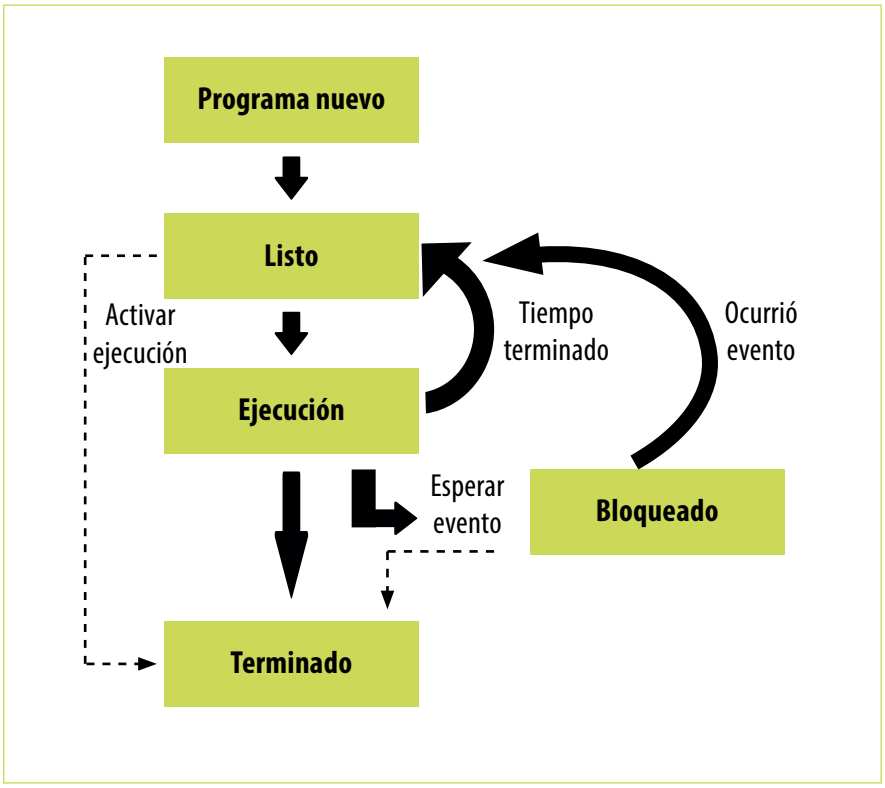


Figura 1. Procesos
Fuente: Propia.

Administración de memoria principal

Un Sistema Operativo consume algunos recursos de un sistema de cómputo durante su propia operación; por ejemplo, ocupa memoria y usa el CPU. Este consumo de recursos constituye una sobrecarga (overhead) que reduce los recursos disponibles a los programas de los usuarios.

> *La **memoria** es como un gran archivador con muchos compartimientos (bytes) a los que se accede mediante una dirección única; estos datos son compartidos por la CPU y los dispositivos de Entrada/Salida.*

Un Sistema Operativo asigna un área de memoria cuando un proceso lo solicita, rastreando las áreas donde se ha liberado memoria por los procesos finalizados. El aspecto clave de la administración de memoria es la fragmentación de memoria, el cual hace referencia a la presencia de áreas libres de memoria inutilizables.

El Sistema Operativo se encarga de administrar la memoria con el fin de identificar qué fragmento de la memoria está siendo utilizada y por quien, disponer qué procesos se asignarán en memoria cuando haya espacio disponible y finalmente, asignar y reclamar espacio de memoria cuando sea necesario.

En la figura 2, se visualiza la memoria que ha sobrado después de haberle asignado a los procesos P1 y P2; estos fragmentos de memoria permanece sin ser utilizada porque es demasiado pequeña para asignar un proceso, pero conjuntamente son bastantes grandes para ser asignadas a un nuevo proceso.

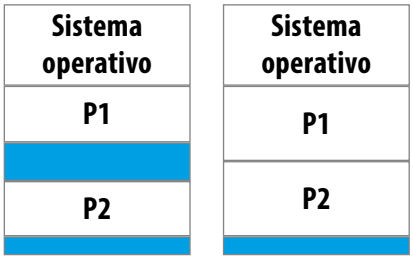


Figura 2. Fragmentación de la memoria
Fuente: Propia.

Administración de ficheros

Los usuarios de la computadora esperan un acceso rápido a los archivos, confiabilidad al encarar las fallas, habilidades para compartir los archivos con compañeros de trabajo y una garantía de que ninguna de las personas no autorizadas puedan usar o manipular indebidamente sus archivos. El sistema de archivo provee una visión lógica al usuario, que consta de una jerarquía del directorio del sistema de archivos, en el cual cada usuario tiene un directorio raíz.

Por lo tanto, cuando un usuario desee remitirse a una información puntual, solo tiene que crear un archivo indicándole el nombre que considere oportuno.

Gestión de los dispositivos de entrada/salida

El Sistema Operativo administra cada uno de los periféricos de un equipo, con el fin de que estos puedan ser compartidos eficientemente por los distintos procesos, informar al usuario las características particulares del hardware que está utilizando.

Asimismo, el Sistema Operativo contiene gestores de periféricos, que son rutinas de E/S encargadas de controlar dispositivos, estos gestores son los únicos que deberán tener en cuenta las peculiaridades concre-

tas de los dispositivos. La rutina de entrada/salida realiza las comprobaciones necesarias sobre la operación que se va a realizar, y hace la petición de servicio al gestor del periférico correspondiente, que será el que efectúe la operación.

Las operaciones de E/S pueden realizarse de tres formas: modo programado de E/S es lento e involucra al CPU en una operación de E/S, por consiguiente, solo puede ejecutarse una operación de E/S. El modo de interrupción es también lento, de modo que se ejecuta una trasferencia de datos byte por byte; de cualquier forma, aquél libera la CPU de las trasferencias de bytes. El modo directo de acceso a la memoria (DMA) puede trasferir un bloque de datos entre la memoria y un dispositivo de E/S sin involucrar al CPU. La interrupción y los medios (DMA) permiten la ejecución simultánea de varias operaciones.

Las operaciones DMA las ejecuta un procesador de propósitos espaciales que están dedicados a la ejecución de las operaciones de E/S.

Clasificación de los sistemas operativos

Sistema de multiprogramación

La concurrencia de operación entre el CPU y el Subsistema de E/s puede explotarse para hacer el sistema haga más trabajo. El Sistema Operativo puede poner a muchos programas del usuario en la memoria y permitir que el CPU ejecute las instrucciones de un programa. A esta técnica se le llama multiprogramación.

La figura 3 visualiza la operación de un Sistema Operativo multiprogramación, donde la memoria contiene tres programas. Una operación de E/S está en curso para el programa 1, mientras que el CPU ejecuta el programa 2. El CPU es conmutado al programa 3 cuando el programa 2 inicia una operación de E/S, y se le conmuta al programa 1 cuando se termina la operación de E/S del programa 1. El kernel de multiprogramación realiza la planificación, la administración de la memoria y la administración de E/S.

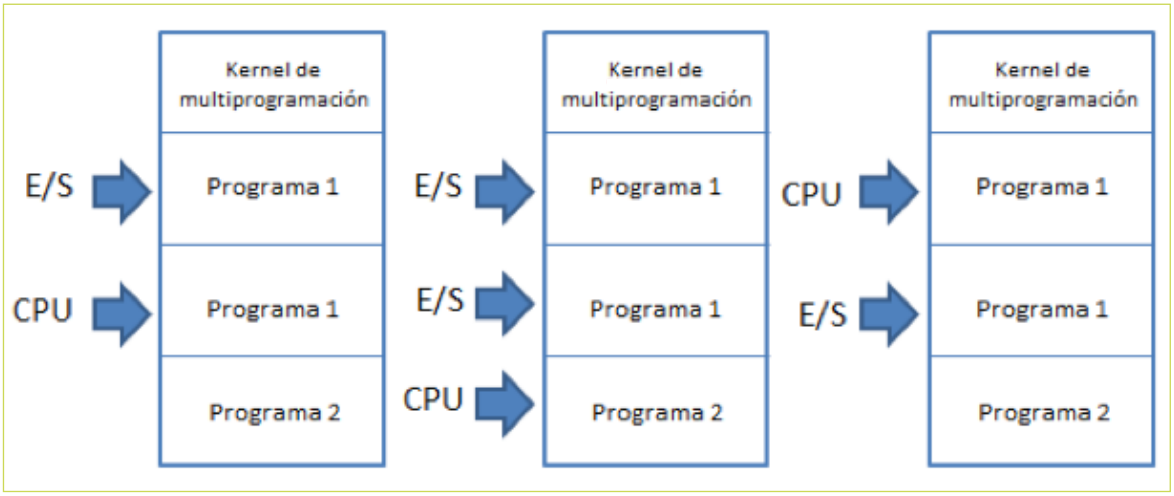


Figura 3. Operación de un sistema multiprogramación
Fuente: Propia (Soporte arquitectónico de un sistema multiprogramación).

DMA: el CPU inicia una operación de E/S cuando se ejecuta una instrucción de E/S. El DMA implementa la trasferencia de datos que interviene en la operación de E/S sin involucrar al CPU. Igualmente, ocasiona una interrupción de E/S cuando termina la trasferencia de datos.

Protección de la memoria: asegura que un programa no tenga acceso ni destruya el contenido de las áreas de la memoria ocupadas por otros programas o por el sistema operativo.

Modo privilegiado del CPU: ciertas instrucciones, llamadas instrucciones privilegiadas, pueden realizarse solamente cuando el CPU esté en el modo privilegiado. Se ocasiona una interrupción de programa cuando el CPU está en el modo de usuario.

En un sistema de multiprogramación, el tiempo de demora es afectado por la cantidad de atención de la CPU que se dedica a otros trabajos que se ejecutan concurrentemente, de modo que el tiempo de demora de un trabajo depende del número de trabajos en el sistema y de las prioridades relativas que el planificador asigna a los diferentes trabajos

Sistemas de tiempo compartido

En un ambiente de cómputo interactivo un usuario no tiene contacto con su programa durante su ejecución. En tal ambiente es natural usar el procesamiento por lotes o el paradigma de la multiprogramación. Los dos sistemas proveen un servicio deficiente al usuario, sin embargo, esto es inevitable debido a la naturaleza de los dispositivos de E/S en uso.

En un ambiente interactivo, un usuario puede proveer entradas a un programa desde el teclado y examinar su salida en la pantalla del monitor. Se usa un paradigma diferente del Sistema Operativo en tales ambientes para proveer un servicio rápido de lo que el usuario solicita, crea la ilusión de que cada usuario tiene un sistema de cómputo para su disposición exclusiva.

El servicio de usuario se caracteriza en términos del tiempo asumido para dar atención a una subpetición, es decir, el tiempo de respuesta. Los beneficios de los buenos tiempos de respuesta se aprecian mejor durante el desarrollo del programa. Una petición típica emitida por el usuario implica la compilación de una declaración o una ejecución de un programa para los datos dados. La respuesta consta de un mensaje del compilador o de resultados computados por un programa. Un usuario emite la siguiente petición después de recibir la respuesta a la petición previa. Los buenos tiempos de respuesta conducirán a una mejora en la productividad del usuario. Cualquier aplicación en el cual un usuario tiene que interactuar con un cómputo derivará beneficios similares de los buenos tiempos de respuesta.

Sistema Operativo de tiempo real

Una aplicación de tiempo real es un programa que responde a las actividades en un sistema externo dentro de un tiempo máximo determinado por el sistema externo.

En una clase de aplicaciones llamadas aplicaciones de tiempo real, los usuarios necesitan que la computadora realice algunas acciones oportunas para controlar las actividades en un sistema externo, o participar de ellas. La puntualidad de acciones se determina por las restricciones de tiempo del sistema externo.

Si la aplicación se toma mucho más tiempo para responder a una actividad, puede ocurrir un fracaso en el sistema externo. Usamos el tiempo requisito de respuesta de un sistema para indicar el valor más grande de tiempo de respuesta para el cual el sistema puede funcionar perfectamente; una respuesta oportuna es aquella cuyo tiempo de respuesta es más pequeño que el requisito de respuesta del sistema.

Algunos ejemplos de este tipo de aplicación en tiempo real, pueden ser las aplicaciones de guía, dirección y control de misiles, control de procesos y control de tráfico aéreo, los sistemas de multimedia y las aplicaciones, como los sistemas bancarios y las reservas que se basan en el uso de base de datos en tiempo real.

Las características de un Sistema Operativo en tiempo real son:

- Permite la creación de procesos múltiple dentro de una aplicación.
- Permite la asignación de prioridades a los procesos.
- Permite al programador definir las interrupciones y las rutinas de procesamientos de las interrupciones.
- Usa planificación accionada por prioridades u orientadas a los plazos límite.
- Suministra tolerancia a las fallas y capacidades de degradación parcial.

Sistemas operativos distribuidos

Un Sistema Operativo distribuido consta de varios sistemas de cómputo individuales conectados a través de una red. Así, muchos recursos de la misma clase, por ejemplo, muchas memorias, CPU y dispositivos de E/S, existen en el sistema distribuido.

Un sistema distribuido saca provecho de la multiplicidad de recursos y de la presencia de una red para proveer las ventajas de los recursos compartidos a través de las computadoras, la fiabilidad de operaciones, la aceleración de aplicaciones y la comunicación entre usuarios.

Sin embargo, la posibilidad de fracasos de la red o del sistema de cómputo individuales complica el funcionamiento del Sistema Operativo y necesita el uso de técnicas especiales en su diseño.

Las principales características de este tipo de sistema son:

- Recursos compartidos: mejora la utilización de los recursos a través de las fronteras de los sistemas de cómputo individuales.
- Fiabilidad: disponibilidad de recursos y servicios a pesar de las fallas.
- Rapidez del equipo: las partes de un cálculo pueden ejecutarse en diferentes sistemas de cómputo para acelerar el cómputo.
- Comunicación: suministra medio de comunicación entre entidades remotas.
- Crecimiento por incrementos: puede magnificarse las capacidades del sistema a un costo proporcional a la naturaleza y tamaño de la magnificación.

Sistemas operativos modernos

Los usuarios se involucran en actividades diversas en un ambiente moderno de computación. Por tanto, un Sistema Operativo moderno no puede usar una estrategia uniforme para todos los procesos; debe usar

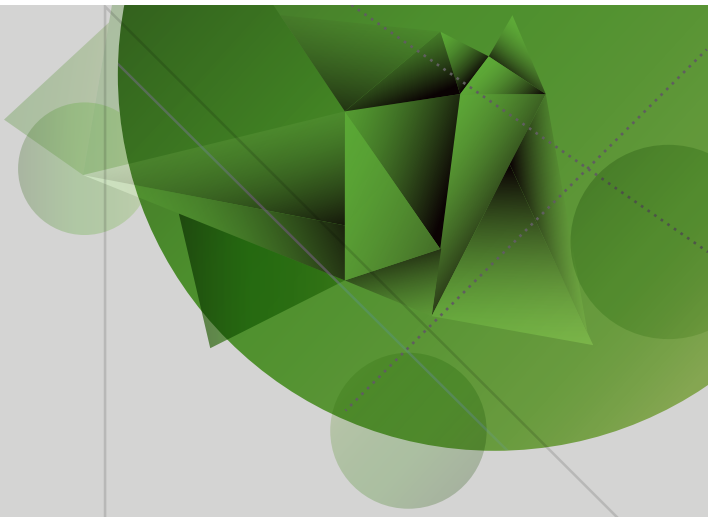
una estrategia que sea apropiada para cada proceso individual. Por ejemplo, un usuario puede abrir un manipulador de correo, editar algunos archivos, ejecutar algunos programas y observar un video al mismo tiempo. Aquí, la ejecución de un programa puede ser interactiva e involucrar una actividad en otro nodo de un sistema de cómputo distribuido, y la vigilancia de un video es una actividad de tiempo real suave, así es que el Sistema Operativo debe usar la planificación round-robin para la ejecución del programa, debe usar planificación basada en prioridades para los procesos de la aplicación de video e implementación de las llamadas de procedimiento remoto para admitir las actividades de otro nodo.

Un Sistema Operativo no puede destinar estrategias diferentes para distintos tipos de procesos, por lo que un Sistema Operativo moderno realmente usa una estrategia individual compleja que adapta su funcionamiento a la satisfacción de cada proceso individual.

2

Unidad 2

Gestión de procesos



Sistemas operativos

Autor: Katherine Roa

AREANDINA
Fundación Universitaria del Área Andina

MIEMBRO DE LA RED
ILUMNO

Introducción

Analizaremos dos puntos de vista de los procesos, el del programador y el del sistema operativo. Desde el punto de vista del programador, estudiaremos como se crean los procesos concurrentes y como interactuar entre sí para alcanzar una meta común. Desde el punto de vista del sistema operativo, analizaremos como crea procesos un sistema operativo, como le sigue la pista a los estados de proceso y de qué manera usa la información del estado de proceso para organizar la ejecución de los programas.

También abordaremos el concepto de un hilo, el cual hace referencia a una ejecución de un programa que usa el ambiente del mismo proceso, es decir, su código, sus datos y sus recursos. De allí, que un sistema operativo aprovecha estas circunstancias para reducir la sobrecarga mientras conmuta entre tales hilos.

U2

Metodología

Para un mayor entendimiento de este capítulo, se recomienda al estudiante realizar una lectura de la guía y generar un reporte crítico del mismo a modo personal, con el fin de afianzar los conocimientos adquiridos en esta cartilla.

Se sugiere adicionalmente, desarrollar las diferentes lecturas y actividades planteadas en la semana, con el fin de comprender mejor la temática trabajada.

U2

Desarrollo temático

Gestión de procesos

Descripción de procesos

Un proceso es una ejecución de un programa. El énfasis en "una" significa que varios procesos pueden representar distintas ejecuciones del mismo programa. Esta situación surge cuando se inician varias ejecuciones de un programa, cada una con sus propios datos y al mismo tiempo está en ejecución un programa que está codificado con técnicas concurrentes de panificación.

Un programador usa los procesos para lograr la ejecución de los programas en una manera secuencial. Un sistema operativo usa procesos para organizar la ejecución de los programas. El concepto de proceso le permite a un sistema operativo ejecutar los programas tanto secuencial como concurrentemente con la misma facilidad.

Un programa es una entidad pasiva que no realiza ninguna acción por sí misma; tiene que ser ejecutado para realizar acciones específicas en él.

Uniendo los dos términos anteriores, procesos y programas, un proceso realiza las acciones especificadas en un programa. Un sistema operativo considera los procesos como entidades para planificarse, de esta manera es como se lleva a cabo la ejecución de programas del usuario.

Existen dos tipos de relaciones entre los procesos y programas:

Uno a uno: cuando se realiza una ejecución individual de un programa secuencial.

Muchos a uno: cuando existen muchas ejecuciones simultáneas de un programa, ejecución de un programa concurrente.

Vamos a tomar como guía el ejemplo que nos expone los autores Candela, S., García, C. Quesada, A. Santana & F. Santos, J. (2007), con el fin de entender un poco más el concepto de proceso,

El programa P mostrado en la figura 1 a) contiene las declaraciones del archivo info y una variable ítem, de la misma manera que las declaraciones que leen los valores de info los usan para realizar algunos cálculos e imprimir un resultado antes de detenerse. Durante la ejecución, las instrucciones de este programa usan valores en su área de datos u la pila a fin de realizar los cálculos pretendidos. b) muestra una vista abstracta de su ejecución. Las instrucciones, los datos y la pila del programa P constituyen su espacio de dirección. Para darse cuenta de la ejecución de P, el sistema operativo asigna la memoria con el fin de acomodar el espacio de la dirección de P, destina una impresora para imprimir sus resultados, establece un acomodamiento a través del cual P puede acceder al archi-

vo info y programa a P para la ejecución. El CPU se muestra como rectángulo con las líneas punteadas porque no siempre ejecuta las instrucciones de P; el sistema operativo comparte el CPU entre la ejecución de P y las ejecuciones de otros programas.

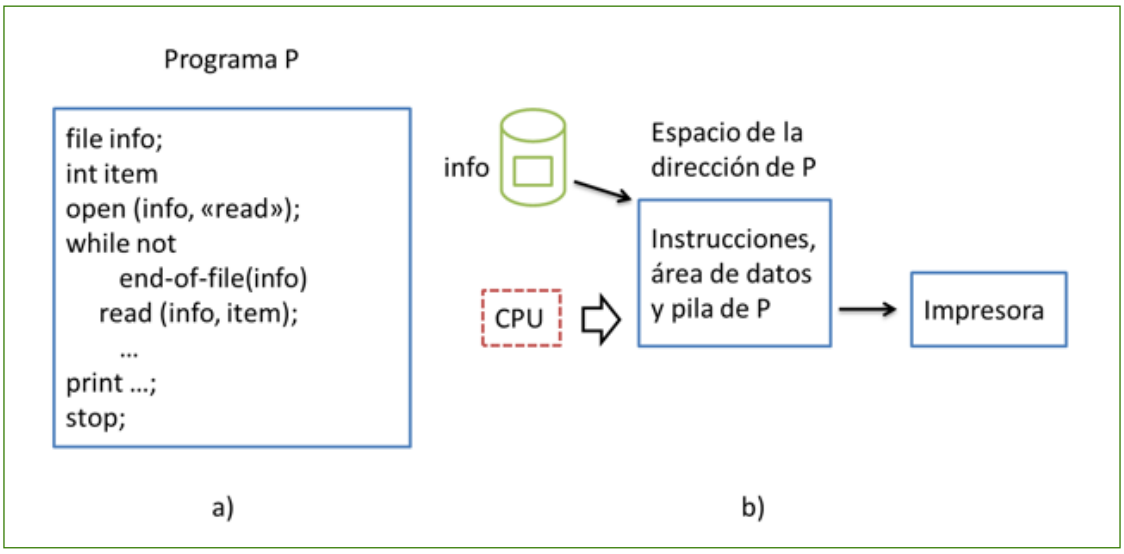


Figura 1. Un programa y una vista abstracta de su ejecución
Fuente: Propia.

Procesos hijo, el sistema operativo inicia la ejecución de un programa creando un proceso para él. Éste es designado el proceso principal para la ejecución, y puede crear otros procesos, que se convierten en sus procesos hijos. Un proceso hijo a su vez puede crear otros procesos. Todos estos forman un árbol que tiene al proceso principal como su raíz.

Las ventajas de los procesos hijos son:

La aceleración del equipo, la creación de procesos múltiples en una aplicación suministra las tareas múltiples. Sus beneficios son similares a los de la multiprogramación; capacita al sistema operativo para interpaginar la ejecución de los procesos acotados por la E/S y por el CPU en una aplicación, suministrando con ello la aceleración de la máquina.

Prioridad de las funciones críticas, a un proceso hijo creado para ejecutar una función crítica en una aplicación se le puede asignar una prioridad más alta que a otras funciones. Estas asignaciones de prioridad pueden ayudar al sistema operativo a cumplir los requerimientos de tiempo real de una aplicación.

Protección del proceso padre contra errores, el sistema operativo cancela un proceso hijo si surge un error durante su ejecución, esta acción no afecta al proceso padre.

Ciclo de vida de un proceso

El estado de proceso es un indicador de la naturaleza de la actividad actual en un proceso.

Un sistema de cómputo convencional contiene un CPU, de modo que cuando mucho

un proceso puede estar en el estado en ejecución, sin embargo, puede existir cualquier número de procesos en los estados bloqueado, listo y terminado. Un sistema operativo puede definir más estados de proceso para simplificar su propio funcionamiento o para admitir funcionalidades adicionales, como hacer un intercambio.

Ejecución: en ese momento el CPU está ejecutando instrucciones en el código del proceso.

Bloqueado: el proceso tiene que esperar hasta que se conceda una petición de recurso formulada por aquel, o hasta que ocurra un evento especifico. No debe asignársele un CPU hasta que su espera esté terminada.

Listo: el proceso desea usar el CPU para continuar su operación; sin embargo, no ha sido planificado.

Terminado: la operación del proceso, es decir, la ejecución del programa representado por aquél, se ha terminado normalmente, o el sistema operativo la ha abordado.

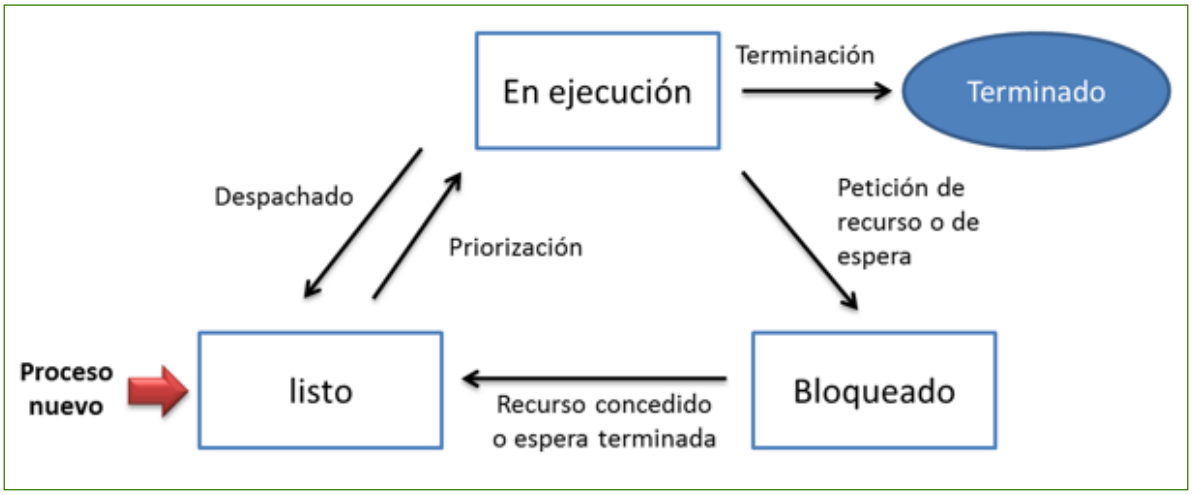


Figura 2. Transiciones de los estados de un proceso
Fuente: Propia.

En la figura 2. Observamos las diferentes transiciones de los estados que un proceso puede tomar. Un proceso puede estar listo, se ejecuta y finaliza su ejecución, puede haber otros casos donde el proceso está listo, se ejecuta, pero este pasa a un modo de espera quedando bloqueado, y hasta que el sistema no le conceda recursos o su tiempo de espera haya finalizado no quedara en el estado listo para finalizar su ejecución y terminación del proceso.

A continuación abordaremos las diferentes causas de las transiciones de estado fundamentales de un proceso:

Listo a en ejecución: el proceso se despacha. El CPU inicia o reanuda la ejecución de sus instrucciones.

Bloqueado a listo: se satisface una petición hecha por el proceso u ocurre un evento que estaba esperándose.

En ejecución a listo: el proceso se prioriza porque el sistema operativo decide planificar otro proceso. Esta transición ocurre porque un proceso de prioridad más alta queda listo o porque expira el intervalo de tiempo del proceso.

En ejecución a bloqueado: el programa que está siendo ejecutado hace una llamada al sistema para indicar que desea esperar hasta que se satisfaga alguna otra petición de recurso hecha por aquél o hasta que ocurra un evento especifico en el sistema. Las principales causas de bloqueo son:

- El proceso solicita una petición de E/S.
- El proceso solicita memoria o cualquier otro recurso.
- El proceso desea esperar durante un intervalo específico de tiempo.
- El proceso espera el mensaje de otro proceso.
- El proceso desea esperar alguna otra acción de otro proceso.

En ejecución a terminado: la ejecución del programa se completa o se termina. Algunas de las causas de terminación son:

- Autoterminación, el programa está siendo ejecutado termina su tarea.
- Terminación por un padre, un proceso hace una llamada de terminar al sistema para terminar el proceso hijo, cuando encuentra que ya no es necesaria o significativa la ejecución del proceso hijo.
- Utilización excesiva de recursos, un sistema operativo puede limitar los recursos que un proceso puede consumir. Un proceso que sobrepase un límite de recursos será terminado por el kernel.
- Condiciones anormales durante la ejecución, el kernel cancela un proceso si surge una condición anormal en el programa que se está ejecutando.
- La interacción incorrecta con otros procesos, el kernel puede cancelar un proceso por la interacción incorrecta con otros procesos.

Control de procesos

Un proceso tiene cinco componentes de proceso: id, código, datos, pila y estado del CPU. El proceso usa el CPU cuando es planificado, también usa otros recursos; estos incluyen los recursos del sistema, como la memoria, y los recursos creados por el usuario, como los archivos. El sistema operativo tiene que mantener la información acerca de todas las características de un proceso.

La visión del sistema operativo de un proceso consta de dos partes:

- El código y las áreas de datos del proceso, incluidos su pila y los recursos asignados.
- La información relativa a la ejecución de un programa.

La figura 3 muestra el arreglo usado para controlar un proceso. Consta de un ambiente de proceso y del bloque de control de proceso (PBC). El id de un proceso se usa para tener acceso a su ambiente de proceso y al PCB. Este arreglo permite que los diferentes módulos del sistema operativo tengan acceso conveniente y eficazmente a los datos relacionado con el proceso.

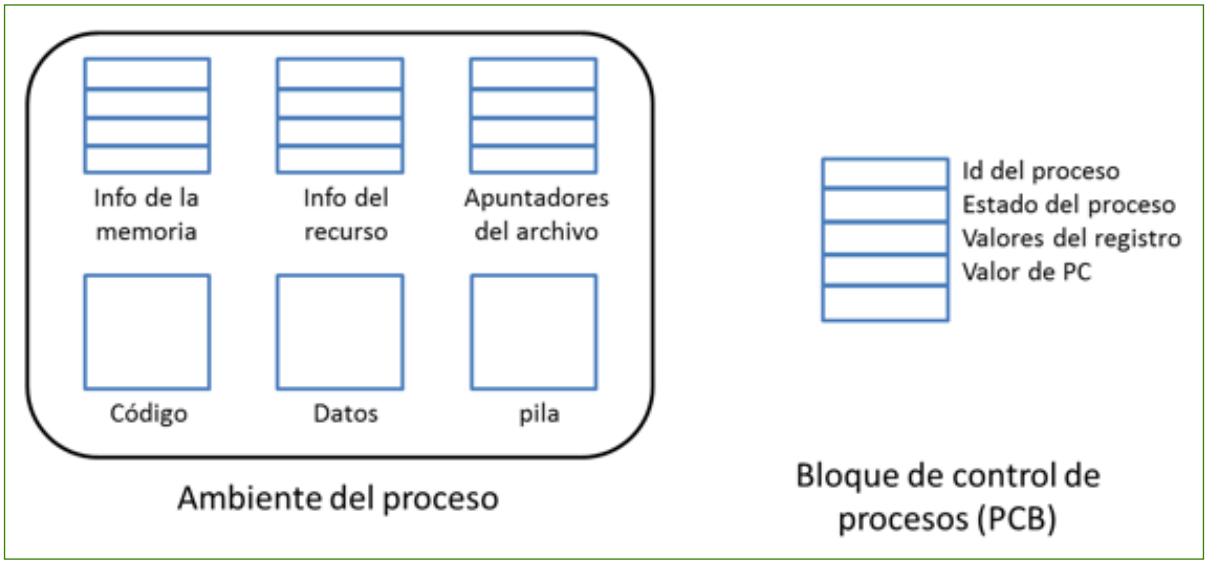


Figura 3. Sistema operativo de un proceso
Fuente: Propia.

El ambiente de proceso contiene el espacio de la dirección de un proceso, es decir, su código, sus datos y la pila, así como toda la información necesaria para acceder y controlar los recursos que se le han asignado. El sistema operativo crea un ambiente de proceso asignándole memoria al proceso, cargando el código de proceso en la memoria asignada y estableciendo su espacio de datos. El sistema operativo también alimenta información relativa al acceso a los recursos asignados al proceso y su interacción con otros procesos y con el sistema operativo.

Los componentes del ambiente de proceso son:

Códigos y datos: código del programa, incluidos sus funciones y sus procedimientos, y sus datos, incluida la pila.

Información de la asignación de memoria: información relativa a las áreas de memoria asignadas al proceso. Esta información se usa para implementar los accesos a la memoria hechos por el proceso.

Estado de las actividades del procesamiento de archivos: los apuntadores a los archivos abiertos por el proceso y las posiciones presentes en los archivos.

Información de los recursos: información relativa a los recursos asignados a un proceso.

Información diversa: información diversa necesaria para la interacción de un proceso con el sistema operativo.

El bloque de control de procesos (PCB) es una estructura de datos del kernel que contiene información referente al id de proceso y el estado del CPU.

El kernel usa tres funciones fundamentales para controlar los procesos:

- La planificación: seleccionar el proceso que debe ejecutarse enseguida en el CPU.

- El despacho: establecer la ejecución del proceso seleccionada en el CPU.
- El contexto de salvar: guardar la información referente a un proceso en curso cuando su ejecución se suspende.

El bloque del proceso contiene toda la información relacionada con un proceso que se usa para controlar su operación, tal como su id, prioridad, estado, PSW y contenido de los registros del CPU, así como la información usada en el acceso a los recursos y a la implementación de la comunicación con otros procesos.

La información prioritaria y de estado es usada por el programador. Pasa el id del proceso seleccionado al despachador. Para un proceso que no está en el estado en ejecución, los campos de los registros del CPU y de la PSW conjuntamente contienen una foto instantánea del CPU del momento en que fue liberado por el proceso por última vez, es decir, tiene el contenido de los diferentes registros de control y de datos del CPU cuando el proceso se bloqueó o cuando fue priorizado.

Ejemplo:

La ejecución del proceso puede reanularse simplemente cargando esta información de su PCB en la CPU. Esta acción será realizada la próxima vez que este proceso se despache.

Llamadas al sistema para la gestión de procesos

La llamada al sistema es la interfase entre los programas y el sistema operativo, este proceso también es conocido como instrucciones ampliadas; este tipo de instrucción funciona a través de comandos que ejecutan un proceso.

Windows, conoceremos dos de los comandos más utilizados en la gestión de un proceso el cual es tasklist y taskkill,

Tasklist: nos permite ver una lista completa de los procesos activos en el sistema, estos se encuentran organizados alfabéticamente. Con este comando podemos visualizar si hay algún código malicioso en la máquina. La información que visualiza el comando es el nombre del proceso y el PID.

Tasklist /? Visualiza los diferentes comandos que se utiliza con el Tasklist, (es la ayuda en el símbolo del sistema).

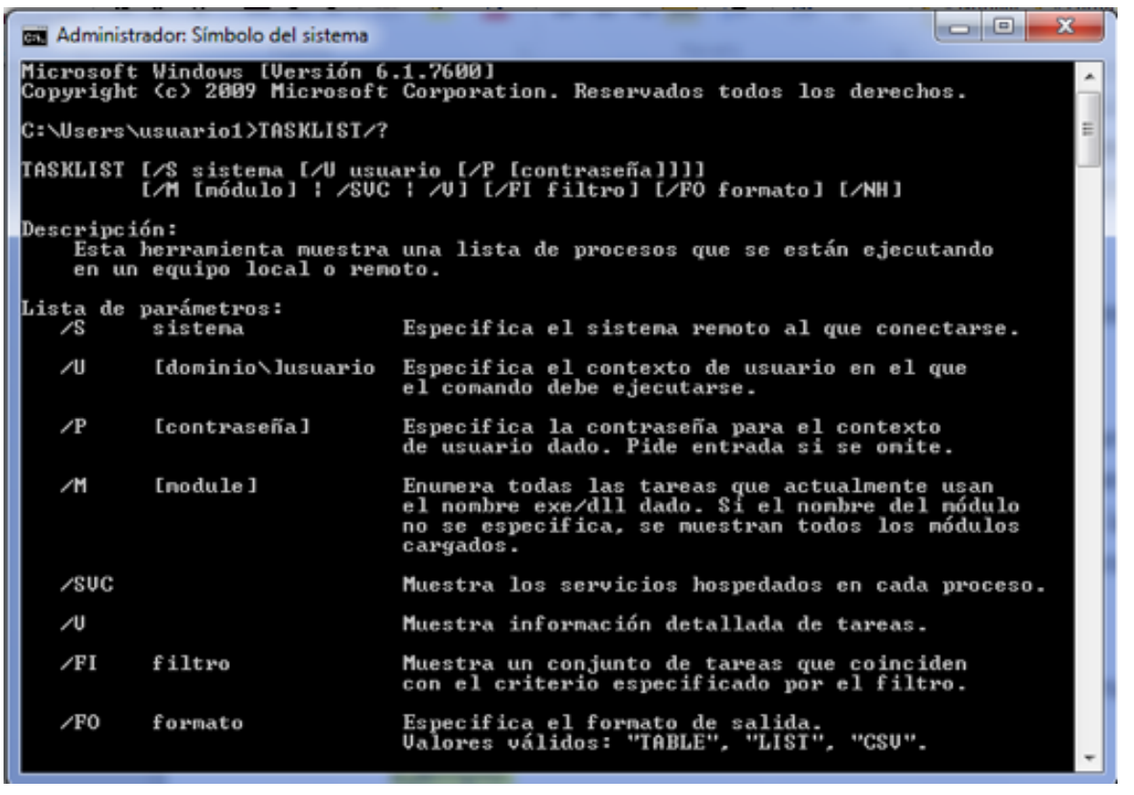


Imagen 1
Fuente: Propia.

Otros comando que hacen uso de Tasklist /…

/M {Modulo} Visualiza todos los procesos que actualmente usan un programa .exe o una .dll. Si el nombre del módulo no se especifica, se muestran todos los módulos cargados.

/SVC. Visualiza las asistencias de cada proceso.

/V. Visualiza al detalle toda la información de los procesos.

/FI filtro. Visualizan los procesos que se especifican en el filtro.

/FO formato. Describe el formato de salida, los valores validos son: TABLE, LIST, CSV.

/NH. Indica que el encabezado de columna no debe aparecer en la salida.

Taskkill: nos permite finalizar uno o más procesos, se debe indicar el PID o el nombre del proceso.

Taskkill /? Visualiza los diferentes comandos que se utiliza con el Taskkill, (es la ayuda en el símbolo del sistema).

Ejemplo:

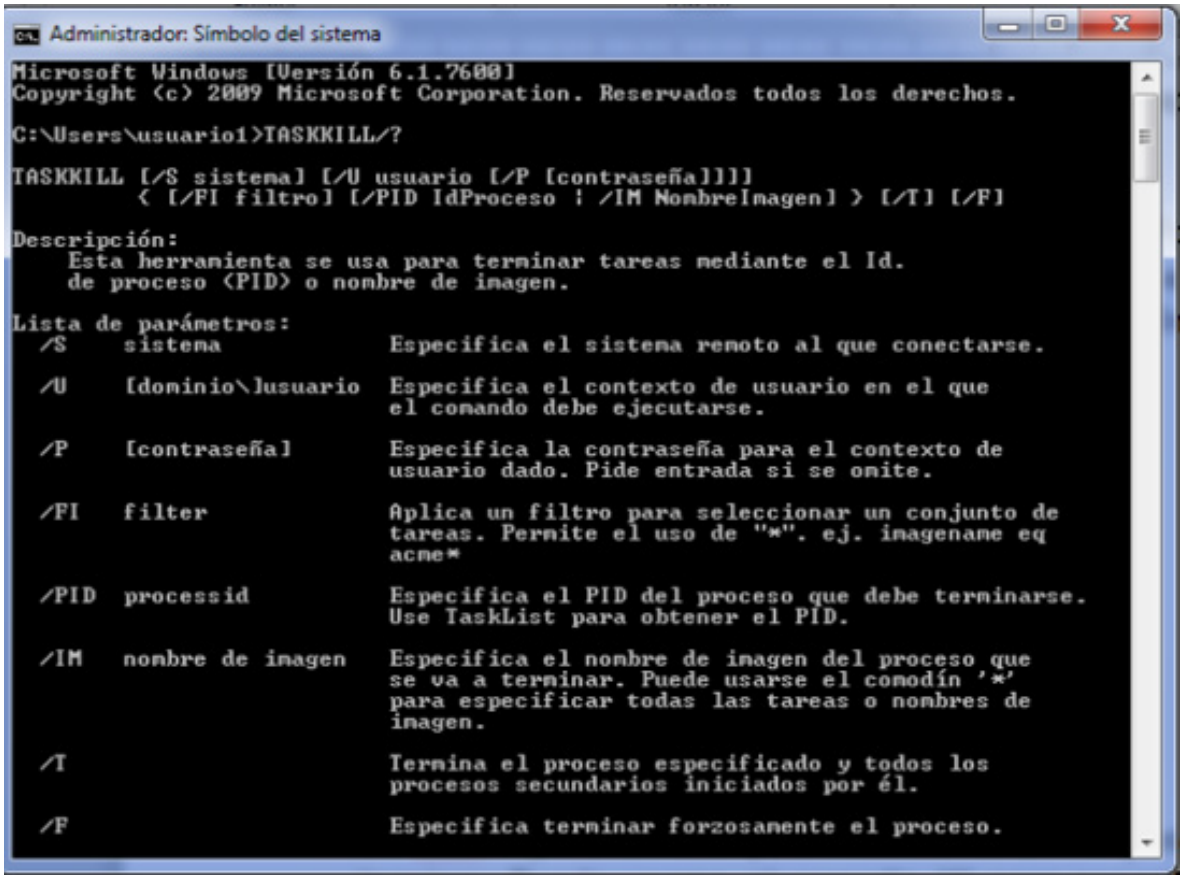


Imagen 2
Fuente: Propia.

Otros comando que hacen uso de **Taskkill /…**

/pid {número de proceso}.

/FI filter. Aplica un filtro para seleccionar un conjunto de proceso que se va a terminar. Puede usarse el comodín "*" para especificar todos los procesos o nombres de imagen.

/T. Finaliza el proceso especificado y todos los procesos secundarios iniciados por este.

/F. Permite terminar forzosamente el proceso.

En el siguiente link podrán visualizar otros comandos que podemos utilizar en la gestión de procesos:

Windows

https://support.microsoft.com/es-co/kb/186592

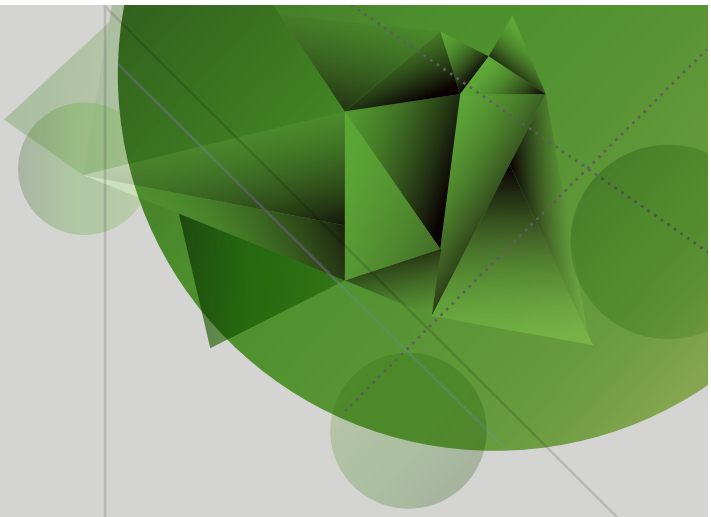
Linux

http://sopa.dis.ulpgc.es/ii-dso/leclinux/miscelaneos/llamadas/LEC_llamadas.pdf

2

Unidad 2

Planificación de procesos



Sistemas operativos

Autor: Katherine Roa

AREANDINA
Fundación Universitaria del Área Andina

MIEMBRO DE LA RED
ILUMNO

Introducción

La política de planificación se utiliza en un Sistema Operativo afecta al servicio del usuario, el uso eficaz de los recursos y al desempeño del sistema. Para lograr los fines que se proponen las políticas de planificación se usan las técnicas fundamentales de priorización, reordenamiento de las solicitudes y variación del intervalo de tiempo.

Un Sistema Operativo debe adaptar su funcionamiento a la disponibilidad de recursos que hay en el sistema; para este efecto, utiliza una combinación de tres programadores denominados de largo, mediano y corto plazos; importantes en la planificación práctica es la imparcialidad en el servicio del usuario. También se estudiará la planificación en un entorno en tiempo real.

El análisis de desempeño de las políticas de planificación es relevante para ajustar el desempeño de una política de planificación y comparar políticas alternativas.

U2

Metodología

Para un mayor entendimiento de este capítulo, se recomienda al estudiante realizar una lectura de la guía y generar un reporte crítico del mismo a modo personal, con el fin de afianzar los conocimientos adquiridos en esta cartilla.

Se sugiere adicionalmente, desarrollar las diferentes lecturas y actividades planteadas en la semana, con el fin de comprender mejor la temática trabajada.

Planificación de procesos

Términos y conceptos de la planificación

Todas las solicitudes en espera de ser atendidas están en una lista de solicitudes pendientes (figura 1). Una solicitud recién llegada se agrega a esta lista. Siempre que se lleve a cabo una planificación, el programador analiza las solicitudes pendientes y elige la que será atendida. Esta se entrega al servidor. Una solicitud sale del servidor cuando está terminado o cuando es priorizada por el programador, en cuyo caso vuelve a ponerse en la lista de solicitudes pendientes. De cualquier modo, el programador realiza la planificación a fin de seleccionar la siguiente solicitud que deberá ser atendida. Así, cuatro eventos relacionados con la programación son la llegada, la planificación, la priorización y la terminación.

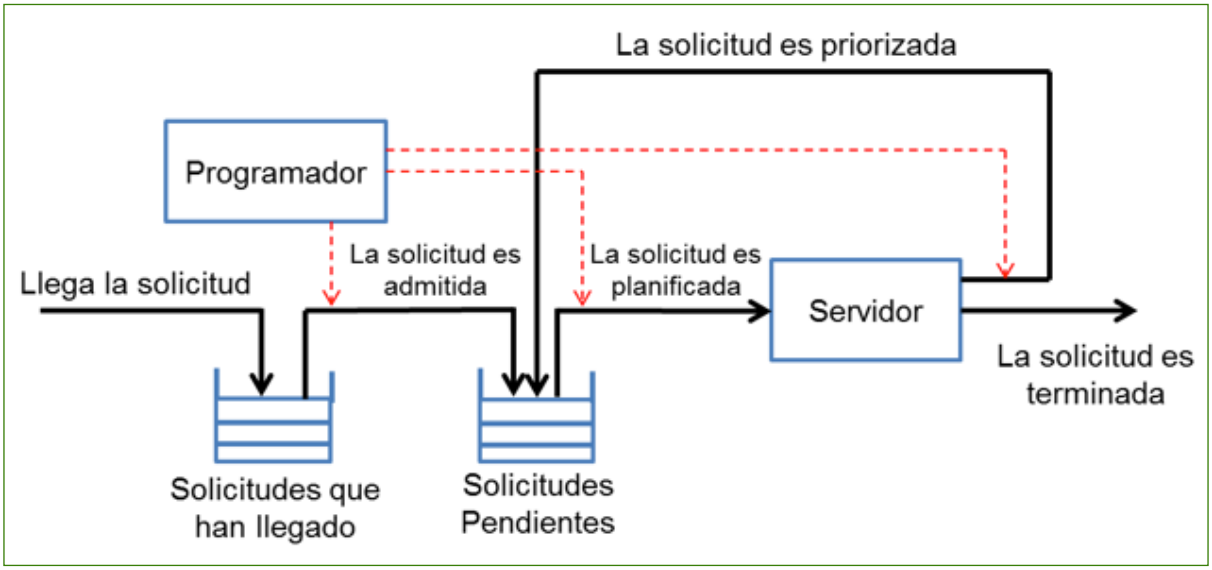


Figura 1. Esquema de la planificación
Fuente: Propia.

Una solicitud es la ejecución de una tarea o de un proceso. Un usuario hace la solicitud y espera su terminación. En un entorno interactivo, un usuario puede interactuar con

un proceso durante su ejecución: el usuario hace una subsolicitud a un proceso y éste responde llevando a cabo una acción o calculan un resultado.

El tiempo de servicio de una tarea o proceso es el tiempo total de CPU y el tiempo E/S requeridos por la tarea o proceso para terminar su operación. Es una propiedad intrínseca de una tarea o proceso. El tiempo total empleado por ellos en el Sistema Operativo puede ser mayor que su tiempo de servicio porque puede hacer momentos en los que el proceso no está siendo ejecutado en el CPU ni está realizando E/S. en consecuencia, su tiempo de terminación depende de su tiempo de llegada, del tiempo de servicio y del tipo de servicio que le proporciona el SO. Los conceptos relacionados con la planificación pueden agruparse en conceptos centrados en el usuario y en conceptos centrados en el sistema.

A continuación abordaremos algunos conceptos relacionados con la planificación,

Solicitud relacionada

- Tiempo de llegada: instante en el que el usuario planea una tarea o un proceso.
- Tiempo de admisión: tiempo en el que el sistema comienza a considerar una tarea/proceso para su planificación.
- Tiempo de terminación: instante en el que se termina una tarea o un proceso.
- Tiempo límite: lapso en el que debe terminarse una tarea o un proceso a fin de satisfacer el requerimiento de respuesta de una aplicación en tiempo real.
- Tiempo de servicio: total de tiempo CPU y tiempo E/S requerido por una tarea, un proceso o una subsolicitud para terminar su operación.
- Priorización: retiro de la asignación forzada del CPU de una tarea o de un proceso.
- Prioridad: la prioridad es una regla de desempate usada para seleccionar una solicitud cuando hay muchas solicitudes en espera de servicio.

Relacionado con el servicio del usuario: solicitud individual

Rebase de tempo límite: cantidad de tiempo en la que el tiempo de terminación de una tarea o un proceso excede su tiempo límite. Los rebases del tiempo límite pueden ser ambos positivos o negativos.

Compartición equitativa: una parte especifica del tiempo de CPU que debe dedicarse a la ejecución de un proceso o de un grupo de procesos.

Relación de respuesta: la relación (tiempo desde la llegada + tiempo de servicio del proceso) / tiempo de servicio del proceso.

Tiempo de respuesta (rt): tiempo que trascurre desde la presentación de una subsolicitud que va a procesar hasta el momento en que su resultado está disponible. Este concepto es aplicable a procesos interactivos.

Tiempo de demora (ta): tiempo que trascurre entre la presentación de una tarea o un proceso y su terminación por el sistema. Este concepto tiene sentido solo para tareas o procesos no interactivos.

Giro ponderado (w): relación que hay entre el tiempo de demora de una tarea o un proyecto y su propio tiempo de servicio

Relación con el servicio del usuario: servicio medio

Tiempo medio de respuesta (rt): promedio de los tiempos de respuesta de todas las subsolicitudes atendidas por el sistema.

Tiempo de demora medio (ta): promedio de los giros de tiempo necesarios de todas las tareas o procesos atendidos por el sistema.

Relacionado con la planificación

Longitud de la planificación: tiempo necesario para terminar un conjunto específico de tareas o procesos.

Rendimiento: número medio de tareas, procesos o subsolicitudes terminadas por el sistema en una unidad de tiempo.

Tipos de planificación

Un Sistema Operativo debe proporcionar una combinación idónea de características centradas en el usuario y en el sistema. También debe adatarse a la naturaleza y cantidad de solicitudes de los usuarios que se espera surja en su entorno, así como a la disponibilidad de recursos. Un solo programador y una sola política de planificación no son capaces de manipular todos sus asuntos.

Por consiguiente, un Sistema Operativo utiliza una disposición que consta de tres programadores denominados programador a largo plazo, programador a mediano plazo y programador a corto plazo encargados de atender diversas cuestiones centradas en el usuario y el sistema.

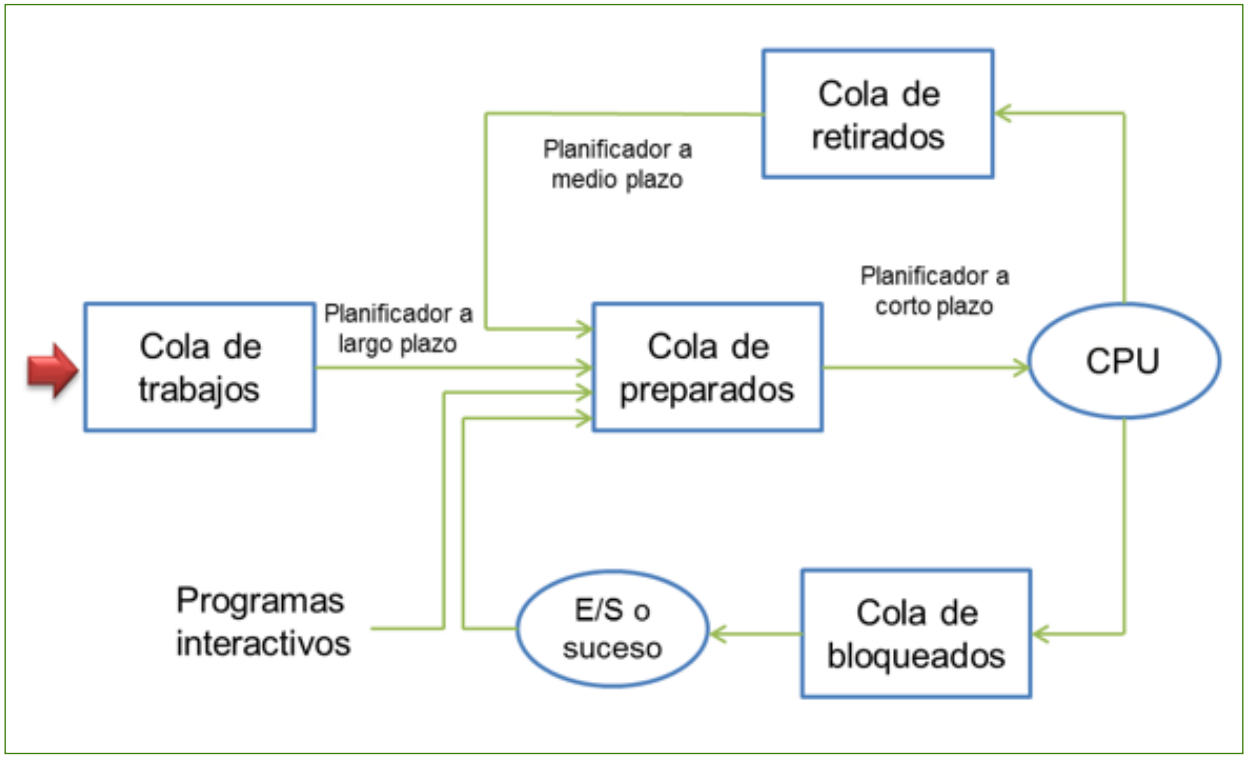


Figura 2. Diagrama de planificaciones
Fuente: Propia.

Planificación a largo plazo

El programador a largo plazo decide cuando empezar a considerar una solicitud para su planificación a mediano y corto plazo. Este hecho se denomina admisión de una solicitud. El programador a largo plazo puede postergar la admisión de una solicitud por dos razones. Puede no estar preparado para asignar recursos suficientes, como estructuras de datos del kernel o dispositivos de E/S., a una petición cuando ésta llega, o puede encontrar que la admisión de una solicitud podría afectar de alguna manera el desempeño del sistema. Por ejemplo, si el sistema contiene actualmente un gran número de solicitudes ligadas al CPU, podría diferir una nueva solicitud ligada a éste, aunque podría diferir una nueva solicitud ligada a éste. Aunque podría admitir de inmediato una nueva ligada a e/s.

La planificación a largo plazo no es relevante si toda petición que llega puede ser admitida de inmediato para su procesamiento, por ejemplo, en un Sistema Operativo de tiempo compartido capaz de crear recursos virtuales cuando es necesario. No obstante, la planificación a largo plazo se utilizó en las décadas de los sesenta y setenta para planificación de tareas porque el equipo de cómputo era costoso y los sistemas de computadoras tenían recursos limitados. Sigue siendo importante en sistemas operativos cuyos recursos son limitados. Su empleo es inevitable si las solicitudes tienen tiempos límite o si un conjunto de solicitudes se repite con una periodicidad.

Planificación a mediano plazo

El programador a mediano plazo trasforma la gran cantidad de solicitudes que puedan caber en la memoria del sistema en un número relativamente menor de solicitudes que pueden caber en la memoria del sistema en cualquier momento. Así, el programador se centra en la suspensión o reactivación de procesos mediante la ejecución de operaciones de suspensión o de activación, de modo que el programador a corto plazo encuentre una cantidad suficiente de procesos listos. El programador a mediano plazo decide cuando suspender un proceso e introduce su PCB en la lista correcta de PCB. Las operaciones de reactivación y suspensión actuales se ejecutan en el administrador de la memoria.

La decisión de suspender un proceso es relativamente fácil de tomar. Puede efectuarse cuando un usuario solicita una suspensión, cuando el kernel se queda sin memoria libre o cuando se encuentra que algún proceso presente en la memoria no tiene probabilidades de ser asignado al CPU durante mucho tiempo. En sistemas de tiempo compartido, los procesos que se encuentran bloqueados o listos son candidatos a ser suspendidos.

La decisión de reactivar un proceso es más complicada. El programador a mediano plazo debe conjeturar cuando un proceso que sigue cronológicamente. El programador puede sustentar su conjetura en la posición que ocupa un proceso en la lista de planificación. Las decisiones de intercambio pueden tomarse en forma periódica o cuando ocurre un evento relacionado.

Planificación a corto plazo

Este tipo de planificación se ocupa del empleo eficiente del CPU. Aquí se elige un proceso de una lista de procesos listos, se decide durante cuánto tiempo debe ejecutarse el proceso y se hace lo necesario para

producir un interruptor de tiempo en la medida en que transcurre el tiempo, seguido, el proceso elegido se entrega al mecanismo de despacho.

En resumen, el programador a largo plazo selecciona los procesos que deben considerarse para su programación por el programador a mediano plazo, que a su vez elige los procesos que deben considerarse para su programación por el programador a corto plazo, y éste selecciona procesos que deben ejecutarse en el CPU. Cada programador aplica sus propios criterios concernientes al uso de recursos y a la calidad del servicio a fin de elegir los procesos para la siguiente etapa de planificación.

Algoritmos de planificación

Los algoritmos de planificación más comunes son:

First Come, First Serve (FCFS)

También conocido como el primero Llegado, el primero servido o FIFO (First In, First On). Es uno de los esquemas más simples de la planificación, apto para multitarea cooperativa.

Atiende al primer proceso que llega y el resto entra en cola de espera. Su principal desventaja es que el tiempo de respuesta es impredecible y penaliza procesos cortos, además de acaparar la CPU hasta que finaliza.

Las solicitudes se programan en el orden en que llegan al sistema. La lista de solicitudes pendientes se organiza como una cola. El programador siempre elige para planificar la primera solicitud de la lista. Un ejemplo de este tipo de planificación es un sistema de procesamiento por lotes en el cual las tareas se ordenan según sus tiempos de llegada y los resultados de una tarea se suministran a un usuario inmediatamente después de su terminación.

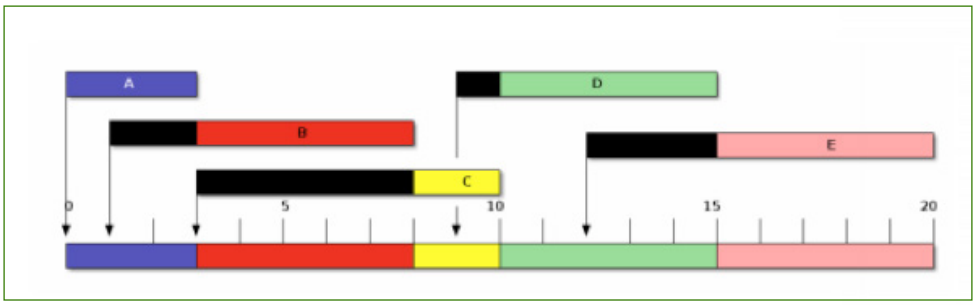


Imagen 1. Planificación FCFS
Fuente: Wolf, G. Planificación de procesos y algoritmos.

Round-Robin

El objetivo de la planificación (RR) con intervalos de tiempo es proporcionar un servicio equitativo a todas las solicitudes. Los intervalos de tiempo se usan para limitar la cantidad de tiempo de CPU que puede usar un proceso cuando es planificado. Una solicitud se prioriza si transcurre el intervalo de tiempo. Esta política preserva los giros ponderados de los procesos aproximadamente iguales al número de procesos activos que hay en el sistema.

Este tipo de planificación puede implementarse organizando la lista de procesos listos como cola. El programador siempre elige el primer proceso de la cola. Si el intervalo de tiempo transcurre durante la ejecución de un proceso, éste se coloca en la fila de la cola. Un proceso que inicia una operación de E/S se retira de la cola de procesos listos, se agrega al final de la lista una vez que termina la operación de E/S. un proceso que se encuentra en la cola de procesos listos avanza paulatinamente al inicio de la cola y es planificado.

La planificación Round Robin se maneja mejor si se mantienen dos listas de BCP. Una debe contener BCP de procesos listos, mientras que la otra debe contener BCP de procesos bloqueados e intercambios suspendidos. La lista de procesos listos debe organizarse como una cola. Debido a que los procesos bloqueados e intercambiados suspendidos se ignoran para efectos de la planificación, no es necesario mantener su lista BCP en algún orden específico. Cuando cambia el estado de algún proceso, el programador cambia la BCP del proceso de una lista BCP a otra.

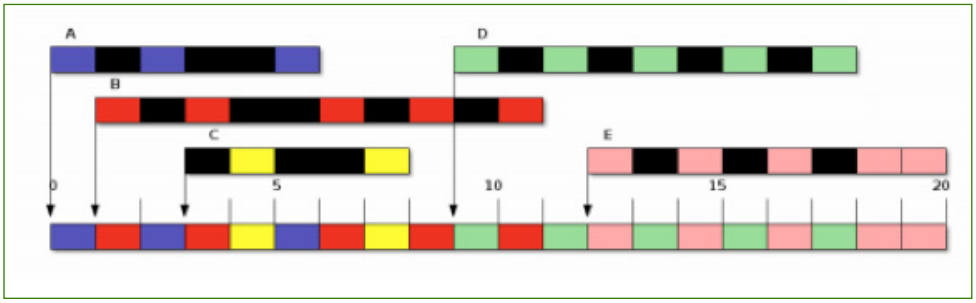


Imagen 2. Planificación Round Robin
Fuente: Wolf, G. Planificación de procesos y algoritmos.

Shortest Process Next (SPN)

También conocido como el proceso más corto a continuación.
El programador SPN siempre planifica la más corta de las solicitudes que han llegado. Así, una solicitud permanece pendiente hasta que se han atendido todas las solicitudes más breves.

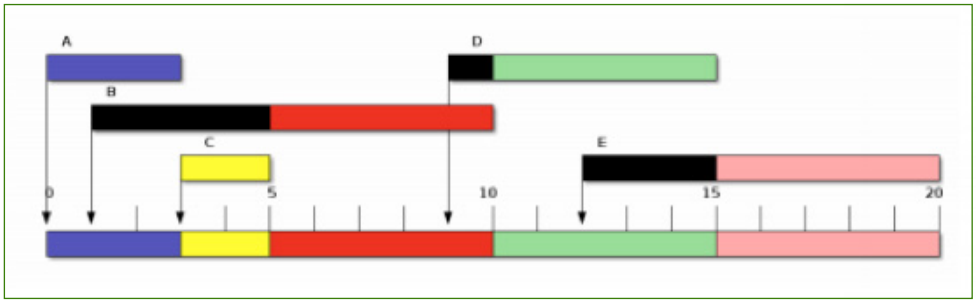


Imagen 3. Planificación SPN
Fuente: Wolf, G. Planificación de procesos y algoritmos.

Prioridad multinivel

Los procesos se asignan por prioridad en diferentes colas. El Sistema Operativo aplicará un determinado algoritmo a cada cola.

La eficacia y un servicio aceptable del usuario proyectan requerimientos contradictorios sobre el programador, de modo que éste tiene que sacrificar tales características entre sí. La multiprogramación y el tiempo compartido representan posiciones extremas en este sacrificio. La multiprogramación proporciona elevados niveles de eficacia, pero no es capaz de garantizar la eficiencia y un buen servicio de usuarios simultáneamente.

Un planificador multinivel mantiene un número de peticiones listas. Los procesos en peticiones diferentes tienen prioridades u tiempos de intervalos de petición diversos. Los procesos con una prioridad alta reciben tiempos de intervalos pequeños, de manera que obtienen tiempos de respuestas buenos. En los procesos con baja prioridad se espera que mantengan el CPU ocupado, de este modo proveen buena eficacia, y reciben grandes tiempos de intervalos.

En la figura 5, en la línea superior al proceso se muestra la cola antes del quantum en que se ejecuta.

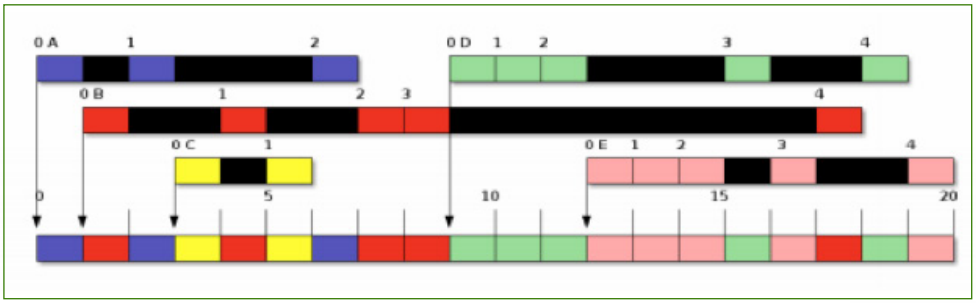


Imagen 5. Planificación multinivel
Fuente: Wolf, G. Planificación de procesos y algoritmos.

Multiprocesamiento

Esta técnica consiste en configurar un sistema de cómputo con varios procesadores. Una de las ventajas de este, es que si un procesador falla, los restantes continúan operándolo cual no es automático y requiere de un diseño y/o asignación de los mismos.

Si un procesador falla, este lo informa a los demás para que alguno de ellos se encargue de las ejecuciones o procesos que llevaba a cargo en ese momento; una vez asignado el nuevo procesador, este ajusta sus estrategias de asignación de recursos para evitar la sobrecarga del sistema que está fallando.

Las metas de los sistemas multiprocesamiento generalmente son la confiabilidad y la disponibilidad de los procesadores, así como el incremento del rendimiento de la máquina.

La planificación de procesos en un sistema multiprocesador tiene dos componentes:

- Planificación temporal: asigna una planificación a cada procesador individual, exactamente igual que si de

un monoprocesador se tratase, salvo por el hecho de que los multiprocesadores es menos relevante para el rendimiento de la maquina; en esta planificación se decide que procesos se van a ejecutar.

- Planificación espacial: define como se asigna los procesadores a los diferentes procesos, en esta planificación se decide en que procesador se ejecutan los procesos.

Políticas de planificación en multiprocesadores

Estas políticas pueden clasificarse en dos grandes grupos no disjuntos, en función de si el conjunto de procesadores se multiplexa entre las aplicaciones en el tiempo

(Políticas de tiempo compartido) o en el espacio (políticas de espacio compartido)

Tomaremos como base lo expuesto por Alegre (2010), para definir cada una de las políticas de planificación en multiprocesadores:

- Políticas de tiempo compartido: una primera opción es mantener los criterios de la planificación de los monoprocesadores y no gestionar espacialmente el uso de los procesadores. En el caso más simple se puede utilizar una cola global única de procesos preparados, que van asignándose a los procesadores libres. Esta política optimiza el reparto de la carga en detrimento de la afinidad al procesador, siendo la alternativa utilizar una cola local para cada procesador, de forma que la asignación al procesador se hace inicialmente y luego el proceso siempre se planifica en el mismo procesador, potenciando la afinidad.
- Políticas de espacio compartido: estas políticas combinan planificación temporal y espacial. Se utilizan en entornos de cálculo intensivo, generalmente con aplicaciones multithread. Pueden clasificarse en dos tipos, que no se excluyen entre sí: políticas de planificación en grupos y políticas de particionado.

Planificación en tiempo real

Las planificaciones en tiempo real imponen algunas restricciones especiales para la planificación, además de la conocida necesidad de satisfacer tiempos límite. En primer lugar, los procesos dentro de una aplicación en tiempo real son interactivos y entre ellos pueden tener prioridades determinadas por la naturaleza de la aplicación; en segundo lugar, los procesos pueden ser periódicos.

Los procesos de una aplicación en tiempo real interactúan entre ellos a fin de asegurar que realizan sus acciones en el orden deseado. Aquí se establecen la hipótesis de simplificación de que esta interacción se lleva a cabo solo al principio o al final de un proceso. Esto provoca dependencia entre los procesos, lo cual debe tomarse en cuenta al determinar los tiempos límite y al llevar a cabo la planificación.

La planificación en tiempo real se centra en la determinación e implementación de una planificación factible para una aplicación, en caso de existir.

Un ejemplo básico de este tipo de planificación es el que nos comparte el autores Candela, S., García, C. Quesada, A. Santana, F. Santos, J. (2007), tabla 1, "…una aplicación para actualizar cada 15 segundos los horarios de salida de vuelos en las pantallas de

información al público, esta aplicación consta de los procesos independientes siguientes, donde el proceso P3 manipula una situación excepcional que ocurre muy rara vez,

Proceso	P1	P2	P3	P4	P5
Tiempo de servicio	3	3	2	4	5

Tabla 1. Ejemplo planificación en tiempo real
Fuente: Propia.

No existe una planificación para terminar en 15 segundos los cinco procesos, por lo que debe ocurrir un rebase del tiempo límite. No obstante, cuando el proceso P5 no está activo son posibles varias planificaciones. El programador en un sistema en tiempo real blando puede encontrar cualquiera de ellas y usarla".

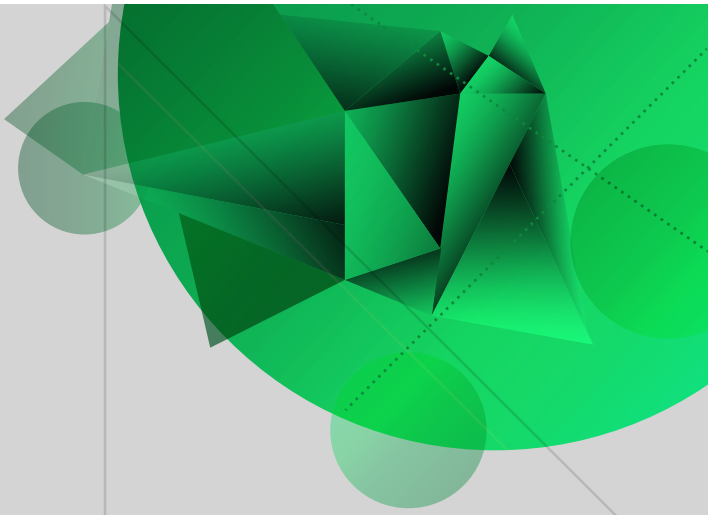
Algunos de los métodos utilizados en la planificación de tiempo real son:

- Planificación estática: antes de iniciar la ejecución, esta realiza una planificación de los procesos, esta se representa en forma de tabla, cuyas filas indican el momento de inicio de la ejecución de los diferentes procesos; durante la operación no se realiza alguna decisión de la planificación.
- Planificación basada en prioridades: la aplicación en tiempo real se analiza para asignar prioridades apropiadas a los procesos que preceden; durante la operación de la aplicación se usa planificación convencional basada en prioridades.
- Planificación dinámica: la planificación se lleva a cabo una vez que se ejecuta una petición para crear el proceso; la creación del proceso tiene éxito solo si es posible garantizar el cumplimiento de los requerimientos de respuesta del proceso.

3

Unidad 3

Gestión de memoria real



Sistemas operativos

Autor: Katherine Roa

AREANDINA
Fundación Universitaria del Área Andina

MIEMBRO DE LA RED
ILUMNO

Introducción

En los capítulos anteriores hemos visto la gestión de procesos, sin embargo, para que un proceso se pueda ejecutar debe estar cargado en la memoria principal, aparte de contar con tiempo de procesamiento; dado que ningún proceso se puede ejecutar sino se le ha asignado algún espacio en la memoria teniendo en cuenta el requerimiento de este. De esta forma, la memoria se convierte en uno de los recursos más importante de los sistemas operativos y la parte encargada de ello se llama gestor de memoria.

En este capítulo se abordará técnicas para obtener un uso eficaz de la memoria, iniciando con una introducción donde se abordan los conceptos y funciones del gestor de memoria, pasando por las jerarquías, direccionamiento e intercambio de memoria y finalizaremos con la presentación de los diferentes mecanismos de gestión de la memoria real.

U3 Metodología

Para un mayor entendimiento de este capítulo, se recomienda al estudiante realizar una lectura de la guía y generar un reporte crítico del mismo a modo personal, con el fin de afianzar los conocimientos adquiridos en esta cartilla.

Se sugiere adicionalmente, desarrollar las diferentes lecturas y actividades planteadas en la semana, con el fin de comprender mejor la temática trabajada.

U3

Desarrollo temático

Gestión de memoria real

Introducción

La memoria es uno de los componentes más importantes de una máquina, la cual gestiona el sistema operativo a través del gestor de memoria, la cual tiene como función primordial la de asignar memoria principal a los procesos que lo requieran.

Otras de las funciones de gestor de memoria son:

- Verificar las zonas de memorias que están o no ocupadas.
- Impedir que un proceso acceda a una zona de memoria que ha sido asignada a otro proceso.
- Asignar memoria a los procesos que la requieran, así como de retirarla cuando ya no la necesiten.
- Administrar el intercambio entre la memoria principal y secundaria, para aquellos casos cuando la memoria principal este completamente asignada.

De esta última función, la gestión de memoria tendrá que:

Reubicar: para un sistema multitarea, el gestor de memoria deberá ubicar los procesos que crea conveniente en la memoria secundaria con el fin de descargar la memoria principal.

Proteger: el gestor de memoria se encarga de impedir que los procesos ocupen zonas de la memoria donde no estén autorizados, para ello comprueba que las referencias a la memoria generadas por un proceso sean las mismas que le asignó a dicho proceso.

Controlar la memoria: el gestor de memoria debe administrar correctamente que parte de la memoria puede asignar e identificar las zonas que se encuentran ocupadas por los procesos.

Intervenir en los casos de fragmentación de la memoria: puede ocurrir dos tipos de fragmentación interna y externa, la primera hace referencia cuando un proceso es más pequeño que la partición asignada, y la segunda, sucede cuando la memoria externa es dividida aún más en cada partición y en estas quedan pequeños bloques disponibles de memoria los cuales son difíciles de reutilizar.

Jerarquías de memoria

La jerarquía de la memoria es un arreglo de varias unidades de memoria con velocidades y tamaños variables que crea una ilusión de una memoria rápida y grande a bajo costo. El CPU recurre a la memoria más rápida, el caché, traslada del siguiente nivel inferior

en la jerarquía de la memoria, que puede ser un proceso, si estos no están disponibles en el caché, se trae del siguiente nivel inferior en la jerarquía de la memoria, que puede ser un caché más lento o la memoria de acceso aleatorio (RAM).

Si la instrucción o los datos requeridos no están disponibles en el siguiente nivel inferior de memoria, se lleva ahí desde un nivel aún más bajo, y así sucesivamente. El desempeño de la jerarquía de la memoria depende de la relación de impactos en varios niveles de la jerarquía, donde este se encuentre en un nivel indica qué fracción de los bytes de las instrucciones o de los datos que se buscaban en ese nivel en realidad estaban ahí.

En la figura 1 se visualiza la jerarquía de la memoria y se describe su operación. Está integrada por la memoria caché, como los cachés L1 y L2, la unidad de administración de la memoria, la memoria y un disco. Los cachés L1 y L2 se manejan en el hardware, de modo que el kernel y los programas del usuario deben emplear técnicas especiales para obtener altas relaciones de impactos de caché.

Figura 1. Administración de la jerarquía de la memoria
Fuente: Propia.

La principal característica del uso de la jerarquía es que permite alojar más procesos del usuario en la memoria para mejorar tanto el desempeño del sistema como el servicio del usuario. El kernel satisface esta preocupación mediante la reutilización eficaz de la memoria cuando un proceso finaliza, lo cual reduce la cantidad de memoria no utilizada en el sistema en cualquier momento. Este proceso se da gracias a la biblioteca en tiempo de ejecución (run-time) del lenguaje de programación en que está escrito el proceso.

Direccionamiento de memoria

El direccionamiento de memoria hace referencia al proceso de intentar acceder a la memoria con el fin de realizar un cambio en esta, ya sea para leerla, escribir o borrar contenido que se encuentre en ella por medio de una dirección.

El direccionamiento de la memoria, se especifica desde el diseño del hardware de una máquina, dado que en esta se define como se va a direccionar la memoria; esta se da según la cantidad de bits que contengan las direcciones que se van a generar.

El máximo direccionamiento posible se puede obtener tomando el número de bits como exponente de 2, y así la máquina disponga de más memoria, será imposible accederla. Por ejemplo, si se tienen direcciones de 8 bits, solo se podrá acceder hasta 256 bits de memoria. Igualmente, ene l diseño del hardware se especifica el tamaño mínimo de bits a direccionar, lo que indica que las siguientes posiciones en memoria tendrán ese tamaño.

Cuando un proceso ha finalizado su ejecución, el espacio de memoria que estaba ocupando se libera y queda disponible para que se cargue otro proceso, de allí que el cargador se encargue de establecer las nuevas direcciones del proceso que se están cargando en memoria y una vez asignadas estas puedan ejecutarse; este proceso se puede realizar en cualquier momento de las siguientes tres etapas:

Tiempo de compilación: solo si se conoce la zona donde va hacer almacenado el programa al momento de ser compilado, dado que no generaría ningún inconveniente con el acceso a la memoria.

Tiempo de carga: si no se conoce dónde va hacer alojado el proceso cuando sea compilado, las direcciones deben permitir ser reasignables hasta cuando sean cargadas en la memoria.

Tiempo de ejecución: se recomienda realizarlo si no se presenta ningún inconveniente con retrasar el proceso de reasignación de direcciones.

La asignación de memoria a un proceso implica especificar las direcciones de la memoria a sus instrucciones y datos; constituye también un aspecto de una acción más general denominada “unión”, la cual hace referencia al acto de especificar un valor de un atributo de dirección de la memoria de una entidad.

De lo anterior, que existan dos tipos de uniones, la asignación estática y la dinámica, donde la primera hace referencia a la unión realizada antes del inicio de la ejecución de un programa, y la segunda, a la unión durante su ejecución.

La asignación estática es posible solo si antes de iniciar la ejecución de un programa se

conocen los tamaños de los datos. Si no se conocen, es necesario conjeturarlos, lo cual puede conducir a desperdicio de memoria y a falta de flexibilidad, por ejemplo, considere un arreglo cuyo tamaño se desconoce durante la compilación; se desperdicia memoria si su tamaño conjeturado es mayor que el real, mientras que un programa no puede ejecutarse correctamente si su tamaño real es mayor que el conjeturado. La asignación dinámica puede evitar estos problemas, ya que el tamaño real del arreglo debe conocerse en el instante de realizar una asignación.

La asignación estática de memoria no requiere acciones de asignación de memoria durante la ejecución de un programa.

La asignación dinámica incurre en sobrecargar de acciones de asignación de memoria realizadas durante la ejecución de un programa. Incluso, algunas de estas acciones se repiten varias veces durante dicha ejecución.

Intercambio

Para iniciar es importante entender el concepto de Intercambio, el cual hace referencia a una zona en el disco que se usa como memoria auxiliar, también es conocido como swap.

Para que un proceso se ejecute debe estar en la memoria, y haciendo uso de una parte del disco como intercambio, es posible aumentar el grado de multiprogramación.

Los autores Martínez, P. y Díaz, J. (1996) exponen que una solución para que la memoria no se vea sobrecargada por los procesos, es el intercambio de procesos, el cual consiste "…en intercambiar temporalmente un proceso que está en memoria con uno que se encuentre en la cola de procesos, guardando en un almacén de respaldo el proceso que se encuentra en memoria, y volviéndolo a llevar luego a memoria para continuar su ejecución".

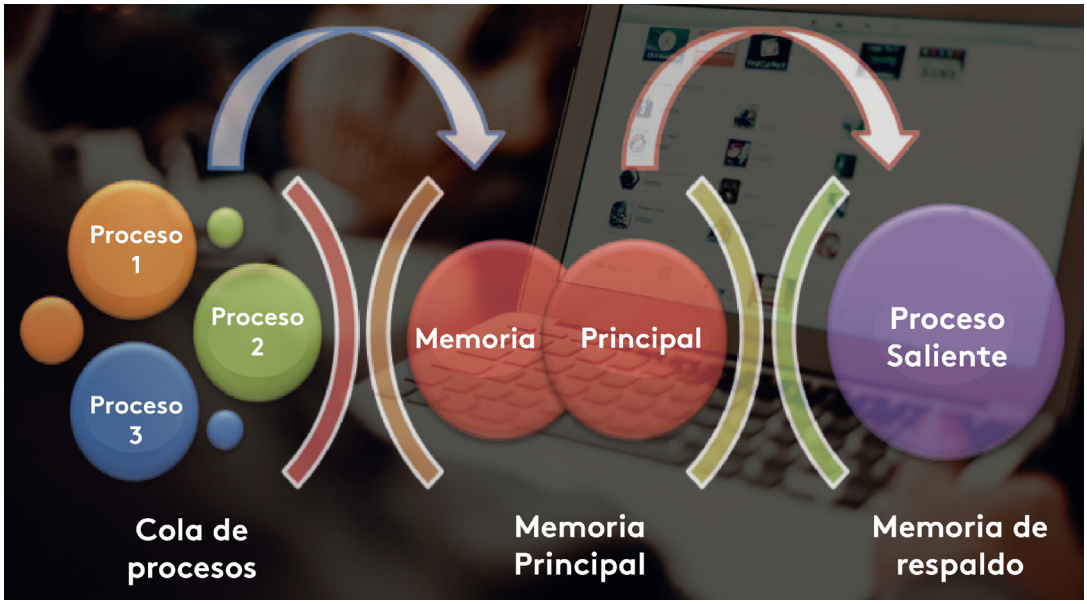


Figura 2. Intercambio de procesos.
Fuente: Martínez, P. y Díaz, J. (1996)

En la figura 2, se observa cómo se realiza el intercambio de procesos, para ello se debe tener una cola de procesos que se encuentran listos para ser ejecutados, de esta es seleccionado el proceso que ingresa a la memoria principal para luego pasar a la memoria de respaldo. Vale aclarar que para que se realice un proceso de intercambio, el proceso que se encuentre en memoria debe estar completamente inactivo.

Este proceso de intercambio tiene un costo de tiempo relativamente alto, donde este se da es por el proceso de trasferencia, dado por la cantidad de memoria intercambiada. Por esto el tiempo del proceso de intercambio es proporcional al tamaño de la memoria canjeada.

Este proceso de intercambio se le puede realizar mejoras para que sea aún más eficiente su labor,

1. Con la utilización de buffer que permita cargar y descargar un proceso mientras otro se encuentra en ejecución.
2. Con el uso de algoritmos de planificación con prioridad, donde se identifique si un proceso tiene mayor prioridad que otro el cual necesita ser ejecutado con anterioridad, cambiándolo por uno que se encuentre inactivo de baja prioridad.

Mecanismos de gestión de la memoria real

Asignación de memoria contigua

La asignación de memoria contigua es el modelo clásico de asignación de memoria, en cada proceso es asignado en una sola área contigua en la memoria. En los primeros sistemas de computadoras, la decisión de asignación de memoria se realizaba estadísticamente, es decir, antes de empezar la ejecución de un proceso. El sistema operativo determinaba la memoria requerida para ejecutar un proceso y asignaba suficiente memoria.

- Algunas de las ventajas de la asignación de memoria contigua son:
- Protección de la memoria.
- Relocalización estática y dinámica de un programa que se va a ejecutar desde el área de memoria que se le ha asignado.
- Medidas para evitar fragmentación de la memoria.

Para este caso vamos a referirnos al caso de fragmentación, dado que los otros ítems ya lo hemos abordado en el módulo.

Manejo de la fragmentación de la memoria

Existen dos tipos de fragmentación de la memoria, el primero, Multiprogramación con Particiones Fijas (MFT) el cual es interna y el segundo, Multiprogramación con Particiones Variables (MVT) la cual es externa.

Multiprogramación con Particiones Fijas (MFT), existe cuando la memoria asignada a un proceso no es completamente utilizada por este. Esta se presenta si el kernel asigna un área de la memoria de tamaño estándar a un proceso, independiente de su solicitud. A este proceso se le denomina como fragmentación interna.

Esto hace que no se utilice el total de la memoria siendo una de las principales deficiencias de este esquema de asignación.

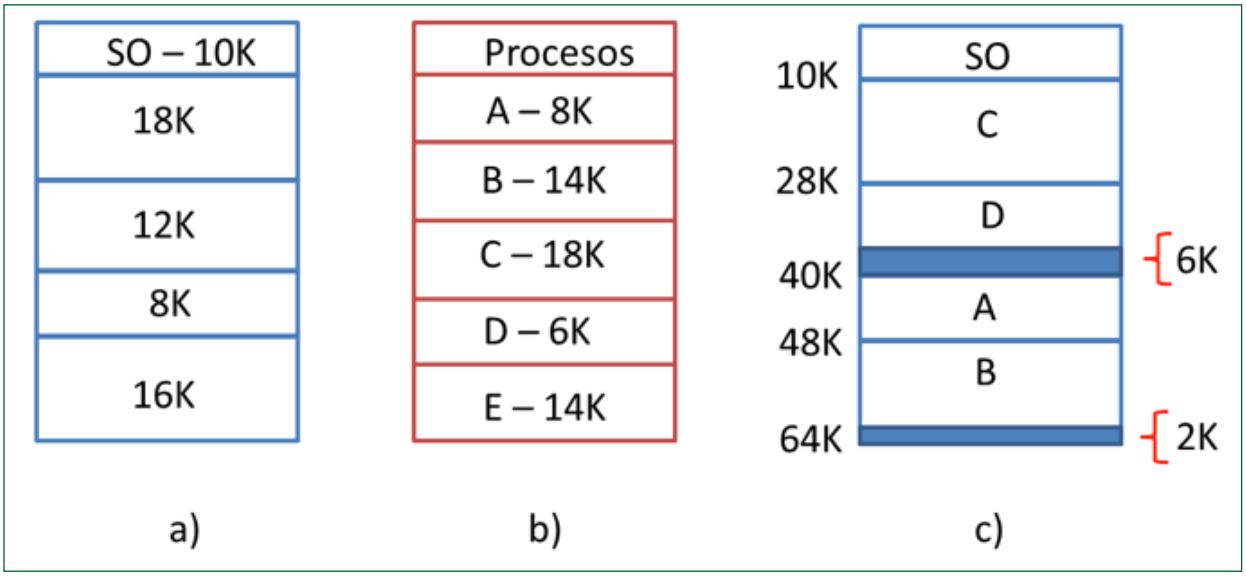


Figura 3. Particiones fijas
Fuente: Propia.

En la figura 3 se visualiza las particiones de fijas (a), y en (b) el tamaño de cada uno de los procesos los cuales necesitan ser ejecutados en la memoria, por lo tanto se revisa el tamaño de los procesos y el tamaño de la partición para ser asignados estos procesos a cada uno de las áreas. Si observamos el proceso C necesita una partición con un tamaño mínimo de 18K, la cual quedaría perfecta en la primera partición del Kernel, para el caso del proceso B, cabe perfectamente en la última partición del Kernel y de esta nos queda disponible 2K, lo mismo hacemos con cada uno de los procesos (c).

Ejemplo: se cuenta con un sistema de gestión de memoria con particiones fijas o MFT. El sistema operativo ocupa la primera posición que tiene un tamaño de 10K; el resto de la memoria está dividida en cuatro particiones cuyas características se visualiza en la Tabla 1. El algoritmo de ubicación asigna a cada proceso la partición más pequeña que quepa. En la tabla 2 se puede visualizar la cantidad de memoria que requiere cada proceso. Realice gráficamente lo solicitado.

Base partición	Tamaño partición
10	18K
28	12K
40	8K
48	16K

Tabla 1
Fuente Propia

Proceso	Tamaño partición
A	8K
B	14K
C	18K
D	6K
E	14K

Tabla 2
Fuente Propia

Tabla B

Solución: lo primero que hacemos es organizar la base de las particiones, como se visualiza en la figura,

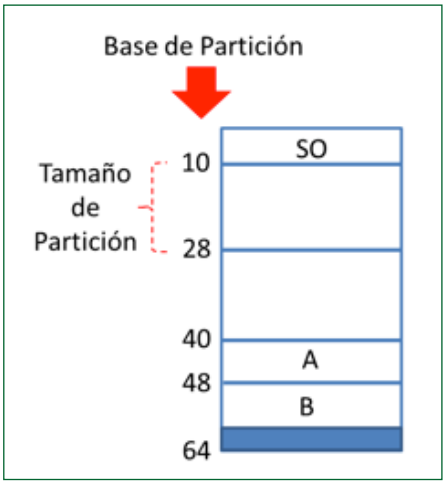


Figura 4
Fuente: Propia.

Segundo, verificamos el tamaño de cada proceso y verificamos en que partición quedaría esta, teniendo en cuenta que debe ser igual o de mayor tamaño.

Así nos quedaría la organización de cada uno de los procesos.

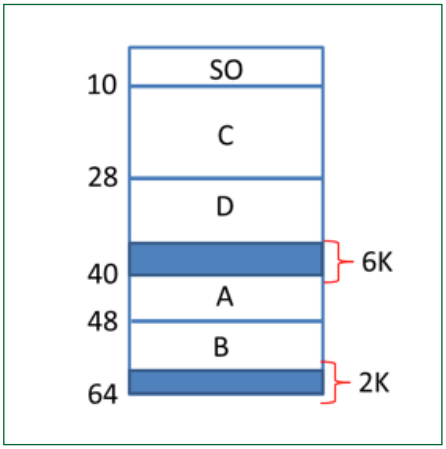


Figura 5
Fuente: Propia.

6K + 2K = 8K (de fragmentación interna)

El proceso E entra en memoria puesto que no hay espacio suficiente para albergarlo.

Multiprogramación con Particiones Variables (MVT), surge cuando las áreas de memoria libres que existen en un sistema son demasiado pequeñas para ser asignadas a procesos, a esto se le denomina fragmentación externa.

Analizaremos dos técnicas utilizadas para superar la fragmentación externa, compactación de memoria y reusó de fragmentos de memoria.

Compactación de memoria, el kernel puede realizar periódicamente compactación de memoria para eliminar fragmentación externa. Esta acción produce una sola área sin usar en la memoria, que puede ser suficientemente grande para albergar uno o más procesos nuevos, incluso en casos en que áreas libres individuales antes de la compactación sean muy pequeñas para este propósito.

La compactación implica el movimiento de procesos en la memoria durante su ejecución.

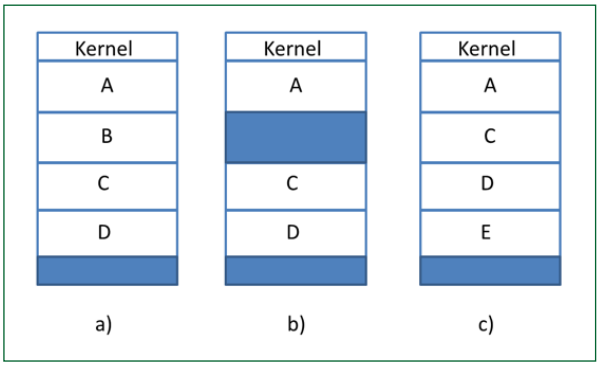


Figura 6. Compactación de la memoria
Fuente: Propia.

En la figura 6 se visualiza los procesos A, B, C y D que existen en la memoria (a), cuando B termina, hay dos procesos activos (b), el kernel lleva a cabo compactación para crear una sola área e iniciar el proceso E en esta área (c).

Reusó de fragmentos de memoria, el reusó de memorias libres evita la sobrecarga de relocalización del programa y no requiere prestaciones especiales de hardware como un registro de relocalización. Sin embargo pueden existir retrasos en el inicio de la ejecución de los programas. La diferencia entre esta técnica y la de compactación es que el tamaño del espacio libre debe ser superior al que se va asignar para que este se pueda reutilizar.

Ejemplo: un sistema es gestionado con un mecanismo de Multiprogramación de Particiones Variables (MVT), en el que la memoria física tiene 4200 palabras. En un instante la memoria está ocupada por 3 bloques de código/datos de la siguiente forma:

Bloque	Dirección inicial	Longitud
Bloque 1	1000	1000
Bloque 2	2900	500
Bloque 3	3400	800

Tabla 3
Fuente Propia

Tenga en cuenta que en primer lugar se utilizará la estrategia de mejor ajuste. De lo anterior, se debe cargar tres bloques con a las siguientes características:

Bloque	Longitud
Bloque 4	500
Bloque 5	1200
Bloque 6	200

Tabla 4
Fuente Propia

Solución: teniendo en cuanta la información inicial entregada, la situación de la memoria es:

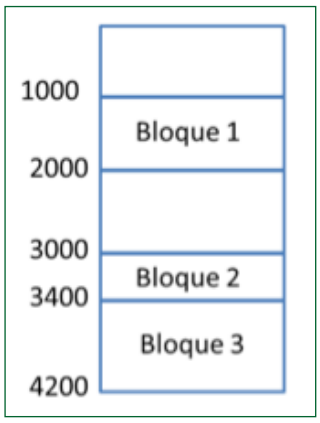


Figura 7
Fuente: Propia.

Con la información anterior, cargamos el Bloque 4 de 500 palabras utilizando la opción de mejor ajuste.

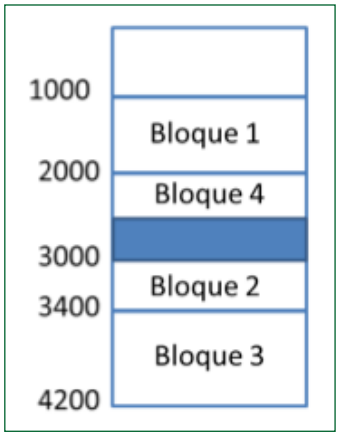


Figura 8
Fuente: Propia.

Seguido cargamos el Bloque 5 con 1200 palabras, pero si observamos la figura anterior, podemos notar que hay espacio suficiente en memoria pero el bloque no cabría es este, de lo anterior que tengamos un problema de fragmentación externa, por tanto se hace necesario utilizar un algoritmo de compactación.

Por último, si observamos la figura anterior, nos queda libre 1400 palabras una vez realizado el proceso de compactación y si sumamos la longitud de los bloques 5 y 6 esta nos da 1400 palabras, lo cual no tendríamos inconveniente en ubicarlos en dichos espacios.

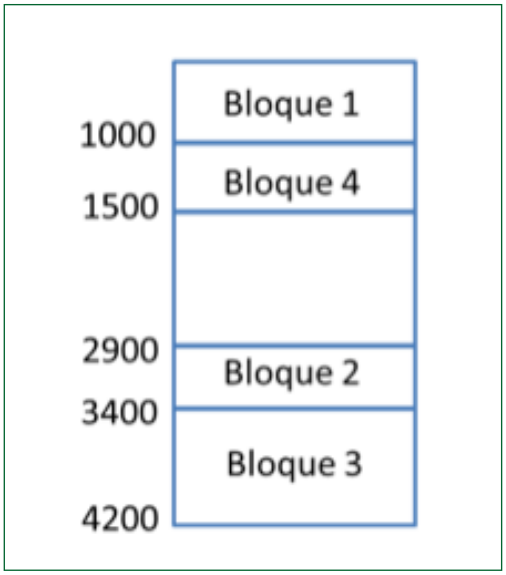


Figura 9
Fuente: Propia.

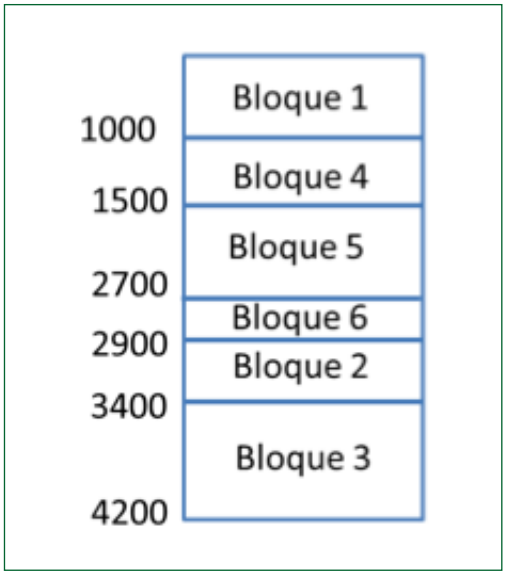
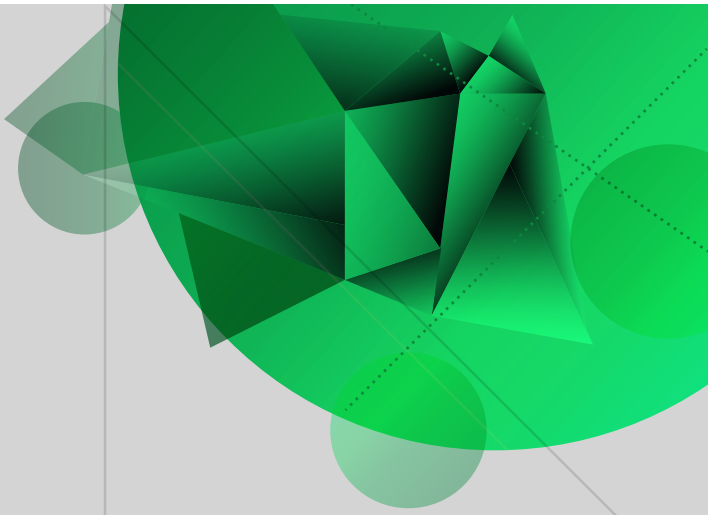


Figura 10
Fuente: Propia.

3

Unidad 3

Gestión de memoria virtual



Sistemas operativos

Autor: Katherine Roa

AREANDINA
Fundación Universitaria del Área Andina

MIEMBRO DE LA RED
ILUMNO

Introducción

En el capítulo anterior se trabajó el concepto memoria real desde los diferentes mecanismos de gestionar la memoria desde el contexto de asignación contigua. Para esta oportunidad abordaremos la memoria virtual pasando por los diferentes conceptos iniciales, conociendo las implementaciones de la memoria virtual desde el modelo de asignación no contigua y finalizaremos con las políticas que se deben tener en cuenta al momento de gestionar una memoria virtual.

U3

Metodología

Para un mayor entendimiento de este capítulo, se recomienda al estudiante realizar una lectura de la guía y generar un reporte crítico del mismo a modo personal, con el fin de afianzar los conocimientos adquiridos en esta cartilla.

Se sugiere adicionalmente, desarrollar las diferentes lecturas y actividades planteadas en la semana, con el fin de comprender mejor la temática trabajada.

U3

Desarrollo temático

Gestión de memoria virtual

Introducción

La memoria virtual es una ilusión de que un sistema de cómputo posee más memoria de la que realmente tiene, lo que permite que un proceso sea independiente del tamaño de la memoria real y que un número de procesos comparta un sistema de cómputo sin restringirse mutuamente. La memoria virtual se implementa a través de la parte de la jerarquía de la memoria que consta de una memoria y un disco; el código y los datos de un proceso se almacenan en un disco y partes de aquellos se cargan en la memoria cuando son necesarios durante la ejecución del proceso. Utiliza el modelo de asignación no contigua y contiene componentes tanto del hardware como del software.

El desempeño de la memoria virtual depende del ritmo al que debe cargarse las partes de un proceso en la memoria desde un disco. El principio de localidad de referencia mantiene la promesa de que este ritmo debe ser bajo si a un proceso se le asigna una cantidad idónea de memoria. El aspecto práctico de la memoria virtual depende entonces del control de la cantidad de memoria. El aspecto práctico de la memoria virtual depende entonces del control de la cantidad de memoria asignada a un proceso y de la decisión sobre qué partes de un proceso debe preservarse en la memoria virtual.

Los usuarios siempre quieren más de un sistema de cómputo: más recursos y más servicios. La necesidad de más recursos se satisface obteniendo un uso más eficaz de los recursos o creando la ilusión de que en el sistema hay más recursos. Una memoria virtual es lo que indica su nombre: una ilusión de que en el sistema de cómputo hay una memoria más grande que la real, es decir, que la RAM.

El kernel implementa la ilusión utilizando la combinación de medios de hardware y software. La componente del software de la memoria virtual se denomina manejador de la memoria virtual.

La base de la memoria virtual es el modelo de asignación de memoria no contigua, se supone que cada proceso consta de dos partes denominadas componentes del proceso. Estas pueden cargarse para su ejecución en áreas de memoria adyacentes. La unidad de administración de la memoria (MMU) traduce la dirección de cada operando o instrucción utilizado mediante un proceso en la dirección del byte de memoria donde reside en realidad el operando o la instrucción. El uso del modelo de asignación de memoria no contigua redice el problema de fragmentación de la memoria porque un área libre

de memoria puede volver a usarse incluso si no es suficientemente grande para admitir todo un proceso. De esta forma es posible alojar más procesos en la memoria, lo cual redunda en beneficio tanto de los usuarios como del sistema operativo.

La ilusión de una gran memoria es creada al permitir la ejecución de un proceso cuyo tamaño excede al de la memoria. Este se logra manteniendo un proceso en un disco y cargando en la memoria en cualquier instante solo sus porciones requeridas. El kernel utiliza esta idea para reducir la asignación de memoria a procesos en general, es decir, incluso los procesos que pueden caber en la memoria no cargan totalmente en elle. Esta estrategia incrementa aún más el número de procesos que pueden alojarse en la memoria.

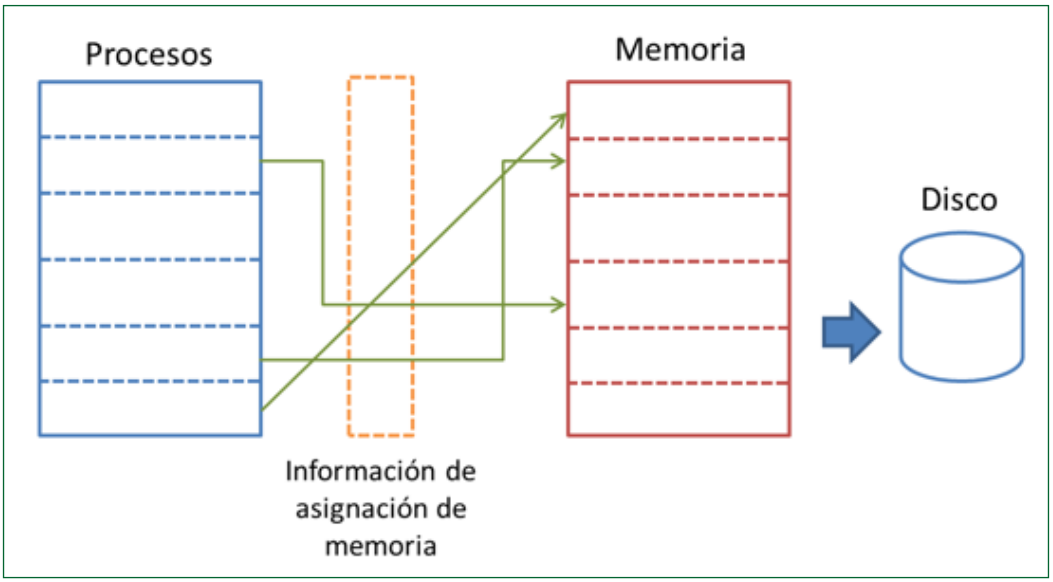


Figura 1. Visión general de la memoria virtual
Fuente: Propia.

La figura 1 visualiza el esquema de la memoria virtual. Un proceso contiene seis componentes, tres de estas se encuentran en ese momento en la memoria. La información sobre las áreas de la memoria donde existen las componentes se mantiene en una estructura de datos del manejador de la memoria virtual. La MMU utiliza esta información durante la traducción de la dirección. Cuando una instrucción en el proceso se refiere a una componente del proceso que no está en la memoria, está cargada desde el disco, ocasionalmente, el manejador de la memoria virtual elimina algunas componentes del proceso de la memoria para tener espacio y poder organizar otras.

Implementaciones de la memoria virtual

Asignación de memoria no contigua

En el modelo de asignación no contigua, varias áreas de memoria no adyacentes son asignadas a un proceso. Ninguna de éstas es suficientemente grande para admitirlo todo, lo que una parte del proceso es cargado en cada una de estas áreas.

Una de las ventajas de esta asignación de memoria no contigua es que un área de memoria que no es suficientemente grande para llevar a cabo un proceso completo aún puede ser utilizada, de modo que resulta menos fragmentación externa. Esta característica reduce la fragmentación externa y mejora la utilización de la memoria; en consecuencia, puede ser innecesario llevar a cabo fusión de áreas de memoria libres o compactación.

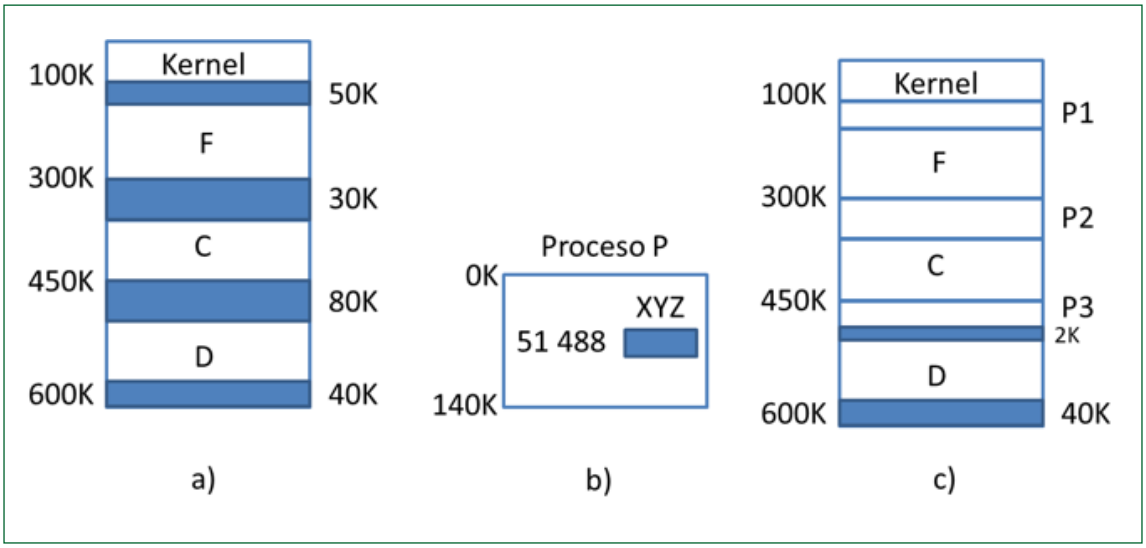


Figura 2. Asignación de memoria no contigua
Fuente: Propia.

En la figura 2 se visualiza que en la memoria hay cuatro áreas de memoria no asignadas de 50KB, 30KB, 80KB y 40KB (a). Se iniciará un proceso P de tamaño de 140 KB. El proceso se separa en tres componentes denominadas P1, P2 y P3 que se cargan en tres de las áreas libres (c).

La asignación no contigua utiliza dos métodos: paginamiento y segmentación.

Paginamiento

En el paginamiento cada proceso consta de componentes de tamaño fijo denominadas páginas. El tamaño de una página se especifica en la arquitectura del sistema del cómputo.

En un sistema en el que se utilice paginamiento, un proceso es considerado como una entidad contigua que contiene instrucciones y datos. En la vista lógica, un proceso consta de un arreglo lineal de páginas. Cada una contiene s bytes, donde s es una potencia de 2. El valor de s se especifica en la arquitectura del sistema del cómputo. Los procesos utilizan direcciones lógicas numéricas. El hardware de la máquina descompone una dirección lógica en un par (pi, bi), donde pi es el número de páginas y bi es un desplazamiento. La vista física consta de dos áreas de memoria no adyacentes asignadas a páginas del proceso.

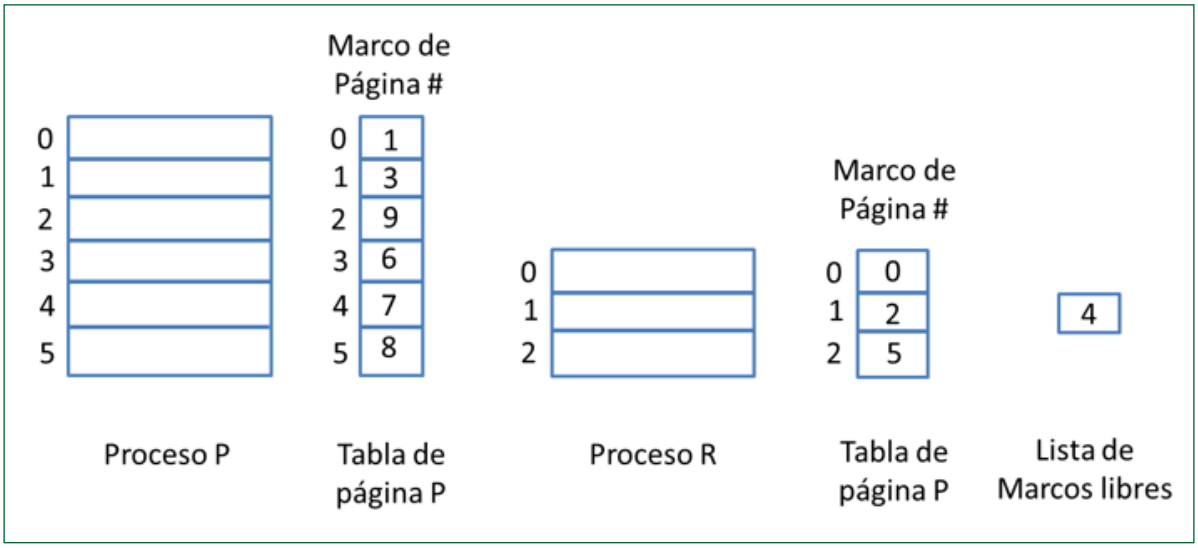


Figura 3. Procesos de paginamiento
Fuente: Propia.

En la figura 3 se visualiza un arreglo utilizado para ejecutar los procesos P y R en un sistema que usa paginamiento. El tamaño de P es de 5.500 bytes. El de la página es de 1 KB (1024 bytes), de modo que P tiene 6 páginas. Estas páginas están numeradas del 0 al 5 y los bytes en una página están numerados de 0 a 1.023. La última página solo tiene 371 bytes (5500/1024). Si un elemento de datos simple tiene la dirección de 5.248, entonces el hardware de la máquina ve esta dirección como el par (5, 128). El proceso R tiene 3 páginas numeradas del 0 al 2.

El Kernel parte la memoria en áreas denominadas marcos de página. Cada uno es del mismo tamaño que la página; es decir, 1 KB. En la figura 6 la maquina tiene una memoria de 10KB

Por lo que los marcos de página están numerados del 0 al 9. En cualquier momento, algunos marcos de página se asignan a páginas del programa y otros quedan libres. El kernel mantiene una lista denominada lista de marcos libres para notar las identificaciones de los marcos de página libres. Por el momento, solo el marco de página 4 es libre.

A la vez que carga el proceso P para su ejecución, el kernel consulta la tabla de marcos y asigna un marco de página libre a cada marco de página de los procesos. A fin de facilitar la traducción de la dirección, se elabora una tabla de página (PT) para el proceso. Cada elemento de la tabla de página indica el número de página. Durante la ejecución del proceso, la MMU consulta la tabla de página para realizar la traducción de la dirección.

En la figura 3, la tabla de marco indica que seis marcos de página están ocupados por el proceso P, por un proceso R y uno está libre. La tabla de página P indica los números de marco asignados a las páginas de P. la dirección lógica debe traducirse a la dirección física 8 320 utilizando el elemento para la página 5 en la tabla de página.

Traducción de la dirección, se utiliza la siguiente notación:

s: tamaño de página.

lt: longitud de una dirección lógica (número de bits que contiene).

lh: longitud de una dirección física.

nb: número de bits necesarios para acceder al último byte de una página.

np: número de bits usados para alojar el número de página en una dirección lógica.

nf: número de bits para alojar el número de páginas en una dirección física.

El tamaño de una página, s, es potencia de 2; por ejemplo s= 2nb. Por lo tanto, los bits menos significativos nb en una dirección lógica proporcionan bi. Los bits restantes en una dirección lógica forman a pi. Los valores de pi y bi pueden obtenerse simplemente agrupando los bits de una dirección lógica como sigue:

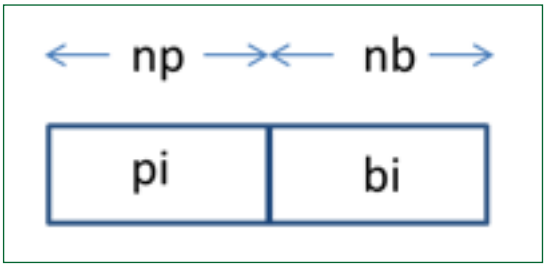


Figura 4
Fuente: Propia.

Donde np=lt-nb. El uso de una potencia de 2 como tamaño de página simplifica de manera semejante la elaboración de la dirección de la memoria efectiva. Considere que a la página pi se ha asignado el marco de página qi. Debido a que el tamaño de las páginas y el de los marcos de página es el mismo, también se requieren nb bits para direccionar los bytes en un marco de página. Sea nf=lp-nb. La dirección física del byte "0" del marco de página qi es

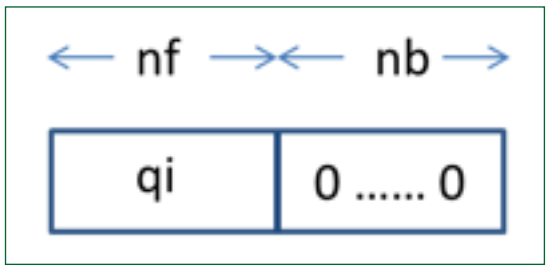


Figura 5
Fuente: Propia.

Por tanto, la dirección física del byte bi en el marco de página qi está dada por

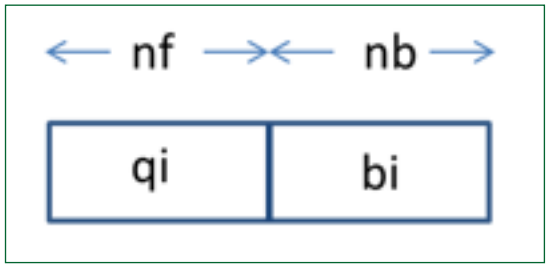


Figura 6
Fuente: Propia.

La MMU puede obtener está dirección simplemente al concatenar qi y bi a fin de obtener un número de bit lh. La traducción de la dirección.

Ejemplo: estamos trabajando con un sistema operativo que emplea una gestión de memoria paginada, cada página tiene un tamaño de 4096 posiciones (o bytes). La memoria física disponible para los procesos es de 16MB. Primero llega un proceso que necesita 63132 posiciones de memoria y después llega otro proceso que consume 36864 posiciones cuando se carga una memoria ¿Qué fragmentación interna genera cada proceso.

Solución:

Tenemos:

Proceso 1: 63132 posiciones.

Proceso 2: 36864 posiciones.

Memoria: 16MB.

Tamaño de páginas: 4096 bytes.

Para el caso del proceso 1.

63132 / 4096 = 15.4 Posiciones.

Lo que indica que tenemos fragmentación interna, dado que el resultado tiene decimales, veamos de cuanto es la fragmentación.

15 * 4096 = 61440.

63132 – 61440 = 1692 Bytes.

Para el caso del proceso 2.

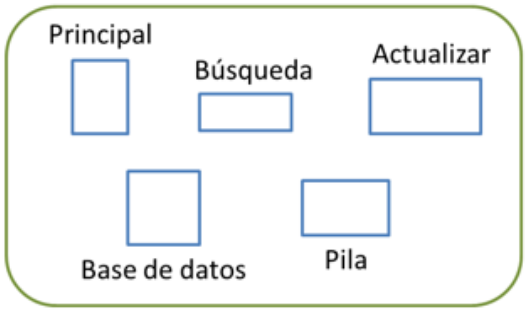
36864 / 4096 = 9 Posiciones.

Lo anterior indica que no tenemos fragmentación.

Segmentación

Un segmento es una entidad lógica en un programa; por ejemplo, una función, una estructura de datos o un objeto. Por tanto, resulta importante administrar un segmento como si fuese una unidad: cargarlo en la memoria para su ejecución o compartirlo con otros programas. En la vista lógica, un proceso consta de un conjunto de segmentos. La vista física consta de áreas no adyacentes de memoria asignadas a segmentos.

Un proceso Q consta de cuatro unidades lógicas con los nombres simbólicos main, database, search, y retrieve, mientras codifica el programa, el programador declara estas unidades lógicas como segmentos en Q. el compilador genera direcciones lógicas mientras traduce el programa. Cada dirección lógica utilizada en Q tiene la forma (si, bi) donde si y bi son las identificaciones de un segmento y un byte dentro de un segmento. Por ejemplo, la instrucción correspondiente a una declaración call get-sample, donde get-sample es un procedimiento en el segmento retrieve, puede utilizar la dirección de operando (retrieve, get-sample), o usar una representación numérica para si y bi.



Nombre	Tamaño	Dirección
main	476	23500
database	20240	32012
search	378	76248
update	642	91376
stack	500	54500

Tabla de segmento de Q

Figura 7. Procesos de segmentación
Fuente: Propia.

En la figura 7, se visualiza la forma en que el kernel maneja al proceso Q. la mitad izquierda de la figura muestra la vista lógica del proceso Q, a fin de facilitar la traducción de la dirección, el kernel elabora una tabla de segmento para Q. cada elemento en esta tabla muestra el tamaño de segmento de Q y direcciones del área de la memoria que se le han asignado. La MMU utiliza la tabla de segmento para llevar a cabo la traducción de la dirección. Los segmentos no tienen tamaños estándares, de modo que no es aplicable la simplificación de la concatenación de bits usada en el paginamieto. En consecuencia, el cálculo de la dirección de la memoria efectiva implica la adición de bi a la dirección inicial de si, por lo que la traducción de la dirección es más lenta que el paginamiento. En la figura 7, si get-sample tiene la compensación de 232 en el segmento retrieve, entonces la traducción de la dirección (retrieve, get-sample) debe producir la dirección 91376 + 232 = 91608.

La asignación de memoria para cada segmento se lleva a cabo como el modelo de asignación de memoria no contigua. El kernel mantiene una lista libre de áreas de la memoria. Mientras carga un proceso, busca en esta lista para realizar la asignación del primer ajuste o del mejor ajuste a cada segmento del proceso. Una vez que termina un proceso, las áreas de la memoria asignadas a sus segmentos se agregan a la lista libre. Hay fragmentación externa porque el tamaño de segmentos varía.

La tarea de escribir cada dirección lógica en la forma (si, bi) se realiza mediante un compilador, mientras compila un referencia a un símbolo xyz, debe decidir a qué segmento pertenece xyz.

Ejemplo: dada la siguiente tabla de segmentos, identifique cuales son las direcciones físicas en las direcciones lógicas,

Segmentos (s)	Límite (l)	Base (b)
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

Tabla 1
Fuente: Propia.

Segmento	Desplazamiento (d)
0	430
1	10
2	500
3	400
4	112

Tabla 2
Fuente: Propia.

Solución:

Para el primer caso,

S = 0 , b = 219 , l = 600 , d = 430

Lo primero que se debe hacer es identificar si el desplazamiento es menor que la longitud (d<l).

430 < 600 (cumple).

Luego tomamos la base y la sumamos con el desplazamiento (b+d), de esto obtenemos la dirección física.

219 + 430 = 649 (dirección física).

Para el segundo caso,

S = 1 , b = 2300 , l = 14 , d = 10

d<l = 10<14 (cumple).

2300 + 10 = 2310 (dirección física).

Tercer caso,

S = 2 , b = 90 , l = 100 , d = 500

d<l = 500<100 (no cumple).

Cuarto caso,

S = 3 , b = 1327 , l = 580 , d = 400

d<l = 400<580 (cumple)

1327 + 400 = 1727

Quinto caso,

S = 4 , b = 1952 , l = 96 , d =1120

d<l = 1120<96 (no cumple).

Por último, realizamos el gráfico de los segmentos:

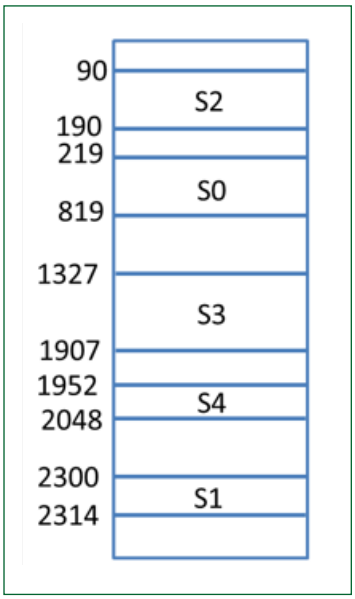


Figura 8
Fuente: Propia.

Segmentación con paginamiento

En este método, cada segmento en un programa se pagina por separado. En consecuencia, a cada segmento se asigna un número entero de páginas. Este método simplifica y acelera la asignación de memoria, y también evita la fragmentación externa. Para cada segmento se elabora una tabla de página, y en el elemento del segmento en la tabla de segmentos se mantiene un apuntador hacia la tabla de página. Luego, la traducción de la dirección para una dirección lógica (si, bi) se realiza en dos etapas. En la primera, el elemento de si se localiza en la tabla de página, luego, el número de byte bi se separa en un par (psi ,bpi), donde psi es el número de páginas en el segmento si, y bpi es el número de byte en la página pi. Después el cálculo de la dirección efectiva se termina como en paginamiento; es decir, se obtiene el número de marco de psi y bpi se concatena con este para obtener la dirección efectiva.

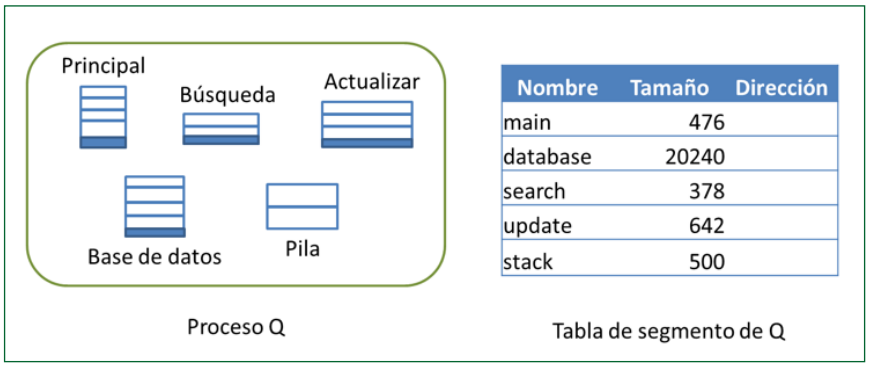


Figura 9. Proceso Q en segmentación y paginamiento
Fuente: Propia.

En la figura 5 se visualiza el proceso Q de la figura 4 en un sistema que utiliza segmentación con paginamiento. Cada segmento se página de forma independiente, por lo que

en la última página de cada segmento hay fragmentación interna. Entonces, cada elemento de la tabla de segmentos contiene un apuntador hacia la tabla de página del segmento. El campo de tamaño en un elemento del segmento se utiliza para facilitar una comprobación de límites para protección de la memoria.

Políticas de gestión de la memoria virtual

Con el fin de gestionar correctamente un sistema con memoria virtual, se debe tener clara las reglas para gestionar el intercambio de páginas o segmentos entre la memoria principal y la secundaria, de allí que se tengan algunas políticas que se debe cumplir:

Políticas de lectura, el sistema operativo debe determinar en qué momento carga las páginas en la memoria principal, teniendo en cuenta las siguientes opciones:

- Paginación por demanda: carga una página en la memoria principal solo cuando sea necesario, al iniciar la ejecución de un proceso o por algún fallo de página.
- Paginación previa: se trasladan a memoria páginas extras a las solicitadas.

Política de reemplazo, cuando la memoria se encuentre llena y se necesita cargar otra página, el sistema operativo debe decidir que página mandar a memoria secundaria. El reemplazo puede ser global (cualquier página), local (solo las páginas que provocó el fallo).

Las políticas que se tiene en cuenta para elegir la página son:

Menos usada recientemente (Least Recently Used, LRU): se selecciona la página utilizada hace con anterioridad, la cual tenga menos probabilidad de ser usada nuevamente.

Usado más Recientemente (Most Recently Used, MRU): se selecciona la página que fue utilizada recientemente.

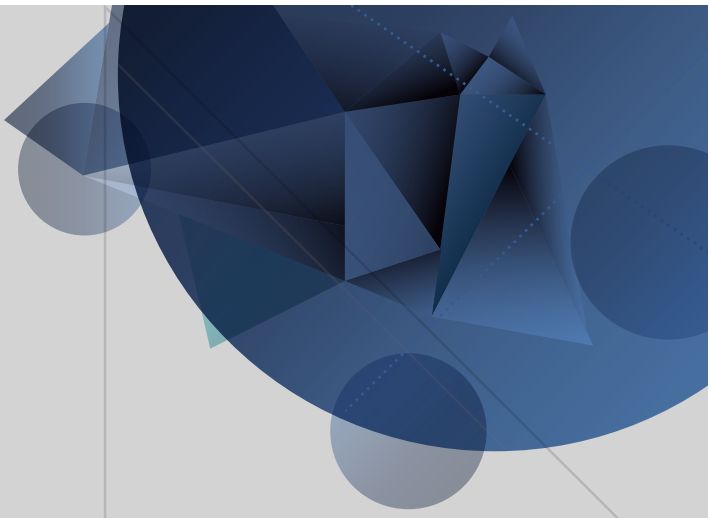
Primera en entrar, primera en salir (FIFO, First In, First Out): se selecciona la página que lleva más tiempo en memoria.

Técnica de reloj: cada página tendrá un bit de control y está selecciona la página que se encuentre en la cabeza.

4

Unidad 4

Gestión de ficheros



Sistemas operativos

Autor: Katherine Roa

AREANDINA
Fundación Universitaria del Área Andina

MIEMBRO DE LA RED
ILUMNO

Introducción

Nos acercamos al final del módulo y para este capítulo abordaremos las diferentes temáticas que encierra la Gestión de Ficheros, iniciando por conocer los conceptos de fichero y directorio, donde también son conocidos como archivos y carpetas respectivamente, luego abordaremos las diferentes semánticas de consistencia y finalizaremos con la implementación del sistema de fichero.

U4 Metodología

Para un mayor entendimiento de este capítulo, se recomienda al estudiante realizar una lectura de la guía y generar un reporte crítico del mismo a modo personal, con el fin de afianzar los conocimientos adquiridos en esta cartilla.

Se sugiere adicionalmente, desarrollar las diferentes lecturas y actividades planteadas en la semana, con el fin de comprender mejor la temática trabajada.

U4

Desarrollo temático

Gestión de ficheros

Introducción

Los usuarios de una máquina almacenan datos en archivos (o ficheros) de modos que puedan usarlos convenientemente y de manera repetida. Un usuario tiene muchas expectativas de un sistema de archivo, algunas de estas son:

- Acceso conveniente y fácil a los archivos.
- Almacenamiento confiable de los archivos.
- Poder compartir de manera controlada los archivos con otros usuarios del sistema.

Los recursos utilizados para este propósito son dispositivos de Entrada/Salida (E/S); su capacidad para almacenar datos y sus velocidades para transferirlos. Como otros recursos, el sistema operativo debe asegurar la utilización eficaz de los dispositivos de E/S.

En muchos sistemas operativos, las funciones mencionadas están organizadas en dos componentes denominados sistemas de archivos y sistema de control de entrada salida (IOCS). El sistema de archivo proporciona servicios que permite al usuario crear archivos, asignarles nombres con sentido, manipularlos y especificar la manera en que serán compartidos con otros usuarios del sistema. El IOCS implementa la organización eficaz y el acceso a los datos en los archivos. Así, el uso del sistema de archivos y del IOCS separa de manera conveniente lo relacionado con el nivel de archivo a partir de cuestiones relativas a E/S.

Ficheros

El sistema de archivo o también llamado fichero, y los módulos IOCS constituyen la jerarquía de las capas que se muestran en la figura 1, cada capa contiene la política y los módulos del mecanismo. Los mecanismos de una capa se implementan utilizando la política y los módulos del mecanismo.

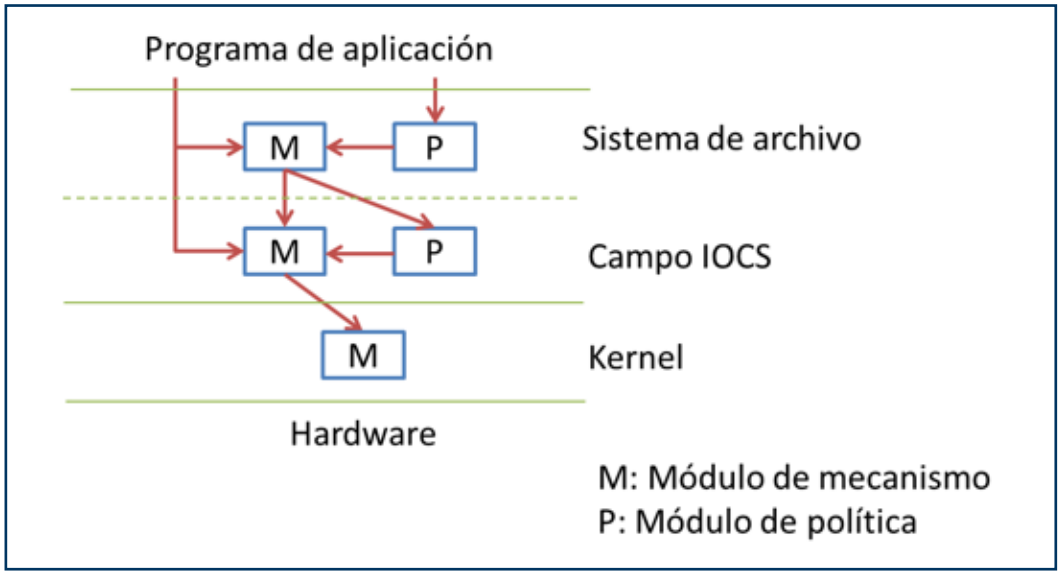


Figura 1. Sistema de archivos y capas IOCS
Fuente: Propia.

La jerarquía del sistema de archivo y las capas IOCS proporcionan una jerarquía de abstracciones.

El kernel interactúa con el hardware de E/S y proporciona facilidades para manejar el inicio y termino de E/S. las capas de IOCS invocan los servicios del kernel a través de llamadas del sistema, utilizando los mecanismos IOCS, es posible efectuar E/S sin conocer los vericuetos de los dispositivos de E/S. Los módulos de la política IOCS aseguran un uso eficaz del subsistema E/S y un buen desempeño de los programas de procesamiento de archivos. Estos invocan los módulos del mecanismo para implementar acciones de E/S. La capa del sistema de archivos utiliza servicios proporcionados por la capa IOCS para implementar sus funciones. Este arreglo oculta todos los detalles de la organización de E/S y las interfaces del módulo IOCS del programa de aplicación; el programa de aplicación solo interactúa con el sistema de archivo.

El sistema de archivo, considera un archivo como una entidad que es propiedad del usuario, que puede ser compartida por un conjunto de usuarios autorizados y que debe ser almacenada de manera confiable durante un intervalo de tiempo. Como un aspecto de la propiedad, proporciona libertad de nombres de los archivos, de modo que el usuario puede asignarle un nombre deseado a un archivo sin preocuparse que otros usuarios hayan creado nombres idénticos, y proporciona privacidad al proteger contra interferencia de otros usuarios.

El IOCS considera un archivo como un conjunto de registros a los que es necesario acceder rápidamente y almacenar en un dispositivo de E/S que requiere ser utilizado eficazmente.

El sistema de archivo provee las estructuras de directorio que permiten que un usuario organice sus datos en grupos lógicos de archivos. Por ejemplo, un usuario podrá

preferir separar datos personales de datos profesionales y estructurar los datos profesionales según las actividades. El sistema de archivo proporciona protección contra acceso ilegal a los archivos y reglas para compartirlos concurrentemente. También asegura que los datos se almacenen de manera confiable, es decir, que ningún dato se pierda cuando ocurra una caída del sistema. El IOCS proporciona mecanismos para efectuar operaciones de E/S y para asegurar el uso eficaz de los dispositivos de E/S. también provee un conjunto de módulos de biblioteca que permiten que un programa procese de manera eficaz un archivo.

Tipos de archivos

Un sistema de archivo contiene diferentes tipos de archivos; por ejemplo, archivos que contienen datos, programas ejecutables, módulos objeto, información textual, documentos, hojas de cálculos, fotos, videos, entre otros. Cada uno de estos archivos tiene su propio formato. Estos tipos de archivos pueden clasificarse en dos clases:

- Archivos estructurados.
- Archivos de flujo de bytes.

Archivos estructurados, es una vista clásica de un archivo que consta de registros y campos, en esta vista un archivo se denomina colección de registros. Un registro es una colección significativa de campos relacionados, y un campo contiene un solo dato. Un registro, es una unidad significativa para el procesamiento de datos. Se supone que cada registro en un archivo contiene un campo llave, el contenido de los campos llave de todos los registros en un archivo es único. Muchos tipos de archivos antes mencionados son archivos estructurados. Los tipos de archivo usados por software del sistema estándar, como compiladores y ligadores, poseen una estructura determinada por el diseñado del SO. La estructura de un tipo de archivo utilizado por una aplicación como un programa de hojas de cálculo es determinada por la aplicación misma. La estructura de un archivo de datos es determinada por el programa que lo crea.

Un archivo de flujo de bytes es plano, no contiene campos ni registros; los procesos que lo utilizan lo consideran como una secuencia de bytes. Unix usa este tipo de archivos para almacenar datos y programas.

Atributos de los archivos

Los atributos de los archivos son características importantes para sus usuarios o para el SO. Los atributos comunes de archivos son los siguientes: tipo, organización, tamaño, ubicación en el sistema, información de control de acceso que indica la manera en que los diferentes usuarios pueden acceder al archivo, nombre del propietario y hora de la última vez que se usó.

Operaciones de los archivos

El sistema operativo es el responsable de crear y mantener un archivo, para asegurar que a él acceden los usuarios solo conforme a los privilegios de acceso especificados para el archivo, y para eliminar el archivo cuando así los solicite su propietario. Estas funciones son realizadas mediante el empleo de la interfaz del sistema de archivos. El verdadero acceso a los archivos, es decir, la lectura o escritura de registros, se implementan utilizando el IOCS.

Apertura de archivos, un proceso ejecuta una declaración open antes de realizar el procesamiento de un archivo. El sistema de archivos localiza el archivo y verifica si

el usuario que está ejecutando el proceso cuenta con los privilegios de acceso necesarios.

Lectura o escritura de un registro, el sistema emite un comando idóneo para leer o escribir un registro. El sistema de archivos considera la organización del archivo e implementa la operación de la lectura/escritura de manera apropiada.

Cierre de un archivo, la ejecución de una declaración close indica al sistema de archivos que se ha terminado el procesamiento del archivo. El sistema de archivos actualiza en sus estructuras de datos la información concerniente al tamaño del archivo, es decir, al número de registros que contiene.

Creación de un archivo, un archivo se crea haciendo una copia de un archivo existente o escribiendo registros en un nuevo archivo. El sistema almacena información relacionada con los privilegiados de acceso para el archivo.

Borrado de un archivo, el archivo se borra de la estructura del directorio del sistema de archivo, el medio de almacenamiento secundario ocupado por este archivo se libera.

Volver a nombrar un archivo, el sistema de archivos recuerda el nuevo nombre del archivo en su estructura del directorio.

Especificación de los privilegios de acceso, el usuario puede especificar o modificar los privilegios de acceso de un archivo creado por su proceso en cualquier momento de la existencia del archivo.

Directorios

Un directorio también es conocido como "carpeta". Un sistema de archivos contiene archivos que son propiedad de varios usuarios. En este contexto son importantes dos características:

- Libertad de nombrado: capacidad del usuario para dar cualquier nombre a un archivo sin estar restringido por los nombres de los archivos utilizados por otros usuarios.
- Compartir archivos: capacidad del usuario para acceder a archivos creados por otro usuario, y de permitir a otros usuarios el acceso a sus archivos.

Un sistema de archivos utiliza directorios para proporcionar estas dos características. Un directorio contiene información relacionada con un archivo: su ubicación, tipo y manera en que otros usuarios del sistema pueden acceder a él. Cuando un proceso emite un comando para abrir un archivo, el sistema de archivos encuentra la entrada del archivo en un directorio y obtiene la localización.

En la figura 2 se visualiza campos en una entrada de directorio típica. El campo flags se utiliza para distinguir entre diferentes tipos de entrada de directorio. El valor de "D" en este campo se pone para indicar que un archivo es un directorio, "L" para indicar que se trata de una liga y "M" para indicar que se trata de un sistema de archivos montado. Estos usos se describen en secciones posteriores. El campo misc info contiene información de tipo: propietario, momento de creación y momento de la última modificación.

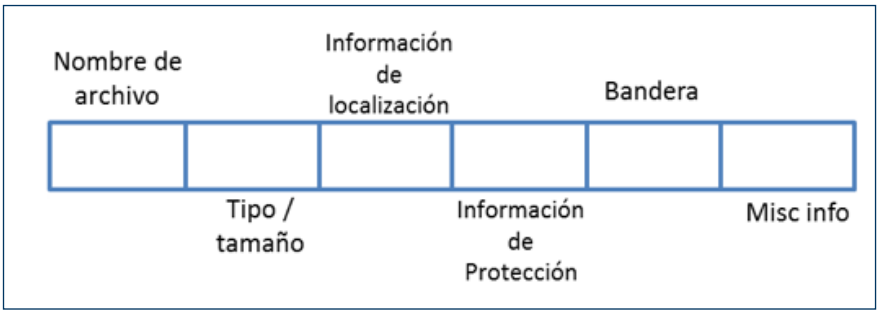


Figura 2. Entrada de directorio típica
Fuente: Propia.

Un sistema de archivo contiene varios directorios. La estructura del directorio del sistema de archivos conecta un directorio con otros en el sistema. Rige la manera en que los archivos pueden ser compartidos por un grupo de usuarios. Se utiliza la convención pictórica de que un directorio se presenta por un rectángulo, mientras un archivo se representa por un círculo. En la figura 3 se visualiza una simple estructura del directorio que incluye a dos directorios. Un user directory (UD) contiene información sobre los archivos que son propiedad de un usuario; cada archivo se describe por medio de una entrada en el directorio. El master directory (MD) contiene información sobre las UD de todos los usuarios registrados por el sistema. Cada entrada en el MD contiene un par (identificación de usuario, apuntador UD). Los usuarios A y B crearon, ambos, un archivo denominado alpha, estos archivos tienen entradas en los UD respectivos. La estructura del directorio que muestra la figura 3 se denomina estructura del directorio de dos niveles.

La utilización de UD proporciona libertad de nombrado. Cuando un proceso iniciado por el usuario A lleva a cabo la llamada open (alpha, …), el sistema de archivos busca el MD para dar cupo a las A del UD, y busca ahí a alpha. Si la llamada open (alpha, …) fue realizada por algún proceso ejecutado por B, entonces el sistema de archivo tendrá que buscar las B de UD para alpha. Este arreglo asegura que se accederá al archivo correcto incluso si en el sistema hay varios archivos con nombres idénticos.

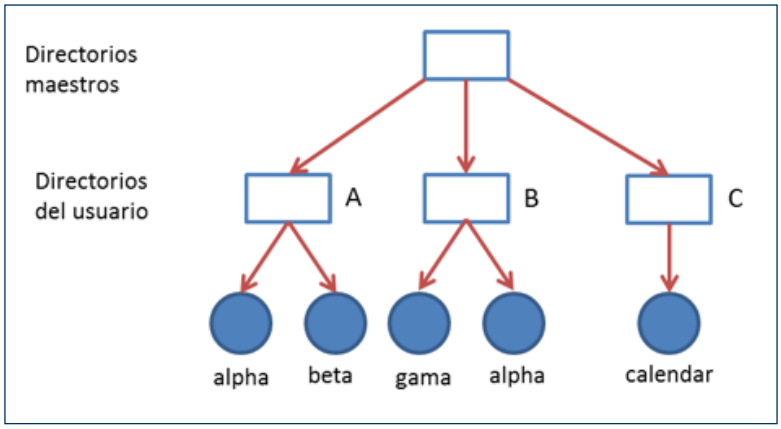


Figura 3. Directorios maestros y del usuario
Fuente: Propia.

Operaciones sobre un directorio

- Búsqueda de un archivo.
- Creación de un archivo.
- Borrado de un archivo.
- Listado de un directorio.
- Renombrado de un archivo.
- Atravesar un sistema de archivos.

Rutas de directorios

La ruta de un directorio especifica de manera única a un archivo o directorio en particular especificando su ubicación. Los nombres de ruta le muestran al usuario cómo ir de un lugar en la jerarquía de directorios a otro, este se puede asimilar como un camino de mapa.

Existes diferentes tipos de ruta según el nivel:

Directorio de un solo nivel

Llamado directorio de un solo nivel dado que todos los usuarios comparten el mismo directorio, lo que genera inconvenientes en cuanto al nombramiento y problemas de agrupamiento; para el primer caso, debido a que los usuarios pueden tener el mismo nombre para diferentes archivos o un mismo archivo puede tener varios nombres diferentes, y para el último caso, se pueden presentar diferentes tipos de archivos.

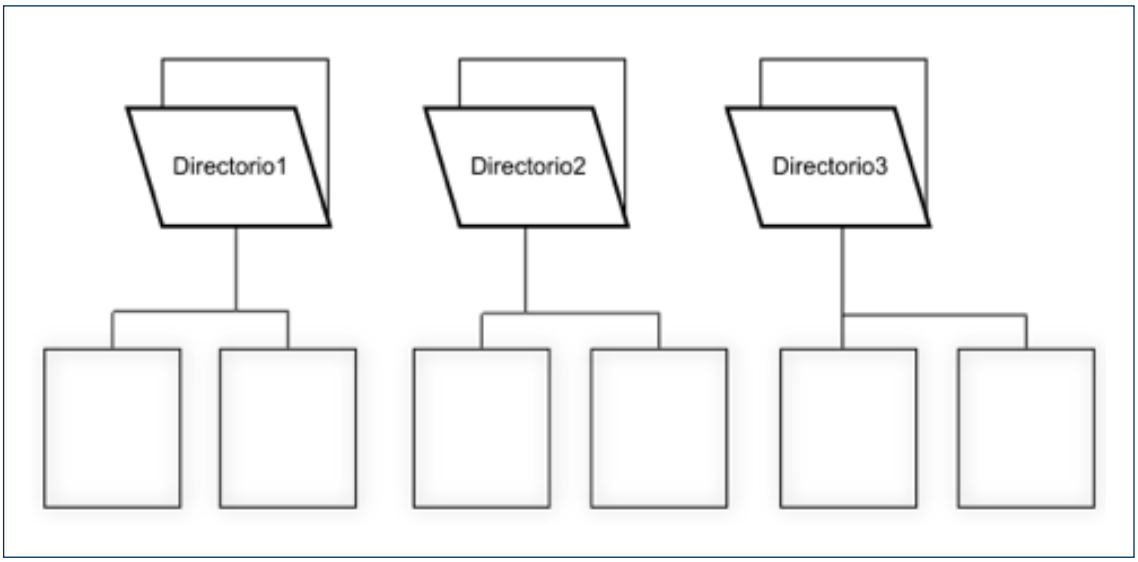


Figura 4. Directorio de un solo nivel
Fuente: Propia.

Directorios de dos niveles

Para el caso de directorios de dos niveles, cada usuario puede tener sus propios directorios, inclusive pueden tener hasta el mismo nombre de archivo para cada usuario.

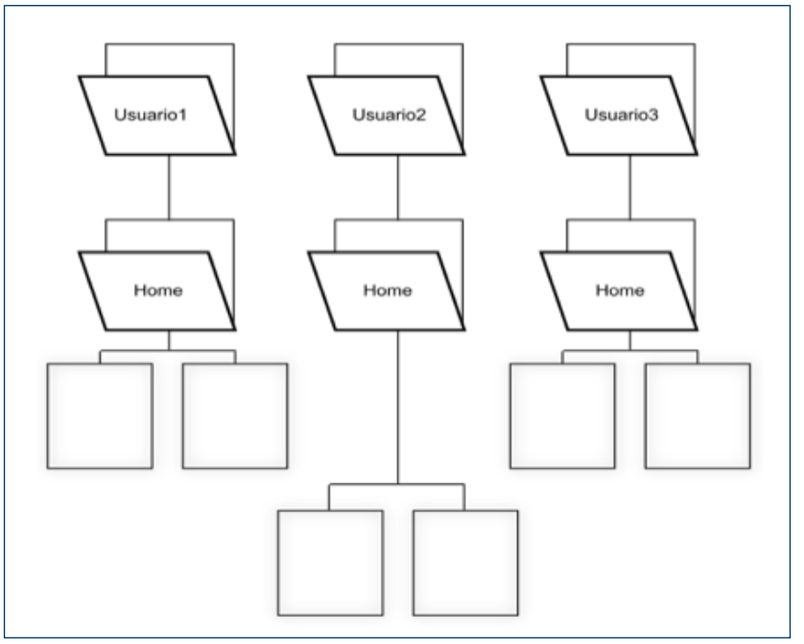


Figura 5. Directorio de un dos niveles
Fuente: Propia.

Directorio de árbol

Como su nombre lo indica, la estructura de los directorios se da de forma de árbol, y en ella cada usuario puede tener varios directorios y subdirectorios dentro de este al igual que el número de archivos que desee. Al incrementar el número de niveles, brinda la opción de agrupamiento y búsqueda eficiente.

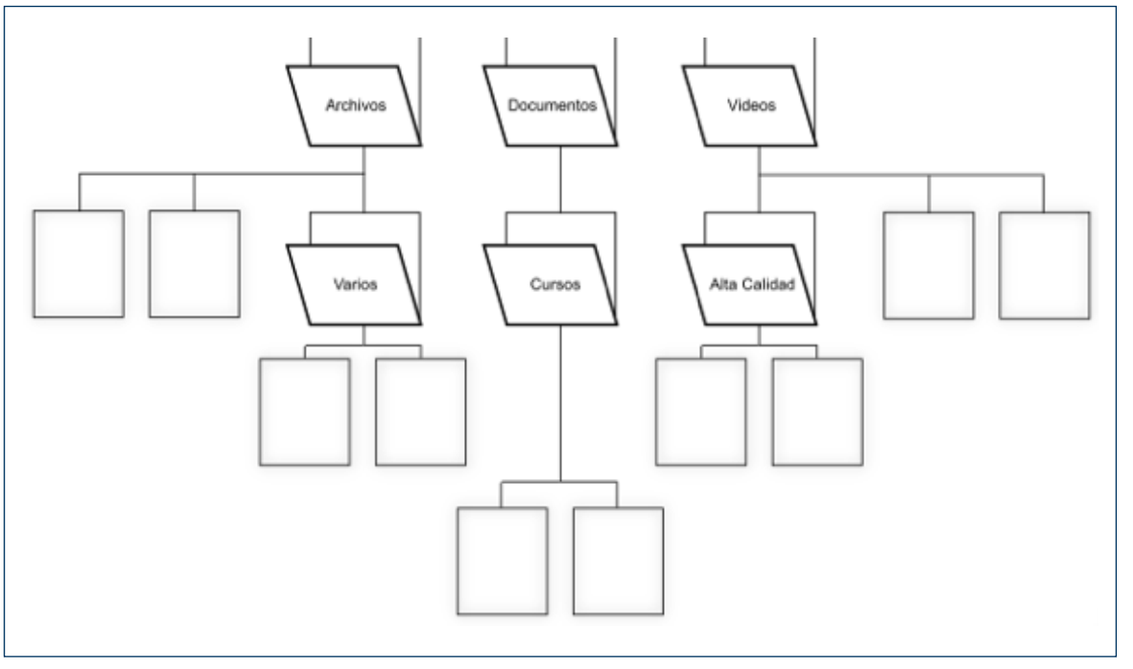


Figura 6. Directorio de árbol
Fuente: Propia.

Semánticas de consistencia

La semántica de consistencia especifica como múltiples usuarios pueden acceder a un archivo compartido simultáneamente, tienden a ser menos complejos debido a las E/S a disco la latencia de la red (sistemas de archivos remotos) Andrew File System (AFS) implementa una semántica muy compleja para compartir archivos. El sistema de archivos Unix (UFS) implementa:

- Las escrituras sobre un archivo abierto son visibles inmediatamente a los otros usuarios que comparten el mismo archivo.
- El puntero a archivos compartidos permite que múltiples usuarios lean y escriban concurrentemente AFS tiene una semántica de sesión.
- Las escrituras son solo visibles solo después que la sesión termina.

En la tabla 1 se presenta un resumen de las diferentes semánticas de consistencia.

Semántica Unix	Semántica de sesión	Semántica de versiones	Semántica inmutable
Las escrituras en un archivo son visibles inmediatamente a todos los procesos (y el nuevo puntero de L/E)	Las escrituras en un archivo no son visibles por otros procesos: al cerrar se hace visible.	Las escrituras se hacen sobre copias con número de versión: son visibles al consolidar versiones.	Si se declara compartido un archivo, no se puede modificar
Una vez abierto (*open*), la familia de procesos creado (*fork*) comparte su imagen.	Una vez cerrado el fichero, los siguientes procesos que lo abran ven las modificaciones.	Usar sincronización explícita para actualizaciones inmediatas.	Hasta no liberar el cerrojo, ni nombre ni contenido pueden modificarse.
Contención por acceso exclusivo a la imagen única del fichero.	Un fichero puede estar asociado a varias imágenes.	Tendrá varias imágenes y coste de consolidar.	No hay concurrencia.
Ext3, ufs, etc.	AFS (*Andrew File System*)	CODA	

Tabla 1. Tipos de semánticas de consistencia
Fuente: Alegre (2010).

Implementación del sistema de ficheros

La expresión "patrón de acceso a un registro" se usa de manera informal para describir el orden en que un proceso accede a los registros de un archivo. Dos patrones de acceso a un registro fundamentales son el acceso secuencial en el que se accede a los registros en el mismo orden en que existen en un archivo (o en orden inverso), y el acceso aleatorio, en el que se accede a los registros en algún orden distinto al secuencial. Un programa se ejecuta de manera eficaz si su patrón de acceso a un registro puede implementarse exitosamente en el sistema de archivos mientras se procesa un archivo.

Una organización de archivos define dos características relacionadas con un archivo: el arreglo de los registros en el archivo y el

procedimiento que se va a realizar para acceder a los registros.

Su uso proporciona un acceso eficaz a los registros para un patrón de acceso a un registro específico. La organización de archivos determina cuán eficazmente puede utilizarse el medio de E/S. Un sistema de archivos proporciona varias organizaciones de archivo, de modo que el programa puede elegir una estructura de archivos que se ajuste mejor a sus necesidades.

Mientras se analiza la actividad de procesamiento de archivos en un proceso se usa la siguiente notación:

tp: tiempo de CPU requerido para procesar la información en un registro.

tw: tiempo de espera por registro, es decir, tiempo que trascurre desde que un proceso hace la solicitud para un registro hasta aquél en el que el registro se vuelve disponible para su procesamiento.

En esta sección se describen tres organizaciones fundamentales de archivo. Otras usadas en la práctica son variaciones de estas organizaciones fundamentales o son organizaciones con un propósito especial que se ajustan a dispositivos de E/S de uso menos común. Los accesos a los archivos que utilizan organizaciones de archivos específicas se implementan por medio de un módulo IOCS denominado método de acceso. Las funciones efectuadas por los métodos de acceso se analizan después de estudiar las organizaciones fundamentales de archivo.

Organización secuencial de archivos

En la organización secuencial de archivos, el campo llave almacena los registros en una secuencia ascendente o descendente. Se espera que el patrón de acceso de una aplicación sea una subsecuencia de esta secuencia. Por consiguiente, un archivo secuencial admite dos tipos de operaciones: leer el siguiente registro y omitir el siguiente registro. En una aplicación se utiliza un archivo secuencial si sus datos pueden preclasificarse convenientemente en orden creciente o decreciente.

A la mayor parte de los dispositivos E/S puede accederse de manera secuencial, de modo que los archivos secuenciales no dependen crucialmente de las características del dispositivo. En consecuencia, un archivo secuencial puede migrase fácilmente a un tipo de dispositivo diferente. La organización secuencial de archivos también se utiliza para archivos de flujo de bytes.

Ejemplo 1: un archivo maestro de datos empleados se organiza como un archivo secuencial. El número de empleados es el campo llave del registro de un empleado (figura 7a). Cada registro en el archivo maestro contiene el número de cuenta bancaria del empleado en el que se deposita

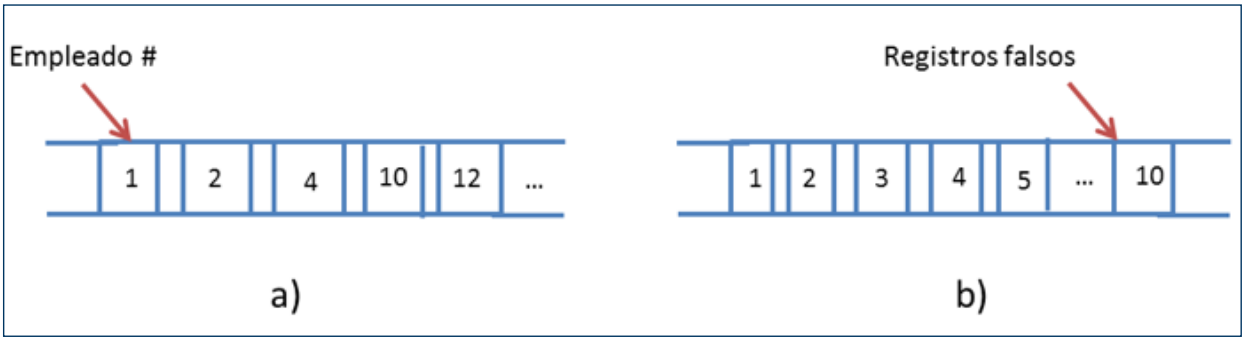


Figura 7. Registros en a) archivo secuencial b) archivo directo
Fuente: Propia.

Organización de archivos de acceso directo

Esta organización proporciona comodidad y eficiencia para el procesamiento de archivos cuando a los registros se accede en orden aleatorio. Para acceder a un registro, un comando de lectura/escritura solo se necesita mencionar su llave; por tanto, el acceso a un registro es independiente de a) cual registro se accedió antes. Esto contrasta con la organización secuencial de archivos. Si un proceso desea acceder al registro del empleado con el número, y si el último registro que se leyó del archivo fue el del empleado con el número 36, éste tendrá que omitir de manera explícita los registros que intervienen si se utiliza la organización secuencial de archivos. Estas acciones se evitan en una organización de archivo de acceso directo, lo cual la hace tanto conveniente como eficaz.

Los archivos directos se registran en disco. Cuando un proceso proporciona un valor llave, el método de acceso para la organización de archivos de acceso directo aplica una trasformación al valor llave a fin de generar una dirección (track-no, record_no). Las cabezas del disco ahora están posicionadas en la pista track_no antes de emitir un comando de lectura/escritura en record_no. Considere el archivo maestro de la información del empleado usada en el ejemplo 1, esta vez organizada como archivo directo. Considere que en una pista del disco se han escrito p registros, suponiendo que los números del empleado, la pista y los números del registro del archivo maestro empiezan desde 1, la dirección del número n de registro del empleado es (track number (tn), record number (rn) donde:

$$tn = n/p$$

$$rn = n - (tn - 1) * p$$

La organización de archivos de acceso directo proporciona eficacia de acceso cuando los registros se procesan de manera aleatoria; sin embargo, posee dos desventajas en comparación con un archivo secuencial:

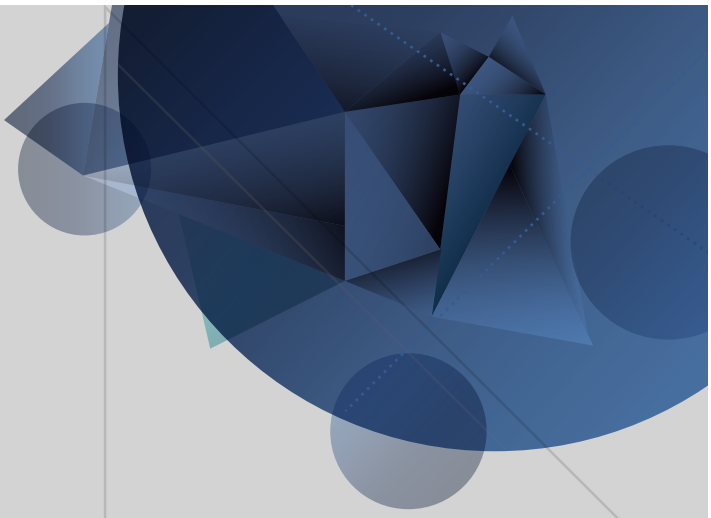
- El cálculo de la dirección del registro consume tiempo de CPU.
- Una consecuencia de la necesidad de registros falsos es una utilización deficiente del medio de E/S. Este aspecto se demuestra en el ejemplo 2.

Ejemplo 2: en la figura 7 se visualiza un arreglo de los registros de los empleados en organizaciones de archivo secuencial y directo. Los empleados con el número de empleados 3, 5 a 9 y 11 han abandonado la organización. Sin embargo, el archivo directo requiere contener un registro para cada uno de los empleados a fin de satisfacer las formulas presentadas anteriormente para el cálculo de la dirección. Este requerimiento conduce a la necesidad de contar con registros falsos en el archivo directo.

4

Unidad 4

Gestión de Entrada/Salida y seguridad en sistemas operativos



Sistemas operativos

Autor: Katherine Roa

AREANDINA
Fundación Universitaria del Área Andina

MIEMBRO DE LA RED
ILUMNO

Introducción

La interferencia en el acceso a los recursos realizada por usuarios es una amenaza grave a un sistema operativo. La naturaleza de la amenaza depende del carácter de un recurso y de la forma en que se utiliza. En este capítulo se analizan amenazas al uso de la información almacenada en archivos. Algunas técnicas que se usan para contrarrestar tales amenazas también son de utilidad para otros recursos.

En consecuencia, finalizaremos el capítulo con un análisis de diferentes tipos de ataques a la seguridad y acerca de cómo son perpetrados mediante el empleo de caballos de Troya, virus y gusanos.

U4 Metodología

Para un mayor entendimiento de este capítulo, se recomienda al estudiante realizar una lectura de la guía y generar un reporte crítico del mismo a modo personal, con el fin de afianzar los conocimientos adquiridos en esta cartilla.

Se sugiere adicionalmente, desarrollar las diferentes lecturas y actividades planteadas en la semana, con el fin de comprender mejor la temática trabajada.

U4

Desarrollo temático

Gestión de Entrada/Salida y seguridad en sistemas operativos

Introducción gestión de Entrada/Salida

Como hemos estudiado en los capítulos anteriores, la CPU es el elemento principal de una máquina, dado que esta procesa los datos e instrucciones; pero para que esto se lleve a cabo necesita de la colaboración de los dispositivos de Entrada/Salida (E/S), algunos de estos dispositivos son:

Periféricos: estos permiten la comunicación entre los usuarios y la computadora, por ejemplo: teclado, mouse (dispositivos de entrada) o impresora, pantalla (dispositivos de salida).

Dispositivos de almacenamiento: estos proporcionan almacenamiento no volátil de memoria y datos; por ejemplo, discos y disquetes (almacenamiento secundario), cintas y sistemas de archivo (almacenamiento terciario).

Dispositivos de comunicaciones: estos permiten conectar el ordenador con otros ordenadores a través de una red, por medio de tarjetas de red, módems, entre otros.

Funciones del sistema de E/S

El sistema de Entrada/Salida tiene como característica principal que se ocupa de facilitar el manejo de los dispositivos de E/S. Las funciones que realiza este son:

- Comunicación con los demás dispositivos por medio de comandos, aceptar sus interrupciones e irrumpir sus errores.
- Presentar una interfaz entre los dispositivos y el resto del sistema el cual sea fácil de usar.
- Optimizar la Entrada/Salida del sistema.
- Suministrar dispositivos virtuales que accedan a cualquier tipo de dispositivo físico.
- Reconocer la conexión de nuevos dispositivos de E/S.

Hardware de Entrada/Salida

Los dispositivos de Entrada/Salida están compuestos por un elemento electrónico, uno mecánico y la estandarización. Donde el primero, hace referencia a los controladores o unidad de E/S el cual permite transferir información entre la memoria principal y los periféricos, conectar al bus con el dispositivo y conectar los diferentes dispositivos. El segundo, hace referencia al dispositivo, el cual permite conectar a la CPU a través de los controladores (cabe aclarar que el sistema operativo trata con el controlador más no con el dispositivo) y por último, la estandarización, el cual permite utilizar el mismo controlador para distintos dispositivos así sus fabricantes sean diferentes.

Controladores de dispositivos

Conforma la interfaz del dispositivo con el bus de la máquina. La trasmisión entre la CPU y el controlador se realiza por medio de los registros del controlador. Estos registros hacen parte del espacio normal de direcciones de memoria o tienen un espacio de dirección especial. Los tipos de registros son:

- Registro de datos: estos almacena los datos de entrada o salida.
- Registro de estado: demuestra si la orden se ha ejecutado, si se han presentados errores, entre otros.
- Registro de control: le entrega al controlador las órdenes a realizar.

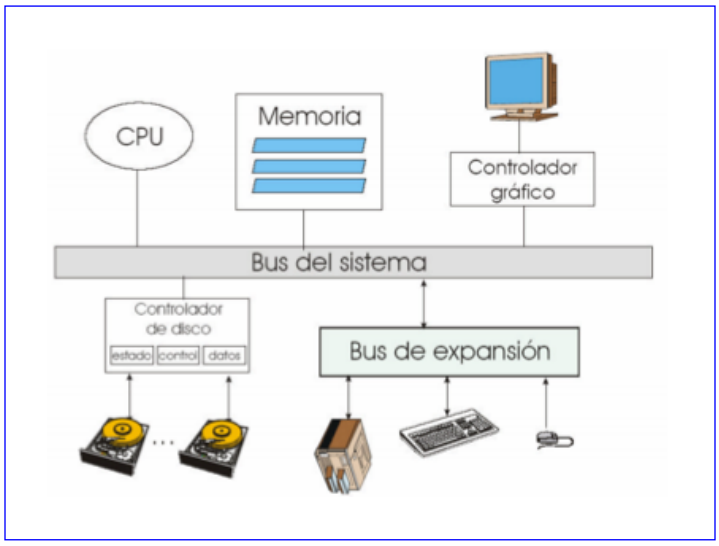


Imagen 1. Controladores de dispositivos
Fuente: Alegre (2010).

Software de Entrada/Salida

Usamos el termino de software de E/S para describir todas las acciones con respecto a la iniciación y finalización de una operación de E/S. la iniciación de E/S se ejecuta cuando ocurre una interrupción de E/S, indicando la finalización de una operación de E/S. para ver los detalles de la programación de E/S, consideremos un programa de aplicación que usa el hardware de la máquina, es decir, un sistema de cómputo que no tiene capas de software entre el programa de aplicación y el hardware de la máquina. Este programa tiene que ejecutar por su propia cuenta todas las acciones relacionadas con la iniciación y finalización de una operación E/S.

Iniciación de E/S, cuando se ejecuta una instrucción de E/S, el CPU evita la dirección del dispositivo al DMA. El DMA interroga al dispositivo para verificar su disponibilidad, este proceso se llama selección de dispositivo. El DMA informa al CPU sobre las acciones de selección de dispositivo y este coloca un correspondiente código de condición en su registro de códigos. Entones la instrucción de E/S está completa; el CPU queda libre para ejecutar otras acciones.

Procesamiento de finalización de E/S, puesto que el programa se lleva a cabo en el hardware de una máquina, y en un sistema de multiprogramación o compartición de tiempo, no puede bloquearse sino hasta que la operación de E/S finaliza. Por tanto, el CPU permanece disponible para el programa incluso durante la operación de E/S, y continua con la ejecución de instrucciones. Sin embargo, no realiza ningún trabajo hasta que finaliza la operación de E/S.

El sistema operativo organiza el software de E/S en tres niveles, con una función e interfaz definida:

- Manejadores de interrupciones.
- Software de E/S independiente del dispositivo.
- Software de las aplicaciones de usuario.

Manejadores de interrupción, son los encargados de tratar las interrupciones generadas por los controladores, una vez se presentada una interrupción es necesario: impedir la ejecución de un proceso, guardar los registros, identificar la fuente de la interrupción y comunicar el evento al manejador del dispositivo, restaurar la ejecución de un proceso (que no tiene por qué ser el interrumpido).

Software independiente del dispositivo, tiene como funciones asignar nombres a los dispositivos de entrada/salida, impedir el acceso a usuarios sin autorización, y gestionar almacenamiento temporal en memoria para evitar el acceso directo a los dispositivos.

Software a nivel usuario, es el medio para realizar las llamadas al sistema operativo, igualmente controla el acceso a dispositivos compartidos y dedicados.

Ejemplos de dispositivos

En la tabla 1 se presenta la clasificación de los dispositivos de E/S de acuerdo al objetivo y el modo de acceso. Una unidad de un medio de E/S se llama volumen de E/S. un cartucho de cinta y un disco puede llamarse entonces un volumen de cinta y un volumen de disco, respectivamente.

Objetivo	Modo de acceso	Ejemplos
Entrada	Secuencial	Teclado, mouse, conexiones de red, cintas, discos.
	Aleatorio	Discos.
Impresión	Secuencial	Impresoras, conexiones de red, cintas, discos.
	Aleatorio	Discos.
Almacenamiento	Secuencial	Cintas, discos.
	Aleatorio	Discos.

Tabla 1. Clasificación de los dispositivos de E/S
Fuente: Propia.

A continuación conoceremos un poco más sobre alguno de estos,

Cintas y cartuchos magnéticos

El medio de E/S en una cinta o cartucho es una tira de material magnético en el cual se escriben informaciones en la forma de unos y ceros, usando principios de la grabación electromagnética. La grabación en cinta es de pista multiplex, una cabeza de lectura/escritura se posiciona en cada pista. Esta cabeza graba o lee informaciones de los bits de un byte, así como también algunas informaciones adicionales que se usan para la detección y corrección de errores.

Las unidades de cintas son dispositivos de acceso secuenciales, las operaciones que se pueden ejecutar en estos dispositivos son las siguientes: read/write, skip y rewind.

Discos

Un disco es un objeto circular plano llamado plato, que gira alrededor de su propio eje. Las superficies circulares de un plato están cubiertas con material magnético, se usa una sola cabeza de lectura/escritura para realizar la grabación en una superficie; por tanto, un byte es grabado en serie a lo largo de una pista. La cabeza, la información grabada forma una pista circular sobre la superficie del disco. En un disco no se usa la información de paridad; una suma verificadora de redundancia cíclica (CRC) se escribe junto con cada registro para admitir la detección de errores.

Seguridad sobre archivos

Las medidas de seguridad y protección evitan interferencias con el uso de recurso lógicos físicos en un sistema. Cuando estas medidas se aplican a información, aseguran que ésta sea utilizada no solo por usuarios autorizados y de la manera desea, igualmente que no se destruya o se dañe. La seguridad tiene que ver con amenazas a la información externa a un sistema de cómputo, mientras que la protección está relacionada con amenazas internas.

La seguridad se implementa mediante la autenticación, utilización de contraseñas, las cuales frustran intentos realizados por entidades externas para enmascararse como usuarios del sistema. Para asegurar la confidencialidad de las contraseñas se utiliza la técnica de cifrado.

Un usuario necesita compartir con colaboradores datos y programas almacenados en archivos. Un privilegio de acceso es una especificación de la manera en que un usuario puede acceder al sistema de archivo. El propietario de un archivo suministra al SO información concerniente a los privilegios de acceso a un archivo. La función de protección de un sistema operativo almacena esta información y la utiliza para asegurar que todos los accesos al archivo coincidan estrictamente con los privilegios de acceso.

Los sistemas operativos utilizan dos conjuntos de técnicas para contrarrestar las amenazas al uso de la información.

- La seguridad implica la protección de los datos y programa de un usuario en contra de interferencia por entidades o personas externas a un sistema; por ejemplo, no usuarios.
- La protección implica proteger los datos y programas de un usuario en contra de interferencia por otros usuarios del sistema.

Existen dos métodos clave usados por los sistemas operativos para implementar seguridad y protección.

La autentificación es el método que se usa para comprobar la identidad de un apersona. La comprobación física de la identidad no es factible en entornos contemporáneos de cómputo, de modo que la autenticación basada en computadoras se fundamenta en un conjunto de suposiciones. Una suposición común es que tal persona es el usuario que reclama serlo si sabe algo que solo que solo el usuario debería saberlo. Esto se denomina autenticación por conocimiento.

El otro método utilizado se llama autorización, el cual consiste en suponer que un persona es un usuario legitimo si posee algo que se supone solo puede estar en manos del usuario, ejemplo, de estos métodos son la comprobación con base en las contraseñas y la autenticación biométrica, es decir, la autenticación basada en ciertas características biológicas únicas e inalterables de un usuario, como sus huellas digitales, su retina o iris. La autorización es el acto que consiste en determinar los privilegios de un usuario. Estos se utilizan para implementar protección.

En casi todos los sistemas operativos la validación de información para un usuario consiste en una forma cifrada de su contraseña. El símbolo de autenticación para un usuario es la identificación de un usuario asignada por el sistema operativo. El sistema operativo recuerda la identificación de un usuario hasta que el usuario hasta que el usuario se sale del sistema. Utiliza esta información siempre que un usuario o un proceso hace una solicitud de un recurso o un servicio. A un usuario autenticado se le permite usar todos los servicios del sistema, excepto porque puede usar un archivo solo si el propietario del archivo lo ha autorizado explícitamente.

La diferencia entre protección y seguridad constituye una clara división de asuntos para el sistema operativo. Un sistema operativo convencional, la cuestión de la seguridad está limitada a garantizar que solo los usuarios registrados pueden utilizar el sistema operativo. Cuando una persona entra, se lleva a cabo una comprobación de seguridad para determinar si la persona es un usuario del sistema operativo y, en caso de serlo, obtiene su identificación de usuario. Luego, de esta comprobación, todas las amenazas a la información almacenada en el sistema constituye asuntos relacionados con la protección; el usuario del SO utiliza la identificación de una persona para decidir si puede acceder a un archivo especifico.

Políticas y mecanismos de seguridad y protección

Seguridad

- Política: si una persona puede volverse usuario del sistema. El administrador del sistema aplica la política mientras registra nuevos usuarios.
- Mecanismos: agregar o quitar usuario, comprobar si una persona es un usuario registrado por medio de una autenticación. Realiza cifrado para asegurar confidencialidad de los datos.

Protección

Política: el propietario del archivo especifica la política de autorización para un archivo. Decide que usuario puede acceder a un archivo y de qué manera.

Mecanismos: establece o cambia información de autorización para un archivo, com-

prueba si una solicitud de procesamiento de un archivo se ajusta a los privilegios del usuario.

Seguridad de acceso

Los intentos de una persona o entidad por violar la seguridad de un sistema se llama ataque a la seguridad, y tal persona o entidad se denomina intruso o adversario. La comunicación es un componente vulnerable de un entorno de cómputo, de modo que muchos ataques se lanzan a través de la componente de comunicación. Dos ataques a la seguridad evidentes son la intercepción de la comunicación y la manipulación. Estos ataques ocurren principalmente en sistemas operativos distribuidos.

Clases de ataques a la seguridad

Enmascaramiento: un intruso es capaz de hacerse pasar por un usuario registrado del sistema, o lo que es peor, puede corromper o destruir información pertinente al usuario. La forma evidente de lanzar este tipo de ataque consiste en romper la contraseña de un usuario y usar este conocimiento para pasar la prueba de autenticación en el momento de ingresar al sistema. Otro método consiste en un enmascaramiento más sutil a través de programas que se importan desde un entorno de software.

Navegación del servicio: también conocido como ataque DoS, este es lanzado mediante el aprovechamiento de alguna vulnerabilidad en el diseño en la operación del sistema operativo, algún aspecto de la operación de un sistema operativo es alterado de modo que no puede admitir algunos servicios como era de esperar. Un ataque DoS puede lanzarse mediante la corrupción de un programa que ofrece algún servicio o de la destrucción de la información de configuración dentro de un kernel; por ejemplo, el uso de un dispositivo de E/S puede negarse cambiando su entrada a la tabla de dispositivos del kernel.

Caballo de Troya, virus y gusanos

Los caballos de Troya, los virus y gusanos constituyen formas novedosas para causar estragos en un sistema de cómputo al colocar programas o códigos capaces de perpetrar una gama de ataques a la seguridad. Algunas de sus características son:

Caballo de Troya, programa que lleva a cabo una función legítima conocida por un sistema operativo o sus usuarios, pero que también posee una parte oculta que puede utilizarse para propósitos infames, como ataques a la seguridad del mensaje o suplantación. Por ejemplo, es capaz de borrar un disco duro en la computadora, lo cual constituye una violación al requerimiento de seguridad, o hacer que un sistema se bloquee o se vuelva más lento, lo cual supone negación del servicio. También puede monitorear el tráfico entre el usuario y otros procesos a fin de reunir información para efectos de enmascaramiento, o iniciar conversaciones espurias con la misma intención. Un ejemplo típico es un programa spoof login que proporciona un mensaje con identificación falda para engañar a un programa a fin de que revele información sobre alguna contraseña. Debido a que un caballo de Troya es cargado de manera explícita por un usuario, n o es difícil seguir la pista de su auto u origen.

Virus, pieza de un código que por sí mismo es capaz de atacar a otros programas en el sistema y dispersarse a otros sistemas cuando se copian o trasfieren los programas. Mientras se adjunta a otro programa, un vi-

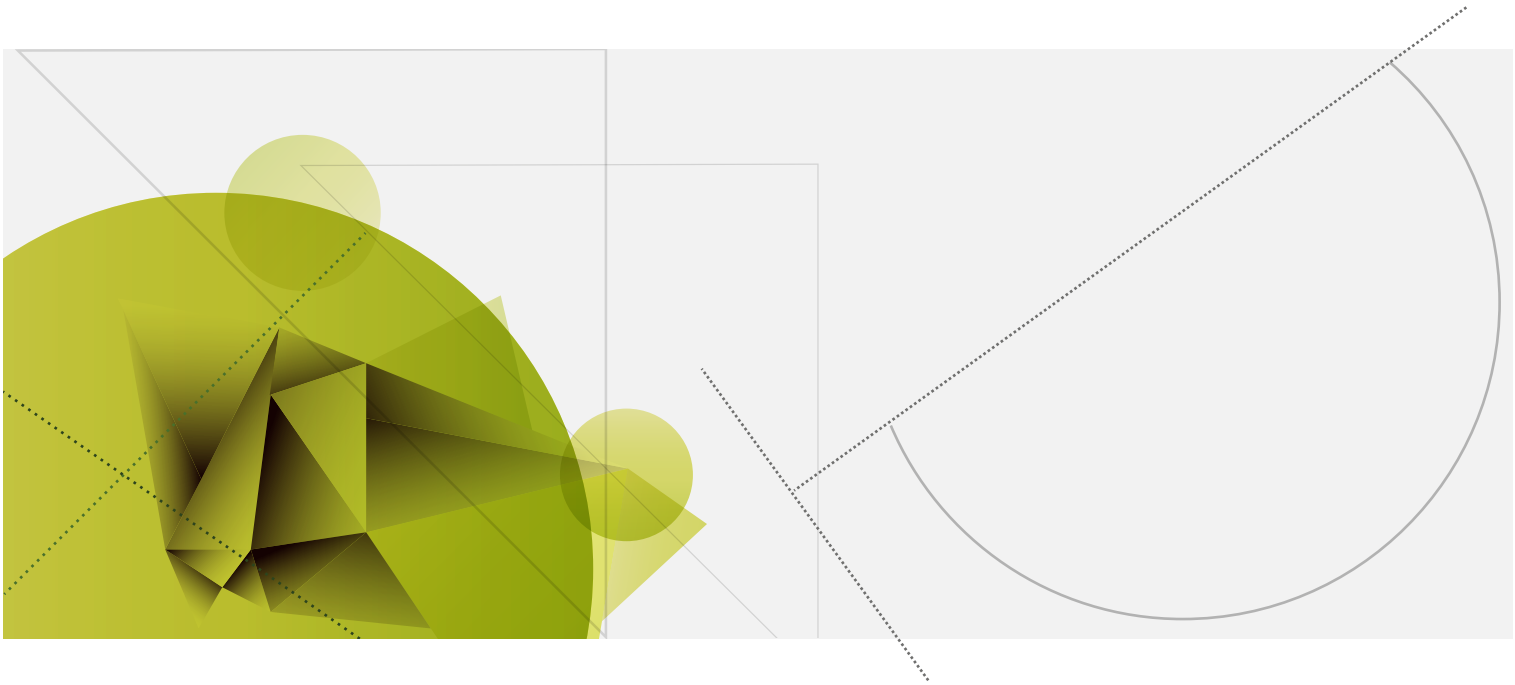
rus escribe su propia dirección como la dirección inicial de ejecución el programa. De esta forma adquiere el control del sistema, al adjuntarse a tales programas. Después de eso, pasa el control al programa genuino para su ejecución. El paso de infección no consume mucho tiempo de CPU, de modo que un usuario no puede saber que el programa ejecutado trasporta un virus. La forma en que un virus se introduce él solo a otro programa mucho más difícil de rastrear que en el caso de un caballo de Troya. Aparte de infectar a otros programas, un virus comúnmente permanece latente hasta que algún evento, como una fecha o un instante especifico, lo que despiertan. Entonces lanza uno de los ataques antes mencionados.

Gusano, programa que se difunde a otros sistemas de cómputo mediante la exploración de huecos de seguridad, como debilidades en las instalaciones para creación de procesos remotos. Se sabe que los gusanos se duplican a ritmos inimaginablemente altos, recargando la red, y consumiendo tiempo de CPU durante la duplicación. Debido a su capacidad de autoduplicación, resulta mucho más difícil de rastrear a un gusano que a un virus.

Bibliografia

- Alegre, M. (2010). Sistemas Operativos Monopuesto. Madrid, España: ediciones Paraninfo.
- Candela et al. (2007). Fundamentos de los Sistemas Operativos. Madrid, España: Thomsom.
- Carretero, J., Miguel, de P., García, F. & Pérez, F. (2001). Sistemas Operativos. Una visión aplicada. Madrid, España: Ed. Mc Graw Hill.
- Martínez, P. & Díaz, J. (1996). Sistemas Operativos: Teoría y práctica. Ediciones Díaz de Santos.
- Stallings, W. (2005). Sistemas Operativos, 5ª edición. México: Ed. Prentice Hall.
- anenbaum, A. (2003). Sistemas Operativos Modernos. México: Pearson Education.

Esta obra se terminó de editar en el mes de noviembre
Tipografá Myriad Pro 12 puntos
Bogotá D.C,-Colombia.



AREANDINA
Fundación Universitaria del Área Andina

MIEMBRO DE LA RED
ILUMNO