UNIVERSIDAD TECNOLÓGICA DE PANAMÁ

FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES

SISTEMAS OPERATIVOS

Dr. Vladimir Villarreal

2017

CONTENIDO

I. INTRODUCCIÓN A LOS SISTEMAS OPERATIVOS

II. GESTIÓN DE PROCESOS Y PROCESADOR

III. GESTIÓN DE MEMORIA

CAPITULO I. INTRODUCCIÓN A LOS SISTEMAS OPERATIVOS

Objetivos:

- Proporcionar un panorama general de la evolución conceptual de los sistemas operativos.
- Describir los componentes, estructuras y diseño de un SO.
- Describir las formas en que se llevan a cabo el arranque y activación de un SO.

¿De qué trata esta sesión de aprendizaje?

En esta sesión de aprendizaje se presentan las características generales de los Sistemas de Operativos (SO). De identifican los elementos básicos que componen los SO. Se plantean lo enfoques de los diferentes SO. Además, se da a conocer su modo de activación y continuación de tareas.

1. Introducción

Las computadoras están equipadas con una capa de software llamada **sistema operativo**, cuyo trabajo es proporcionar a los programas de usuario un modelo de computadora mejor, más simple y pulcro, así como encargarse de la administración de todos los recursos tales como: procesadores, una memoria principal, discos, impresoras, un teclado, un ratón, una pantalla o monitor, interfaces de red y otros dispositivos de entrada/salida.

Por encima del hardware se encuentra el software. La mayoría de las computadoras tienen dos modos de operación: modo kernel y modo usuario. El sistema operativo es la pieza fundamental del software y se ejecuta en **modo kernel** (también conocido como **modo supervisor**). En este modo, el sistema operativo tiene acceso completo a todo el hardware y puede ejecutar cualquier instrucción que la máquina sea capaz de ejecutar. El resto del software se ejecuta en **modo usuario**, en el cual sólo un subconjunto de las instrucciones de máquina es permitido.

El programa de interfaz de usuario, shell o GUI, es el nivel más bajo del software en modo usuario y permite la ejecución de otros programas, como un navegador Web, lector de correo

electrónico o reproductor de música. Estos programas también utilizan en forma intensiva el sistema operativo.

1.1. Definición

Sistema operativo

Un sistema operativo es un programa que controla la ejecución de los programas de aplicación y que actúa como interfaz entre el usuario de un computador y el hardware de la misma.

Otras definiciones:

- Es una colección de programas que comparten los mismos mecanismos de distribución. Se genera con el propósito de administrar y extender los recursos o capacidades de los sistemas de información. **Daniel Sol**.
- El sistema operativo es el software que controla la operación general de una computadora, proporciona los medios por los que un usuario puede almacenar y recuperar archivos, provee la interfaz por la que un usuario puede solicitar la ejecución de programas y provee el ambiente necesario para que los programas solicitados se ejecuten. **J. Glenn Brookshear. Computer Science, an Overview, Pearson/Addison-Wesley, 9ª ed.2007**.
- Un sistema operativo es un programa que maneja el hardware de la computadora. También provee la base para los programas de aplicación y actúa como intermediario entre el usuario de la computadora y el hardware de esta. **Abraham Silberschatz et al. Operating System Concepts, Jhon Wiley & Sons, Inc., 7ª ed.**

  Algunos objetivos principales de los sistemas operativos son:

  • **Comodidad*:*** un sistema operativo hace que un computador sea más cómodo de utilizar.

  • **Eficiencia*:*** un sistema operativo permite que los recursos de un sistema informático se aprovechen de una manera más eficiente.

  • **Capacidad de evolución***:* un sistema operativo debe construirse de modo que permita el desarrollo efectivo, la verificación y la introducción de nuevas funciones en el sistema y, a la vez, no interferir en los servicios que brinda.

1.2 Evolución de los sistemas operativos

Para intentar comprender los requisitos básicos de un sistema operativo y el significado de las características principales de un sistema operativo contemporáneo, resulta útil considerar cómo han evolucionado los sistemas operativos a lo largo de los años.

- **Proceso en serie**

En los primeros computadores, de finales de los 40 hasta mediados de los 50, el programador interactuaba directamente con el hardware; no había sistema operativo. La operación con estas máquinas se efectuaba desde una consola consistente en unos indicadores luminosos, unos conmutadores, algún tipo de dispositivo de entrada y una impresora. Los programas en código máquina se cargaban a través del dispositivo de entrada (un lector de tarjetas, por ejemplo). Si se detiene el programa por un error, la condición de error se indicaba mediante los indicadores luminosos. El programador podía examinar los registros y la memoria principal para determinar la causa del error. Si el programa continuaba hasta su culminación normal, la salida aparecería en la impresora. Estos primeros sistemas presentaban dos problemas principales:

• **Planificación:** La mayoría de las instalaciones empleaban un formulario de reserva de tiempo de máquina. Normalmente, un usuario podía reservar bloques de tiempo en múltiplos de media hora o algo por el estilo. Un usuario podía reservar una hora y terminar a los 45 minutos; esto daba como resultado un desperdicio del tiempo del computador. Por el contrario, el usuario podía tener dificultades, no terminar en el tiempo asignado y verse forzado a parar sin haber solucionado el problema.

• **Tiempo de preparación:** Un programa sencillo, llamado trabajo, cargaba un compilador y un programa en lenguaje de alto nivel (programa fuente) en la memoria, salvaba el programa compilado (programa objeto) y luego montaba y cargaba el programa objeto junto con las funciones comunes. Cada uno de estos pasos podía implicar montar y desmontar cintas o preparar paquetes de tarjetas. Si se producía un error, el infortunado usuario tenía que volver al inicio de este proceso de preparación. De este modo, se perdía un tiempo considerable en

preparar un programa para su ejecución. Este modo de operación podría denominarse proceso en serie porque refleja el hecho de que los usuarios tenían que acceder al computador en serie. Con el paso del tiempo se desarrollaron varias herramientas de software de sistemas para intentar hacer más eficiente este proceso en serie. Entre éstas se incluían bibliotecas de funciones comunes, montadores, cargadores, depuradores y rutinas de manejo de E/S que estaban disponibles como un software común para todos los usuarios.

- **Proceso por lotes**

Las primeras máquinas eran muy caras y, por tanto, era importante maximizar la utilización de las mismas. El tiempo desperdiciado por la planificación y la preparación era inaceptable. Para mejorar el uso, se desarrolló el concepto de sistema operativo por lotes (batch). El primer sistema operativo por lotes fue desarrollado a mediados de los 50 por la General Motors para usar en un IBM 701. Este concepto fue refinado posteriormente e implementado en un IBM 704 por una serie de clientes de IBM. A principios de los 60, un conjunto de constructores ya habían desarrollado sistemas operativos por lotes para sus computadores. IBSYS, el sistema operativo de IBM para los computadores 7090/7094, es particularmente notable por su amplia influencia en otros sistemas. La idea central que está detrás del esquema sencillo de proceso por lotes es el uso de un elemento de software conocido como monitor. Con el uso de esta clase de sistema operativo, los usuarios ya no tenían acceso directo a la máquina, en su lugar, el usuario debía entregar los trabajos en tarjetas o en cinta al operador del computador, quien agrupaba secuencialmente los trabajos por lotes y ubicaba los lotes enteros en un dispositivo de entrada para su empleo por parte del monitor. Cada programa se construía de modo tal que volviera al monitor al terminar su procesamiento y, en ese momento, el monitor comenzaba a cargar automáticamente el siguiente programa. Para entender cómo funciona este esquema, se va a ver desde dos puntos de vista: el del monitor y el del procesador. Desde el punto de vista del monitor, él es quien controla la secuencia de sucesos. Para que esto sea posible, gran parte del monitor debe estar siempre en memoria principal y disponible para su ejecución. Esta parte del monitor se conoce como monitor residente. El resto del monitor consta de utilidades y funciones comunes que se cargan como subrutinas en los programas de los usuarios al comienzo de cualquier trabajo que las necesite. El monitor lee los trabajos uno a uno del dispositivo de entrada (normalmente, un lector de

tarjetas o una unidad de cinta magnética). A medida que lo lee, el trabajo actual se ubica en la zona del programa de usuario y el control pasa al trabajo. Cuando el trabajo termina, se devuelve el control al monitor, quien lee inmediatamente un nuevo trabajo. Los resultados de cada trabajo se imprimen y entregan al usuario. Considérese ahora esta secuencia desde el punto de vista del procesador. En un cierto momento, el procesador estará ejecutando instrucciones de la zona de memoria principal que contiene al monitor. Estas instrucciones hacen que el trabajo siguiente sea leído en otra zona de la memoria principal. Una vez que el trabajo se ha leído, el procesador encuentra en el monitor una instrucción de desvío que ordena al procesador continuar la ejecución en el inicio del programa de usuario. El procesador ejecuta entonces las instrucciones del programa de usuario hasta que encuentre una condición de finalización o de error. Cualquier de estos dos sucesos provocan que el procesador vaya a por la instrucción siguiente del programa monitor. De este modo, la frase "el control se le pasa al trabajo" quiere decir simplemente que el procesador pasa a leer y ejecutar instrucciones del programa de usuario, mientras que la frase "el control vuelve al monitor" quiere decir que el procesador pasa ahora a leer y ejecutar las instrucciones del programa monitor. Debe quedar claro que es el monitor el que gestiona el problema de la planificación. Se pone en cola un lote de trabajos y éstos son ejecutados tan rápido como es posible, sin que haya tiempo alguno de desocupación. ¿Qué ocurre con la preparación de los trabajos? El monitor también se encarga de esto. Con cada trabajo, se incluyen instrucciones de una forma primitiva de lenguaje de control de trabajos (JCL, Job Control Lenguaje), que es un tipo especial de lenguaje de programación empleado para dar instrucciones al monitor.

1.2.3 Sistemas multiprogramados

La programación multitareas o los sistemas multiprogramados buscaban maximizar el tiempo de uso efectivo del procesador ejecutando varios procesos al mismo tiempo. El hardware requerido cambió fuertemente. Si bien se esperaba que cada usuario fuera responsable con el uso de recursos, resultó necesario que apareciera la infraestructura de protección de recursos: un proceso no debe sobreescribir el espacio de memoria de otro (ni el código, ni los datos), mucho menos el espacio del monitor. Esta protección se encuentra en la Unidad de Manejo de Memoria (MMU), presente en todas las computadoras de uso genérico desde los años noventa. Ciertos dispositivos requieren bloqueo para ofrecer acceso exclusivo/único: cintas e impresoras, por ejemplo, son de

acceso estrictamente secuencial, y si dos usuarios intentaran usarlas al mismo tiempo, el resultado para ambos se corrompería. Para estos dispositivos, el sistema debe implementar otros mecanismos de bloqueo.

1.2.4 Sistemas de tiempo compartido

El modo de interactuar con las computadoras se modificó drásticamente durante los años sesenta, al extenderse la multitarea para convertirse en sistemas interactivos y multiusuarios, en buena medida diferenciados de los anteriores por la aparición de las terminales (primero teletipos seriales, posteriormente equipos con una pantalla completa como se conocen hasta hoy). En primer término, la tarea de programación y depuración del código se simplificó fuertemente al poder hacer en el programador directamente cambios y someter el programa a la ejecución inmediata. En segundo término, la computadora nunca más estaría simplemente esperando a que esté listo un progama: mientras un programador editaba o compilaba su programa, la computadora seguía calculando lo que otros procesos requirieran. Un cambio fundamental entre el modelo de multiprogramación y de tiempo compartido es el tipo de control sobre la multitarea.

Multitarea cooperativa o no apropiativa. La implementaron los sistemas multiprogramados: cada proceso tenía control del CPU hasta que éste hacía una llamada al sistema (o indicara su disposición a cooperar por medio de la llamada yield: ceder el paso). Un cálculo largo no era interrumpido por el sistema operativo, en consecuencia, un error de programador podía congelar la computadora completa.

Multitarea preventiva o apropiativa. En los sistemas de tiempo compartido, el reloj del sistema interrumpe periódicamente a los diversos procesos, transfiriendo forzosamente el control nuevamente al sistema operativo. Éste puede entonces elegir otro proceso para continuarla ejecución. Además, fueron naciendo de forma natural y paulatina las abstracciones que se conocen hoy en día, como los conceptos de archivos y directorios, y el código necesario para emplearlos iba siendo enviado a las bibliotecas de sistema y, cada vez más (por su centralidad) hacia el núcleo mismo del, ahora sí, sistema operativo.

Un cambio importante entre los sistemas multiprogramados y de tiempo compartido es que la velocidad del cambio entre una tarea y otra es mucho más rápido: si bien en un sistema

multiprogramado un cambio de contexto podía producirse sólo cuando la tarea cambiaba de un modo de ejecución a otro, en un sistema interactivo, para dar la ilusión de uso exclusivo de la computadora, el hardware emitía periódicamente al sistema operativo interrupciones (señales) que le indicaban que cambie el proceso activo (como ahora se le denomina a una instancia de un programa en ejecución). Diferentes tipos de proceso pueden tener distinto nivel de importancia ya sea porque son más relevantes para el funcionamiento de la computadora misma (procesos de sistema), porque tienen mayor carga de interactividad (por la experiencia del usuario) o por diversas categorías de usuarios (sistemas con contabilidad por tipo de atención). Esto requiere la implementación de diversas prioridades para cada uno de éstos

1.3 Tipos de sistemas operativos

- **Sistemas operativos de mainframe**

En el extremo superior están los sistemas operativos para las mainframes, las computadoras del tamaño de un cuarto completo que aún se encuentran en los principales centros de datos corporativos. La diferencia entre estas computadoras y las personales está en su capacidad de E/S. Una mainframe con 1000 discos y millones de gigabytes de datos no es poco común; una computadora personal con estas especificaciones sería la envidia de los amigos del propietario. Las mainframes también están volviendo a figurar en el ámbito computacional como servidores Web de alto rendimiento, servidores para sitios de comercio electrónico a gran escala y servidores para transacciones de negocio a negocio. Los sistemas operativos para las mainframes están profundamente orientados hacia el procesamiento de muchos trabajos a la vez, de los cuales la mayor parte requiere muchas operaciones de E/S. Por lo general ofrecen tres tipos de servicios: procesamiento por lotes, procesamiento de transacciones y tiempo compartido. Un sistema de procesamiento por lotes procesa los trabajos de rutina sin que haya un usuario interactivo presente. El procesamiento de reclamaciones en una compañía de seguros o el reporte de ventas para una cadena de tiendas son actividades que se realizan comúnmente en modo de procesamiento por lotes. Los sistemas de procesamiento de transacciones manejan grandes cantidades de pequeñas peticiones, por ejemplo: el procesamiento de cheques en un banco o las reservaciones en una aerolínea. Cada unidad de trabajo es pequeña, pero el sistema debe manejar

cientos o miles por segundo. Los sistemas de tiempo compartido permiten que varios usuarios remotos ejecuten trabajos en la computadora al mismo tiempo, como consultar una gran base de datos. Estas funciones están íntimamente relacionadas; a menudo los sistemas operativos de las mainframes las realizan todas. Un ejemplo de sistema operativo de mainframe es el OS/390, un descendiente del OS/360. Sin embargo, los sistemas operativos de mainframes están siendo reemplazados gradualmente por variantes de UNIX, como Linux.

- **Sistemas operativos de servidores**

En el siguiente nivel hacia abajo se encuentran los sistemas operativos de servidores. Se ejecutan en servidores, que son computadoras personales muy grandes, estaciones de trabajo o incluso mainframes. Dan servicio a varios usuarios a la vez a través de una red y les permiten compartir los recursos de hardware y de software. Los servidores pueden proporcionar servicio de impresión, de archivos o Web. Los proveedores de Internet operan muchos equipos servidores para dar soporte a sus clientes y los sitios Web utilizan servidores para almacenar las páginas Web y hacerse cargo de las peticiones entrantes. Algunos sistemas operativos de servidores comunes son Solaris, FreeBSD, Linux y Windows Server 200x.

- **Sistemas operativos de multiprocesadores**

Una manera cada vez más común de obtener poder de cómputo de las grandes ligas es conectar varias CPU en un solo sistema. Dependiendo de la exactitud con la que se conecten y de lo que se comparta, estos sistemas se conocen como computadoras en paralelo, multicomputadoras o multiprocesadores. Necesitan sistemas operativos especiales, pero a menudo son variaciones de los sistemas operativos de servidores con características especiales para la comunicación, conectividad y consistencia.

Con la reciente llegada de los chips multinúcleo para las computadoras personales, hasta los sistemas operativos de equipos de escritorio y portátiles convencionales están empezando a lidiar con multiprocesadores de al menos pequeña escala y es probable que el número de núcleos aumente con el tiempo. Por fortuna, se conoce mucho acerca de los sistemas operativos de multiprocesadores gracias a los años de investigación previa, por lo que el uso de este conocimiento en los sistemas multinúcleo no debe presentar dificultades. La parte difícil será

hacer que las aplicaciones hagan uso de todo este poder de cómputo. Muchos sistemas operativos populares (incluyendo Windows y Linux) se ejecutan en multiprocesadores.

1.4 Estructura de un sistema operativo

1.4.1 Monolítico

Es la estructura de los primeros sistemas operativos constituidos fundamentalmente por un sólo programa compuesto de un conjunto de rutinas entrelazadas de tal forma que cada una puede llamar a cualquier otra. Las características fundamentales de este tipo de estructura son:

- Construcción del programa final a base de módulos compilados separadamente que se unen a través del encadenador (linker).
- Buena definición de parámetros de enlace entre las distintas rutinas existentes, lo que puede provocar mucho acoplamiento.
- Carecen de protecciones y privilegios al entrar a rutinas que manejan diferentes aspectos de los recursos del computador, como memoria, disco, etc.
- Generalmente están hechos a medida, por lo que son eficientes y rápidos en su ejecución y gestión, pero por lo mismo carecen de flexibilidad para soportar diferentes ambientes de trabajo o tipos de aplicaciones

1.4.2 Capas o niveles

A medida que fueron creciendo las necesidades de los usuarios y se perfeccionaron los sistemas, se hizo necesaria una mayor organización del software, del sistema operativo, donde una parte del sistema contenía sub partes y esto organizado en forma de niveles.

Se dividió el sistema operativo en pequeñas partes, de tal forma que cada una de ellas estuviera perfectamente definida y con un claro interface con el resto de los elementos; una estructura jerárquica o de niveles en los sistemas operativos. Otra forma de ver este tipo de sistema es la denominada de anillos concéntricos o "rings". En el sistema de anillos, cada uno tiene una apertura, conocida como puerta o trampa (trap), por donde pueden entrar las llamadas de las capas inferiores. De esta forma, las zonas más internas del sistema operativo o núcleo del sistema estarán más protegidas de accesos indeseados desde las capas más externas. Las capas más internas serán, por tanto, más privilegiadas que las externas.

1.4.3 Máquina Virtual

Se trata de un tipo de sistemas operativos que presentan una interface a cada proceso, mostrando una máquina que parece idéntica a la máquina real subyacente.

Estos sistemas operativos separan dos conceptos que suelen estar unidos en el resto de los sistemas:

1. La multiprogramación y

2. La máquina extendida.

El objetivo de los sistemas operativos de máquina virtual es el de integrar distintos sistemas operativos dando la sensación de ser varias máquinas diferentes. El núcleo de estos sistemas operativos se denomina monitor virtual y tiene como misión llevar a cabo la multiprogramación, presentando a los niveles superiores tantas máquinas virtuales como se soliciten. Estas máquinas virtuales no son máquinas extendidas, sino una réplica de la máquina real, de manera que en cada una de ellas se pueda ejecutar un sistema operativo diferente, que será el que ofrezca la máquina extendida al usuario.

1.4.4 Cliente/Servidor

Puede ser ejecutado en la mayoría de los computadores, ya sean grandes o pequeños. Este sistema sirve para toda clase de aplicaciones, por tanto, es de propósito general y cumple con las mismas actividades que los sistemas operativos convencionales.

El núcleo tiene como misión establecer la comunicación entre los clientes y los servidores. Los procesos pueden ser tanto servidores como clientes. Por ejemplo, un programa de aplicación normal es un cliente que llama al servidor correspondiente para acceder a un archivo o realizar una operación de entrada/salida sobre un dispositivo concreto. A su vez, un proceso cliente puede actuar como servidor para otro. Este paradigma ofrece gran flexibilidad en cuanto a los servicios posibles en el sistema final, ya que el núcleo provee solamente funciones muy básicas de memoria, entrada/salida, archivos y procesos, dejando a los servidores proveer la mayoría que el usuario final o programador puede usar. Estos servidores deben tener mecanismos de seguridad y protección que, a su vez, serán filtrados por el núcleo que controla el hardware.

1.5 Componentes de un S.O

Los componentes básicos de un sistema operativo son los siguientes:

- **Gestión de procesos:** un proceso es, sencillamente, un programa en ejecución que necesita una serie de recursos para realizar su tarea: tiempo de CPU (Central Process Unit o Unidad de Proceso Central, es decir, el procesador principal del ordenador), memoria, archivos y dispositivos de E/S (entrada/salida).

  Es función del sistema operativo:

☐ **- Planificación de procesos:** decide qué proceso emplea el procesador en cada instante de tiempo.

☐ **- Mecanismos de comunicación entre procesos:** permiten comunicar a dos procesos del sistema operativo.

☐ **- Mecanismos de sincronización:** permiten coordinar a procesos que realizan accesos concurrentes a un cierto recurso.

- **Administración de memoria principal:** la memoria es como un gran almacén con casillas (bytes) a los que se accede mediante una dirección única. Este almacén de datos es compartido por la CPU y los dispositivos de E/S. El Sistema operativo se encarga de gestionar este espacio como responsable de:

✓ Conocer qué partes de la memoria están siendo utilizadas y por quién.

✓ Decidir qué procesos se cargarán en memoria cuando haya espacio disponible.

✓ Asignar y reclamar espacio de memoria cuando sea necesario.

- **Administración de ficheros:** gestiona la manera en que la información se almacena en dispositivos de entrada/salida que permiten el almacenamiento estable.
- **Gestión de los dispositivos de entrada/salida (driver):** parte del sistema operativo que conoce los detalles específicos de cada dispositivo, lo que permite poder operar con él. Además, el sistema operativo ofrece:
  - **Lanzador de aplicaciones:** permite el lanzamiento de un programa. Esto incluye los intérpretes de órdenes textuales y los basados en gestores de ventanas.
  - **Llamadas al sistema:** conjunto de servicios que los procesos pueden solicitar al sistema operativo.

1.6 Arranque, Activación y parada del S.O.

La secuencia de eventos que ocurren entre el tiempo que se enciende la computadora y el tiempo que se pone listo para aceptar comando, recibe el nombre de proceso de arranque.
Y para comprender como funciona en cada sistema operativo, es necesario entenderlo de una manera general.

Unos de los componentes de más importancia en una computadora es la memoria RAM, pero esta es volátil, es decir, no almacena ninguna información cuando la computadora es apagada, por tal razón la computadora no puede usar el contenido de la memoria RAM para recordar muchas de sus funciones básicas, de cómo comunicarse con sus puertos de entrada y de salida con el exterior.

Una computadora necesita de alguna manera dar a la memoria RAM los archivos del Sistema Operativo para que esta pueda iniciar.

Este es uno de los principales objetivos del proceso de arranque.

En general el proceso de arranque sigue estas seis etapas:

1. Encendido (Cuando usted presiona el botón de encendido del case, el power light es iluminado y la energía es distribuida por circuitos internos de la computadora)
2. -Comienzo del programa de arranque (El microprocesador empieza a ejecutar las instrucciones almacenadas en la memoria ROM).
3. Auto prueba de encendido o power-on self-test (La computadora realiza un diagnostico crucial del sistema y de cada componente de la computadora)
4. Carga del Sistema Operativo (El sistema operativo es copiado desde el disco duro a la memoria RAM).
5. Chequeo de la configuración (El microprocesador lee los datos de configuración del CMOs y ejecuta cualquier rutina específica para su uso).
6. Lista para aceptar datos y comandos. (La computadora esta lista para aceptar comandos y datos).

1.7 Interfaces de usuario y del programado

La principal función de la interfaz de usuario del sistema operativo es permitir al usuario acceder y manipular sus objetos; es lo que el usuario ve en pantalla y hace posible la interacción con hardware, permitiéndole dar órdenes o ejecutar programas.

- **Interfaz basada en caracteres:** el usuario utiliza sólo caracteres (letras, números, símbolos). Es la interfaz usada por el usuario cuando abre un terminal.
- **Interfaz gráfica de usuario:** el usuario interacciona utilizando in dispositivo apuntador (por ejemplo, ratón o pantalla táctil) sobre ventanas, iconos y menús. Es tipo de interfaz usada por los sistemas operativos con gestión de ventanas como Linux y Windows.

BIBLIOGRAFÍA

- Tanenbaum S. Adrews. Sistemas Operativos Modernos. (Segunda Edición). Pearson Educación. Madrid.
- Stallings William. (2006). Sistemas Operativos Aspectos Internos y Principios de Diseño (Quinta Edición). (pág. 872). Pearson Educación, S.A., Madrid.
- Tanenbaum S. Andrews. (2009). Sistemas Operativos Modernos. (tercera edición). (pág. 1104). Pearson Educación, México.
- Stallings William. (1997). Sistemas Operativos. (segunda edición). (pág. 732). Prentice Hall. Madrid.
- Meza E., Ruiz E., Wolf G. (2015). Fundamentos de Sistemas Operativos. Universidad Nacional Autónoma de México. México.
- Martínez David. (2001). Sistemas Operativos. (pág. 926). Universidad Nacional del Nordeste. Argentina.

CAPÍTULO II. GESTIÓN DE PROCESOS Y PROCESADOR

Objetivos:

- Describir e identificar el comportamiento y técnicas empleadas para gestionar los procesos en los sistemas operativos.

- Describir la función principal que realizan los procesos en los S.O. Modernos

¿De qué trata esta sesión de aprendizaje?

En esta sesión de aprendizaje se presenta la gestión de procesos dentro del sistema operativo. Se identifican los posibles estados de procesos. Se da a conocer los métodos para la planificación de procesos y los criterios que definen cada método de planificación.

2. Gestión de Procesos y Procesador

2.1 Concepto de proceso

Proceso es el nombre con el que se denomina la ejecución de un programa individual, representado por una serie de instrucciones que el procesador debe ejecutar, la mayoría de los sistemas operativos modernos basan toda su estructura de diseño alrededor de este concepto, debido a esta importancia, Stallings (en [STALLINGS-97]) presenta tres requisitos fundamentales que debe seguir todo sistema operativo en relación a los procesos:

- ✓ El sistema operativo debe intercalar la ejecución de un conjunto de procesos para maximizar la utilización del procesador ofreciendo a la vez un tiempo de respuesta razonable.
- ✓ El sistema operativo debe asignar los recursos a los procesos en conformidad con una política específica (por ejemplo, ciertas funciones o aplicaciones son de prioridad más alta), evitando, al mismo tiempo el interbloqueo.
- ✓ El sistema operativo podría tener que dar soporte a la comunicación entre procesos y la creación de procesos por parte del usuario, labores que pueden ser de ayuda en la estructuración de aplicaciones.

Algunas definiciones del término proceso, que se incluyen en la 5ª edición de Stallings, Sistemas Operativos, Aspectos Internos y Principios de Diseño, son:

- Un programa en ejecución.
- Una instancia de un programa ejecutado en un computador.
- La entidad que se puede asignar y ejecutar en un procesador.
- Una unidad de actividad que se caracteriza por la ejecución de una secuencia de instrucciones, un estado actual, y un conjunto de recursos del sistema asociados.

También se puede pensar en un proceso como en una entidad que consiste en un número de elementos. Los dos elementos esenciales serian el **código de programa** (que puede compartirse con otros procesos que estén ejecutando el mismo programa) y un **conjunto de datos** asociados a dicho código. Supongamos que el procesador comienza a ejecutar este código de programa y que nos referimos a esta entidad en ejecución como un proceso. En cualquier instante puntual del tiempo, mientras el proceso está en ejecución, este proceso se puede caracterizar por una serie de elementos, incluyendo los siguientes:

- **Identificador:** un identificador único asociado a este proceso, para distinguirlo del resto de procesos.
- **Estado:** si el proceso está actualmente corriendo, está en el estado de ejecución.
- **Prioridad:** nivel de prioridad relativo al resto de procesos.
- **Contador de programa:** la dirección de la siguiente instrucción del programa que se ejecutará.
- **Punteros a memoria:** incluye los punteros al código de programa y los datos asociados a dicho proceso, además de cualquier bloque de memoria compartido con otros procesos.
- **Datos de contexto**: estos son datos que están presentes en los registros del procesador cuando el proceso está corriendo.

- **Información de estado de E/S:** incluye las peticiones de E/S pendientes, dispositivos de E/S (por ejemplo, una unidad de cinta) asignados a dicho proceso, una lista de los ficheros en uso por el mismo, etc.
- **Información de auditoría:** puede incluir la cantidad de tiempo de procesador y de tiempo de reloj utilizados, así como los límites de tiempo, registros contables, etc.

Esta información se almacena en una estructura de datos, que se suele llamar bloque de control de proceso (process control block), que el sistema operativo crea y gestiona. El punto más significativo en relación al bloque de control de proceso, o BCP, es que contiene suficiente información de forma que es posible interrumpir el proceso cuando está corriendo y posteriormente restaurar su estado de ejecución como si no hubiera habido interrupción alguna. El BCP es la herramienta clave que permite al sistema operativo dar soporte a múltiples procesos y proporciona multiprogramación, Cuando un proceso se interrumpe, los valores actuales del contador de programas y los registros del procesador se guardan en los campos correspondientes del BCP y el estado del proceso se cambia a cualquier otro valor, como bloqueado o listo. El sistema operativo es libre ahora para poner otro proceso en estado de ejecución. El contador de programa y los datos de contexto se recuperan y cargan en los registros del procesador y este proceso comienza a correr.

Así se puede decir que un proceso está compuesto del código de programa y los datos asociados, además del bloque de control de proceso o BCP. Para un computador monoprocesador, en un instante determinado, como máximo un único proceso puede estar corriendo y dicho proceso estará en estado de ejecución.

2.2 Estados del proceso

Un proceso, a lo largo de su vida, alterna entre diferentes estados de ejecución. Éstos son:

- **Nuevo:** se solicitó al sistema operativo la creación de un proceso, y sus recursos y estructuras están siendo creadas.
- **Listo:** está listo para iniciar o continuar su ejecución, pero el sistema no le ha asignado un procesador.
- **En ejecución:** el proceso está siendo ejecutado en este momento. Sus instrucciones están siendo procesadas en algún procesador.

- **Bloqueado:** en espera de algún evento para poder continuar su ejecución (aun si hubiera un procesador disponible, no podría avanzar).
- **Zombie:** el proceso ha finalizado su ejecución, pero el sistema operativo debe realizar ciertas operaciones de limpieza para poder eliminarlo de la lista.1
- **Terminado:** el proceso terminó de ejecutarse; sus estructuras están a la espera de ser limpiadas por el sistema operativo.

2.3 Descripción del proceso

En un sistema multiprogramado o de tiempo compartido, un proceso es la imagen en memoria de un programa, junto con la información relacionada con el estado de su ejecución.
Un programa es una entidad pasiva, una lista de instrucciones; un proceso es una entidad activa, que empleando al programa define la actuación que tendrá el sistema.
En contraposición con proceso, en un sistema por lotes se habla de tareas.

2.3.1 Proceso nulo

Un problema que debe resolver un sistema operativo multitarea es, qué debería hacer el sistema cuando no hay nada que ejecutar. Por ejemplo, cuando la cola de listos se encuentra vacía. Este problema es resuelto en muchos sistemas operativos con el proceso NULO que es creado por el sistema en el momento de arranque. El proceso nulo nunca termina, no tiene E/S y tiene la prioridad más baja en el sistema. En consecuencia, la cola de listos nunca está vacía, además la ejecución del planificador puede hacerse más rápida al eliminar la necesidad de comprobar si la cola de listos está vacía o no. Algunas de las tareas que se le pueden dar al proceso nulo, por ejemplo, es realizar estadísticas de uso de procesador, o asistencia de vigilancia de la integridad del sistema, etc.

2.3.2 Estados del procesador

Básicamente la información del estado del procesador está formada por el contenido de los registros del procesador. Por supuesto, mientras un proceso está ejecutándose, la información está en los registros. Cuando se interrumpe el proceso, toda la información de los registros debe salvarse de forma que pueda restaurarse cuando el proceso reanude su ejecución. La naturaleza y el número de los registros involucrados dependen del diseño del procesador. Normalmente, en el

conjunto de registros se incluyen los registros visibles para el usuario, los registros de control y de estado.

- **Registros visibles para el usuario:** es aquel al que puede hacerse referencia por medio del lenguaje máquina que ejecuta el procesador. Normalmente, existen de 8 a 32 de estos registros, aunque algunas implementaciones RISC tienen más de 100.
- **Registros de control y de estado:** hay varios registros del procesador que se emplean para controlar su funcionamiento. Entre estos se incluye:
    - **Contador de programa*:*** contiene la dirección de la próxima instrucción a ser tratada
    - **Códigos de condición:** muestran el resultado de la operación aritmética o lógica más reciente (signo, cero, acarreo, igualdad, desbordamiento).
    - **Información de estado:** incluye los indicadores de habilitación o inhabilitación de interrupciones y el modo de ejecución.
- **Punteros de pila:** cada proceso tiene una o más pilas LIFO del sistema asociadas. Las pilas se utilizan para almacenar los parámetros y las direcciones de retorno de los procedimientos y de las llamadas al sistema. El puntero de pila siempre apunta a la cima de la pila.

2.3.3 Imagen del proceso

La posición de la imagen del proceso dependerá del esquema de gestión de memoria que se utilice. En el caso más simple, la imagen del proceso se mantiene como un bloque de memoria contiguo, o continuo. Este bloque se mantiene en memoria secundaria, habitualmente disco. Para que el sistema operativo pueda gestionar el proceso, al menos una pequeña porción de su imagen se debe mantener en memoria principal. Para ejecutar el proceso, la imagen del proceso completa se debe cargar en memoria principal o al menos en memoria virtual. A sí mismo el sistema operativo necesita conocer la posición en disco de cada proceso y, para cada proceso que se encuentre en memoria principal, su posición en dicha memoria.

La mayoría de los sistemas operativos modernos utilizan algún tipo de esquema de gestión de memoria en el que la imagen de un proceso consiste en un conjunto de bloques que no tienen por

qué estar almacenados consecutivamente. Dependiendo del tipo de esquema utilizado, estos bloques pueden ser de longitud variable (llamados segmentos) o de longitud fija (llamadas páginas) o una combinación de ambos. En cualquier caso, tales esquemas permiten al sistema operativo tener que traer solo una parte de un proceso en particular. De este modo, en un momento dado, una parte de La imagen de un proceso puede estar en la memoria principal y el resto en la memoria secundaria. Por tanto, las tablas de procesos deben mostrar la ubicación de cada segmento y/o página de cada imagen de proceso. Si la imagen del proceso contiene varios bloques, entonces esta información estará guardada directamente en la tabla principal o con referencias cruzadas a entradas de las tablas de memoria. Desde luego, esta representación es genérica; cada sistema operativo tiene su propia forma de organizar la información de ubicación.

Datos del Usuario	La parte modificable del espacio de usuario. Puede guardar datos del programa, una zona para una pila del usuario y programas que pueden modificarse.
Programa de Usuario	El programa a ejecutar.
Pila del Sistema	Cada proceso tiene una o más pilas (el ultimo que entra es el primero en salir) asociadas a él. Una pila se utiliza para almacenar los parámetros y las direcciones de retorno.
Bloque de control de proceso	Información necesaria para que el sistema operativo controle al proceso.

Tabla 2.1: Elementos típicos de una imagen de proceso

2.3.4 Información del BCP

La información que debe manipular el sistema operativo relativa a cada uno de los procesos actuales se suele almacenar en una estructura llamada bloque de control de proceso (BCP - Process Control Block). El BCP incluye campos como:

Estado del proceso	El estado actual del proceso
Contador de programa	Cuál es la siguiente instrucción a ser ejecutada por el proceso
Registro del CPU	La información específica del estado del CPU mientras el proceso está en ejecución (debe ser respaldada y restaurada cuando se registra un cambio de estado).
Información de planificación (scheduling)	La prioridad del proceso, la cola en que está agendado, y demás información que puede ayudar al sistema operativo a planificar los procesos.
Información de administración de memoria	La información de mapeo de memoria (páginas o segmentos, dependiendo del sistema operativo), incluyendo la pila de llamadas.
Información de contabilidad	Información de la utilización de recursos que ha tenido este proceso, puede incluir el tiempo total empleado y otros (de usuario, cuando el procesador va avanzando sobre las instrucciones del programa propiamente, de sistema cuando el sistema operativo está atendiendo las solicitudes del proceso), uso acumulado de memoria y dispositivos, etc.
Estado de E/S	Listado de dispositivos y archivos asignados que el proceso tiene abiertos en un momento dado.

Tabla 2.2: Contenido del bloque de control de procesos.

El bloque de control de proceso es la estructura de datos central y más importante de un sistema operativo. Cada bloque de control de proceso contiene toda la información de un proceso necesaria para el sistema operativo. Los bloques son leídos y/o modificados por casi todos los módulos de un sistema operativo, incluyendo aquellos que tienen que ver con la planificación, la asignación de recursos, el tratamiento de interrupciones y el análisis y supervisión del rendimiento. Puede decirse que los conjuntos de los bloques del control de procesos definen el estado del sistema operativo. Esto saca a relucir una cuestión importante de diseño. Una serie de rutinas del sistema operativo necesitarán acceder a la información de los bloques de control de procesos. La provisión de acceso directo a estas tablas no es difícil.

Cada proceso está dotado de un único ID que puede utilizarse como índice en una tabla de punteros a los bloques de control de procesos. La dificultad no está en el acceso, sino más bien en la protección. Existen dos problemas:

- Un error en una sola rutina, como la de tratamiento de interrupciones, puede dañar los bloques de control de procesos, lo que destruirla la capacidad del sistema para administrar los procesos afectados.
- Un cambio de diseño en la estructura o en la semántica del bloque de control de procesos podría afectar a varios módulos del sistema operativo.

Estos problemas se pueden abordar exigiendo a todas las rutinas del sistema operativo que pasen a través de una rutina de manejo, cuya única tarea serla la de proteger los bloques de control de proceso y que se constituiría en el único árbitro para leer y escribir en estos bloques. La concesión en el empleo de una rutina tal está en el rendimiento y en el grado con el que pueda confiarse en que el resto del software del sistema sea correcto.

2.3.5 Estructura del control del S.O

El sistema operativo construye tablas de información sobre cada entidad que esté administrando, información actual sobre cada proceso y de cada recurso.

- **Tablas de memoria:** se emplean para saber que uso reciben las memorias principales y secundarias. También para obtener cualquier otro tipo de información, relacionada con la memoria.
- **Tablas de ficheros:** almacenan toda la información que contiene un fichero o archivo.
- **Tablas de dispositivos de E/S:** contienen toda la información sobre los periféricos o dispositivos de E/S.
- **Tablas de procesos:** se utilizan para gestionar toda la información que utiliza el Sistema Operativo al manejar los distintos procesos.

2.3.6 Control de procesos

Continuando con el desarrollo sobre la forma en que el sistema operativo gestiona los procesos, hace falta distinguir entre el modo de ejecución del procesador que normalmente se asocia con el sistema operativo y el modo que normalmente se asocia con los programas de usuario. La mayoría de los procesadores dan soporte para dos modos de ejecución por lo menos:

- **modo núcleo:** este modo es controlado y utilizado por el Sistema Operativo.
- **módulo usuario:** En este modo se ejecutan los procesos del usuario, no se puede acceder a las estructuras del sistema. Para cambiar entre modos se utilizan ciertos métodos para pasar de modo núcleo a modo usuario, basta con realizar la instrucción:

Changemode (CM): para pasar de modo usuario a modo núcleo el sistema es mucho más complejo. Si el usuario necesita utilizar las estructuras del sistema, deberá realizar una 'llamada al sistema".

- **La llamada al sistema:** funciona como una interrupción, pero proviene del software. De esta manera al producirse la interrupción se pasa automáticamente al modo núcleo. Cuando se produce una de estas 'interrupciones' el microprocesador hace 'un cambio de contexto', es decir guarda toda la información en registros PCB.

2.4 Planificación de procesos

Un sistema operativo debe asignar recursos del computador entre las necesidades potencialmente competitivas de múltiples procesos. En el caso del procesador, el recurso que se debe asignar es el tiempo de ejecución de múltiples procesos. La forma de asignarlo es la planificación.

La función de la planificación debe estar diseñada para satisfacer varios objetivos que incluyen equidad, ausencia de inanición de cualquier proceso, uso eficiente del tiempo del procesador y poca sobrecarga. Además, la función de planificación puede necesitar tener en cuenta diferentes niveles de prioridad o plazos de tiempo real para el inicio o finalización de ciertos procesos.

2.4.1 Conceptos básicos

Cuando más de un proceso es ejecutable desde el punto de vista lógico, el sistema operativo debe decidir cuál de ellos debe ejecutarse en primer término.

El planificador es la porción del sistema operativo que decide y el algoritmo de planificación es el utilizado.

Los principales conceptos relacionados con la Planificación de Procesos son los siguientes:

- ✓ **Planificación apropiativa:** es la estrategia de permitir que procesos ejecutables (desde el punto de vista lógico) sean suspendidos temporalmente.
- ✓ **Planificación no apropiativa:** es la estrategia de permitir la ejecución de un proceso hasta terminar.
- ✓ **Planificación del procesador**: determinar cuándo deben asignarse los procesadores y a qué procesos, lo cual es responsabilidad del sistema operativo.

La planificación involucra también los siguientes objetivos y conceptos:

- ✓ **Ser justa:**
  - Todos los procesos son tratados de igual manera.
  - Ningún proceso es postergado indefinidamente.
- ✓ **Maximizar la capacidad de ejecución:** maximizar el número de procesos servidos por unidad de tiempo.
- ✓ **Maximizar el número de usuarios interactivos:** que reciban unos tiempos de respuestas aceptables.
- ✓ **Ser predecible:** un trabajo dado debe ejecutarse aproximadamente en la misma cantidad de tiempo independientemente de la carga del sistema.
- ✓ **Minimizar la sobrecarga.**
- ✓ **Equilibrar el uso de recursos:** favorecer a los procesos que utilizarán recursos infrautilizados.
- ✓ **Equilibrar respuesta y utilización:** la mejor manera de garantizar buenos tiempos de respuestas es disponer de los recursos suficientes cuando se necesitan, pero la utilización de los recursos podrá ser pobre.
- ✓ **Evitar la postergación indefinida:** se utiliza la estrategia del "envejecimiento"; mientras un proceso espera por un recurso su prioridad debe aumentar, así la prioridad llegará a ser tan alta que el proceso recibirá el recurso esperado.
- ✓ **Asegurar la prioridad:** los mecanismos de planificación deben favorecer a los procesos con prioridades más altas.

- ✓ **Dar preferencia a los procesos que mantiene recursos claves:**
  - Un proceso de baja prioridad podría mantener un recurso clave, que puede ser requerido por un proceso de más alta prioridad.
  - Si el recurso es no apropiativo, el mecanismo de planificación debe otorgar al proceso un tratamiento mejor del que le correspondería normalmente, puesto que es necesario liberar rápidamente el recurso clave.
- ✓ **Degradarse suavemente con cargas pesadas:**
  - Un mecanismo de planificación no debe colapsar con el peso de una exigente carga del sistema; esto se logra no permitiendo que se creen nuevos procesos cuando la carga ya es pesada.

Muchos de estos conceptos y objetivos se encuentran en conflicto entre sí, por lo que la planificación puede llegar a ser muy compleja; definiremos algunos criterios y algoritmos que nos ayudarán a desarrollar un mejor esquema de planificación.

2.4.2 Criterios de planificación

Los principales criterios respecto de un buen algoritmo de planificación según Tanenbaum son la equidad, la eficacia, el tiempo de respuesta, el tiempo de regreso y el rendimiento.

Estos criterios son independientes, y es imposible optimizar todos ellos de forma simultánea. Por ejemplo, proporcionar un buen tiempo de respuesta puede requerir un algoritmo de planificación que cambie entre procesos frecuentemente; esto incrementa la sobrecarga del sistema, reduciendo las prestaciones. De esta forma, el diseño de las políticas de un planificador implica un compromiso entre requisitos competitivos; los pesos relativos dados a los distintos requisitos dependerán de la naturaleza y el uso dado al sistema.

	Criterios orientados al usuario, criterios de rendimientos
Tiempo de respuesta	Para un proceso interactivo, es el intervalo de tiempo transcurrido desde que se emite una solicitud hasta que se comienza a recibir respuesta. A menudo, un proceso empieza a generar alguna salida para el usuario mientras que continúa procesando la solicitud. Así pues, ésta es una medida mejor que el tiempo de retorno desde el punto de vista del usuario. La disciplina de la planificación debe intentar alcanzar un tiempo de respuesta bajo.
Tiempo de retorno	Tiempo transcurrido desde que se lanza un proceso hasta que finaliza. Se incluye el tiempo de ejecución sumado con el tiempo de espera por los recursos, incluyendo el procesador. Es una medida apropiada para trabajos por lotes.
Plazos	Cuando se pueden especificar plazos de terminación de un proceso, el planificador debe subordinar otras metas a la maximización del porcentaje de plazos cumplidos.
	Criterios orientados al usuario
Previsibilidad	Un trabajo dado debería ejecutarse aproximadamente en el mismo tiempo y con el mismo costo a pesar de la carga del sistema. Una gran variación en el tiempo de respuesta o en el tiempo de estancia es malo desde el punto de vista de los usuarios, Puede significar una gran oscilación en la sobrecarga del sistema o la necesidad de poner a punto el sistema para eliminar las inestabilidades.
	Criterios orientados al sistema, Criterios de rendimiento
Productividad	La política de planificación debe intentar maximizar el número de procesos terminados por unidad de tiempo. Esta medida indica la cantidad de trabajo que se está realizando, Depende de la longitud media de cada proceso, pero también está influida por la política de planificación.
Utilización del procesador	Es el porcentaje de tiempo que el procesador está ocupado. Para un sistema compartido costoso, es un criterio significativo.
	Criterios orientados al sistema, otros
Equidad	En ausencia de orientación de los usuarios o de orientación proporcionada por otro sistema, los procesos deben ser tratados de la misma manera, y ningún proceso debe sufrir inanición
Prioridades	Cuando se asignan prioridades a los procesos, la política de planificación debe favorecer a los de mayor prioridad.
Equilibrio de recursos	La política de planificación debe mantener ocupados los recursos del sistema. Se debe favorecer a los procesos que no utilicen recursos sobrecargados. Este criterio también afecta a la planificación de medio y largo plazo.

Tabla 2.3: Criterios de Panificación

2.4.3 Tipos de planificación

La planificación del procesador consiste en asignar los procesos al procesador o los procesadores para que sean ejecutados en algún momento, de forma que se cumplan objetivos del sistema tales como el tiempo de respuesta, la productividad y la eficiencia del procesador. En muchos sistemas, la actividad de planificación se divide en tres funciones independientes de planificación: planificación a largo, medio y corto plazo. Los nombres hacen referencia a la frecuencia relativa con la que son ejecutadas estas funciones.

La planificación a largo plazo se lleva a cabo al crear un proceso nuevo. La creación de un nuevo proceso parte de la decisión de si añadir un proceso al conjunto de procesos activos. La planificación a medio plazo forma parte del proceso de intercambio y tiene como origen la decisión de añadir un proceso a los que se encuentran, al menos parcialmente, en memoria principal y, por tanto, disponibles para ejecutar. La planificación a corto plazo es la decisión de qué proceso en estado Listo será el que ejecute a continuación.

La planificación afecta al rendimiento del sistema, pues determina qué proceso esperará y qué proceso continuará. Fundamentalmente, la planificación no es sino una gestión de colas que minimice la espera y optimice el rendimiento del entorno.

2.4.4 Algoritmos de planificación

- ✓ **Planificación a Plazo fijo**: esta es compleja considerando los siguientes factores:
  - El usuario debe suministrar anticipadamente una lista precisa de recursos necesarios para el proceso, pero generalmente no se dispone de dicha información.
  - La ejecución del trabajo de plazo fijo no debe producir una grave degradación del servicio a otros usuarios.
  - El sistema debe planificar cuidadosamente sus necesidades de recursos hasta el plazo fijo, lo que se puede complicar con las demandas de recursos de nuevos procesos que ingresen al sistema.
  - La concurrencia de varios procesos de plazo fijo (activos a la vez) puede requerir métodos sofisticados de optimización.

- La administración intensiva de recursos puede generar una considerable sobrecarga adicional.

✓ **Planificación Garantizada:** se establecen compromisos de desempeño con el proceso del usuario, por ejemplo, si existen "n" procesos en el sistema, el proceso del usuario recibirá cerca de "1/n" de la potencia de la cpu. El sistema debe tener un registro del tiempo de cpu que cada proceso ha tenido desde su entrada al sistema y del tiempo transcurrido desde esa entrada.

Con los datos anteriores y el registro de procesos en curso de ejecución, el sistema calcula y determina qué procesos están más alejados por defecto de la relación "1/n" prometida y prioriza los procesos que han recibido menos cpu de la prometida**.**

✓ **Planificación del primero en entrar primero en salir (FIFO):** es muy simple, los procesos se despachan de acuerdo con su tiempo de llegada a la cola de listos. Una vez que el proceso obtiene la cpu, se ejecuta hasta terminar, ya que es una disciplina "no apropiativa".

Puede ocasionar que procesos largos hagan esperar a procesos cortos y que procesos no importantes hagan esperar a procesos importantes.

Es más predecible que otros esquemas. No puede garantizar buenos tiempos de respuestas interactivos. Suele utilizarse integrado a otros esquemas, por ejemplo, de la siguiente manera:

- Los procesos se despachas con algún esquema de prioridad.
- Los procesos con igual prioridad se despachan "FIFO".

✓ **Planificación de asignación en rueda (RR: Round Robin):** los procesos se despachan en "FIFO" y disponen de una cantidad limitada de tiempo de cpu, llamada 2división de tiempo" o "cuanto". Si un proceso no termina antes de expirar su tiempo de cpu ocurren las siguientes acciones:

- La cpu es apropiada.
- La cpu es otorgada al siguiente proceso en espera.

- El proceso apropiado es situado al final de la lista d listos.

Es efectiva en ambientes de tiempo compartido. La sobrecarga de la apropiación se mantiene baja mediante mecanismos eficientes de intercambio de contexto y con suficiente memoria principal para los procesos.

- ✓ Tamaño del Cuanto o Quantum
- ✓ **Planificación del trabajo más corto primero (SJF):** es una disciplina no apropiativa y por lo tanto no recomendable en ambientes de tiempo compartido. El proceso en espera con el menor tiempo estimado de ejecución hasta su terminación es el siguiente en ejecutarse.

  Los tiempos promedio de espera son menores que con "FIFO"; también estos tiempos son menos predecibles.

  Favorece a los procesos cortos en detrimentro de los largos.

  Tiende a reducir el número de procesos en espera y el número de procesos que esperan detrás de procesos largos; también requiere de un conocimiento preciso de tiempo de ejecución de un proceso, lo que generalmente se desconoce. Estos se pueden estimar en base a series de valores anteriores.

- ✓ **Planificación del tiempo restante más corto (SRT):** es la contraparte apropiativa del SJF. Es útil es sistemas de tiempo compartido.
  - El proceso con el tiempo estimado de ejecución menor para finalizar es el siguiente en ser ejecutado.
  - Un proceso en ejecución puede ser apropiado por un nuevo proceso con un tiempo estimado de ejecución menor.
  - Tiene mayor sobrecarga que la planificación SJF.
  - Debe mantener un registro del tiempo de servicio transcurrido del proceso en ejecución, lo que aumenta la sobrecarga.
  - Los trabajos largos tienen un promedio y una varianza de los tiempos de espera aún mayor que en SJF,

- La apropiación de un proceso a punto de terminar por otro de menor duración recién llegado podría significar un mayor tiempo de cambio de contexto (administración del procesador) que el tiempo de finalización del primero.

✓ **Planificación el siguiente con relación de respuesta máxima (HRN**): corrige algunas de las debilidades de SJF, tales como el exceso de perjuicio hacia los procesos largos y el exceso de favoritismo hacia los nuevos trabajos cortos. Es un esquema no apropiativo.

La prioridad de cada proceso está en función no solo del tiempo de servicio del trabajo, sino que también influye la cantidad de tiempo que el trabajo ha estado esperando ser servido.

Cuando un proceso ha obtenido la cpu, corre hasta terminar.

2.5 Monoprocesadores

Los monoprocesadores, solo pueden ejecutar un proceso a la vez. Esto quiere decir que, si se requiere que se ejecuten varias tareas al mismo tiempo, no va a ser posible que se realicen con satisfacción. Lo que pueden hacer los monoprocesadores es alternar las tareas, y por eso los procesadores con grandes capacidades de velocidad, dan una simulación de multiprocesadores, ya que parece que se están ejecutando varios procesos al mismo tiempo, pero esto no es cierto.

2.6 Procesos ligeros o hebras

Un proceso ligero (thread o hebra) es un programa en ejecución que comparte la imagen de la memoria y otras informaciones con otros procesos ligeros.

Los procesos ligeros son una unidad básica de utilización de la CPU consistente en un juego de registros y un espacio de pila. Comparte el código, los datos y los recursos con sus hebras pares

Una tarea (o proceso pesado) está formada ahora por una o más hebras.

Una hebra sólo puede pertenecer a una tarea

- **Características**
  - ✓ Se comparten recursos. La compartición de la memoria permite a las hebras pares comunicarse sin usar ningún mecanismo de comunicación inter-proceso del SO.

- ✓ La conmutación de contexto es más rápida gracias al extenso compartir de recursos
- ✓ No hay protección entre las hebras. Una hebra puede escribir en la pila de otra hebra del mismo proceso.

2.7 Gestión de procesos en Linux

Un proceso, o tarea, en Linux se representa por una estructura de datos task_struct, que contiene información de diversas categorías:

- **Estado:** estado d ejecución del proceso.
- **Información de planificación:** un proceso puede ser normal o de tiempo real o tener prioridad.
- **Identificadores:** son únicos por cada proceso, también hay para usuario y grupo; asignan privilegios.
- **Comunicación entre procesos:** soporta el mecanismo IPC.
- **Enlaces:** incluyen enlaces a sus padres, a sus hermanos y a sus hijos.
- **Tiempos y temporizadores:** tiempo de creación de proceso y tiempo en ser procesado. Pueden tener asociados uno o más temporizadores definidos por una llamada al sistema.
- **Sistema de archivos:** incluye punteros a cualquier archivo abierto por el proceso, punteros a los directorios actual y raíz.
- **Espacio de direcciones:** espacio de direcciones virtual asignado a el proceso.

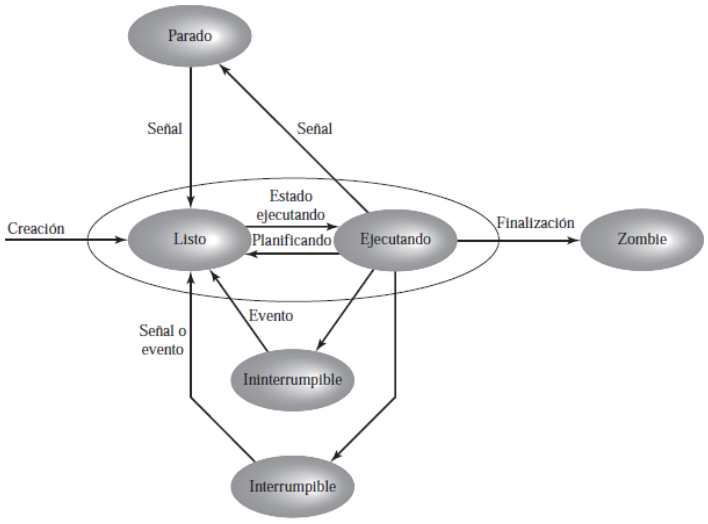

Figura 2.1: Modelo de procesos el Linux.

- **Contexto específico del procesador:** información de los registros y de la pila que constituyen el contexto del proceso.
- **Ejecutando:** este valor de estado se corresponde con dos estados. Un proceso puede estar ejecutando o está listo para ejecutar.
- **Interrumpible:** es un estado bloqueado, en el que el proceso está esperando por un evento, como la finalización de una operación de e/s.
- **Ininterrumpible:** estado bloqueado esperando directamente sobre un estado del hardware y por tanto no manejará ninguna señal.
- **Detenido:** proceso parado, puede ser reanudado por la acción positiva de otro proceso.
- **Zombie:** el proceso se ha terminado, pero, por alguna razón, todavía debe tener su estructura de tarea en la tabla de procesos.

2.8 Gestión de procesos en Windows

Las estructuras de los procesos y los servicios proporcionados por el núcleo de Windows son relativamente sencillos y de propósito general, permitiendo a cada subsistema del sistema operativo que emule una estructura y funcionalidad particular del proceso. Algunas características importantes de los procesos Windows son las siguientes:

- Los procesos están implementados como objetos.
- Un proceso ejecutable puede contener uno o más hilos.
- El objeto proceso tiene funcionalidades de sincronización preconstruidas.

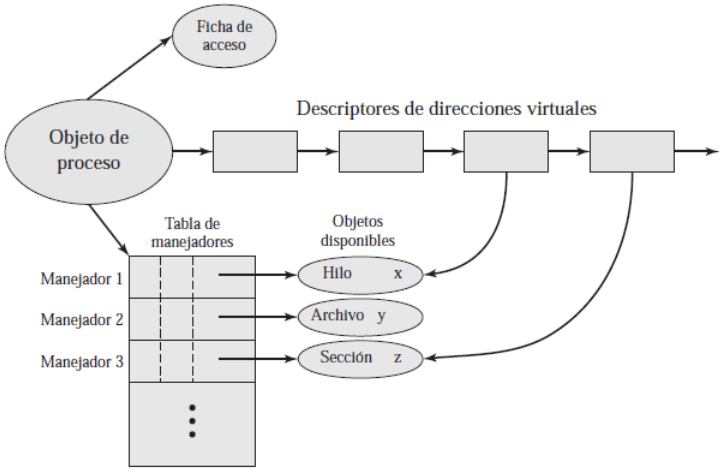

Figura 2.2: Un proceso de Windows y sus recursos.

La figura 2.2 muestra la forma en la que un proceso se asocia los recursos que controla o utiliza. A cada proceso se le asigna un testigo(token) de seguridad de acceso, denominada la ficha principal del proceso. Cuando un usuario inicia sesión, Windows crea una ficha de acceso que incluye el ID de seguridad para el usuario. Cada proceso que se crea o ejecuta en representación de este usuario, tiene una copia de este testigo de acceso. Windows lo utiliza para comprobar si el usuario puede acceder objetos de seguridad, o puede realizar funciones restringidas en el sistema o en un objeto de seguridad. El testigo de acceso controla si un proceso puede modificar sus propios atributos. En este caso, el proceso no tiene un manejador abierto hacia su testigo de acceso. Si el proceso intenta abrir este manejador, el sistema de seguridad determinará si está permitido y por tanto si el proceso puede modificar sus propios atributos.

También relacionado con el proceso, hay una serie de bloques que definen el espacio de direcciones virtuales actualmente asignado al proceso. El proceso no puede modificar directamente estas estructuras, ya que dependen del gestor de memoria virtual, que proporciona servicios de asignación de memoria a los procesos.

BIBLIOGRAFÍA

- Stallings William. (2006). Sistemas Operativos Aspectos Internos y Principios de Diseño (Quinta Edición). (pág. 872). Pearson Educación, S.A., Madrid.
- Tanenbaum S. Andrews. (2009). Sistemas Operativos Modernos. (tercera edición). (pág. 1104). Pearson Educación, México.
- Meza E., Ruiz E., Wolf G. (2015). Fundamentos de Sistemas Operativos. Universidad Nacional Autónoma de México. México.
- Stallings William. (1997). Sistemas Operativos. (segunda edición). (pág. 732). Prentice Hall. Madrid.
- H.M. Deitel. (1987). Introducción a los Sistemas Operativos. Addison-Wesley Iberoamericana, México.
- Pachón L. Wilfredo. (2003). Fundamentos de Sistemas Operativos con Énfasis en GNU/LINUX. (pág.155).
- Martínez David. (2001). Sistemas Operativos. (pág. 926). Universidad Nacional del Nordeste. Argentina.

CAPÍTULO III. GESTIÓN DE MEMORIA

Objetivos

- Identificar los requerimientos necesarios para la gestión de memoria.
- Identificar los mecanismos fundamentales utilizados en la administración de memoria.
- Conocer el modelo de memoria de un proceso en memoria.
- Describir la partición, asignación e intercambio de memoria.
- Describir los elementos que conlleva la memoria virtual en un SO.
- Describir el manejo de memoria en los sistemas operativos actuales

¿De qué trata esta sesión de aprendizaje?

En esta sesión de aprendizaje se analizará la gestión de memoria como parte fundamental del sistema operativo. Así como a su vez la administración de memoria y los distintos métodos y operaciones que se encargan de obtener la máxima utilidad de la memoria.

3.1 Conceptos fundamentales

La memoria es uno de los recursos más valioso que gestiona el sistema operativo. Uno de los elementos principales que caracterizan un proceso es la memoria que utiliza. Ésta está lógicamente separada de la de cualquier otro proceso del sistema (excepto los theads de un mismo proceso que comparten normalmente la mayor parte de la memoria que tiene asignada) Un proceso no puede acceder, al espacio de memoria asignado a otro proceso, lo cual es imprescindible para la seguridad y estabilidad del sistema. En un sistema monoprogramado, la memoria principal se divide en dos partes: una parte para el sistema operativo (monitor residente, núcleo) y otra parte para el programa que se ejecuta en ese instante. En un sistema multiprogramado, la parte de "usuario" de la memoria debe subdividirse aún más para hacer sitio a varios procesos. La tarea de subdivisión la lleva a cabo dinámicamente el sistema operativo y se conoce como **gestión de memoria.**

En un sistema multiprogramado resulta vital una gestión efectiva de la memoria. Si sólo hay unos pocos procesos en memoria, entonces la mayor parte del tiempo estarán esperando a la E/S y el procesador estará desocupado. Por ello, hace falta repartir eficientemente la memoria para meter tantos procesos como sea posible.

3.2 Requerimientos de la gestión de memoria

La administración de memoria se refiere a los distintos métodos y operaciones que se encargan de obtener la máxima utilidad de la memoria, organizando los procesos y programas que se ejecutan de manera tal que se aproveche de la mejor manera posible el espacio disponible.

Para poder lograrlo, la operación principal que realiza es trasladar la información que deberá ser ejecutada por la unidad central de procesamiento o procesador, a la memoria principal. Actualmente esta administración se conoce como memoria virtual, porque no es la memoria física del procesador sino una memoria virtual que la representa. Entre algunas ventajas, esta memoria permite que el sistema cuente con una memoria más extensa teniendo la misma memoria real, por lo que esta se puede utilizar de manera más eficiente. Y por supuesto, que los programas que son utilizados no ocupen lugar innecesario.

Las técnicas que existen para la carga de programas en la memoria son: partición fija, que es la división de la memoria libre en varias partes (de igual o distinto tamaño) y la partición dinámica, que son las particiones de la memoria en tamaños que pueden ser variables, según la cantidad de memoria que necesita cada proceso.

Entre las principales operaciones que desarrolla la administración de memoria se encuentran la reubicación, que consiste en trasladar procesos activos dentro y fuera de la memoria principal para maximizar la utilización del procesador; la protección, mecanismos que protegen los procesos que se ejecutan de interferencias de otros procesos; uso compartido de códigos y datos, con lo que el mecanismo de protección permite que ciertos procesos de un mismo programa que comparten una tarea tengan memoria en común.

3.2.1 Reubicación

En un sistema multiprogramado, la memoria principal disponible se comparte generalmente entre varios procesos. Normalmente, no es posible que el programador sepa anticipadamente que programas residirán en memoria principal en tiempo de ejecución de su programa, así como también poder intercambiar procesos en la memoria principal para maximizar la utilización del procesador, proporcionando un gran número de procesos para la ejecución. Una vez que un

programa se ha llevado al disco, sería bastante limitante tener que colocarlo en la misma región de memoria principal donde se hallaba anteriormente, cuando éste se trae de nuevo a la memoria. Por el contrario, podría ser necesario reubicar el proceso a un área de memoria diferente.

Por tanto, no se puede conocer de forma anticipada dónde se va a colocar un programa y se debe permitir que los programas se puedan mover en la memoria principal, debido al intercambio o swap.

Estos hechos ponen de manifiesto algunos aspectos técnicos relacionados con el direccionamiento, como se muestras en la siguiente figura. La figura 3.1 representa la imagen de un proceso, el sistema operativo necesitará conocer la ubicación de la información de control del

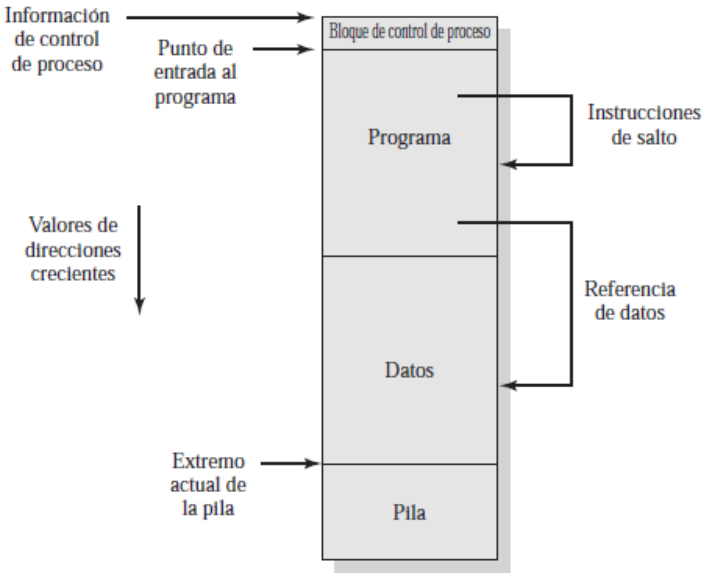

Figura 3.1: Requisitos de direccionamiento para un proceso

proceso y de la pila de ejecución, así como el punto de entrada que utilizará el proceso para iniciar la ejecución. Debido a que el sistema operativo se encarga de gestionar la memoria y es responsable de traer el proceso a la memoria principal, estas direcciones son fáciles de adquirir, además también el procesador debe tratar con referencias de memoria dentro del propio programa. Las instrucciones de salto contienen una dirección para referenciar la instrucción que se va a ejecutar a continuación. Las instrucciones de referencia de los datos contienen la dirección del byte o palabra de datos referenciados. De alguna forma, el hardware del procesador

y el software del sistema operativo deben poder traducir las referencias de memoria encontradas en el código del programa en direcciones de memoria físicas, que reflejan la ubicación actual del programa en la memoria principal.

3.2.2 Protección

Cada proceso debe protegerse contra interferencias no deseadas por parte de otros procesos, sean accidentales o intencionadas. Por tanto, los programas de otros procesos no deben ser capaces de referenciar sin permiso posiciones de memoria de un proceso, tanto en modo lectura como escritura. La mayoría de los lenguajes de programación permite el cálculo dinámico de direcciones en tiempo de ejecución; por tanto, todas las referencias de memoria generadas por un proceso deben comprobarse en tiempo de ejecución para poder asegurar que se refieren sólo al espacio de memoria asignado a dicho proceso. Los mecanismos para reasignación también dan soporte a los de protección.

Normalmente, un proceso de usuario no puede acceder a cualquier porción del sistema operativo, ni al código ni a los datos. Sin un trato especial, un programa de un proceso no puede acceder al área de datos de otro proceso. El procesador debe ser capaz de abortar tales instrucciones en el punto de ejecución.

Los requisitos de protección de memoria deben ser satisfechos por el procesador en el lugar del sistema operativo. Esto es debido a que el sistema operativo no puede anticipar todas las referencias de memoria que un programa hará. Incluso si tal anticipación fuera posible, llevaría demasiado tiempo calcularlo para cada programa a fin de comprobar la violación de referencia de la memoria. Por tanto, solo es posible evaluar la permisibilidad de una referencia en tiempo de ejecución de la instrucción que realiza dicha referencia.

3.2.3 Compartición

Cualquier mecanismo de protección debe tener flexibilidad de permitir a varios procesos acceder a la misma porción de memoria principal. Por ejemplo, si varios programas están ejecutando el mismo programa, es ventajoso permitir que cada proceso pueda acceder a la misma copia del programa en lugar de tener su propia separada. Procesos que estén cooperando en la misma tarea podrían necesitar compartir el acceso controlado a áreas de memoria compartidas sin

comprometer la protección esencial. Igualmente, los mecanismos que le dan soporte a la reubicación también se lo brindan a la compartición.

3.3 Organización lógica y física

- **Organización lógica:**

  La mayoría de los programas se organizan en módulos, algunos de los cuales no se pueden modificar (sólo lectura, sólo ejecución) y algunos de los cuales contienen datos que si se pueden modificar. Si el sistema operativo y el hardware del computador pueden tratar de forma efectiva los programas de usuarios y los datos de módulos de algún tipo, entonces se pueden lograr varias ventajas:

1. Los módulos se pueden escribir y compilar independientemente, con todas las referencias de un módulo desde otro resueltas por el sistema en tiempo de ejecución.

2. Con una sobrecarga adicional modesta, se puede proporcionar diferentes grados de protección a los módulos (solo lectura, sólo ejecución).

3. Es posible introducir mecanismos por los cuales los módulos se pueden compartir entre los procesos. La ventaja de proporcionar compartición a nivel de módulo es que se corresponde con la forma en la que el usuario ve el problema, y por tanto es fácil para éste especificar la compartición deseada.

- **Organización física:**

  La memoria se organiza en dos niveles, la memoria principal y la secundaria. La memoria proporciona acceso rápido, pero generalmente no proporciona almacenamiento permanente. La memoria secundaria es más lenta y proporciona la gran mayoría de las veces almacenamiento permanente. Por tanto, la memoria secundaria de larga capacidad puede proporcionar almacenamiento para programas y datos a largo plazo, mientras que una memoria principal más pequeña contiene programas y datos actualmente en uso.

  En este esquema de dos niveles, la organización del flujo de información entre la memoria principal y secundaria supone una de las preocupaciones principales del sistema. La responsabilidad para este flujo podría asignarse a cada programador en particular, pero no es practicable o deseable por estos motivos:

1. La memoria principal disponible para un programa más sus datos podría ser insuficiente. En este caso, el programador debería utilizar una técnica conocida como superposición, en la cual los programas y los datos se organizan de tal forma que se puede asignar la misma región de memoria a varios módulos, con un programa principal responsable para intercambiar los módulos entre disco y memoria según las necesidades. Incluso con la ayuda de herramientas de compilación, la programación con superposición malgasta tiempo del programador.

2. En un entorno multiprogramado, el programador no conoce en tiempo de codificación cuánto espacio estará disponible o dónde se localizará dicho espacio.

3.4 Modelo de memoria de un proceso

Cuando un sistema operativo inicia un proceso, no se limita a volcar el archivo ejecutable a memoria, sino que tiene que proporcionar la estructura para que éste vaya guardando la información de estado relativa a su ejecución.

- **Sección (o segmento) de texto** Es el nombre que recibe la imagen en memoria de las instrucciones a ser ejecutadas. Usualmente, la sección de texto ocupa las direcciones más bajas del espacio en memoria.
- **Sección de datos** Espacio fijo preasignado para las variables globales y datos inicializados (como las cadenas de caracteres, por ejemplo). Este espacio es fijado en tiempo de compilación, y no puede cambiar (aunque los datos que cargados allí sí cambian en el tiempo de vida del proceso).
- **Espacio de libres** Espacio de memoria que se emplea para la asignación dinámica de memoria durante la ejecución del proceso. Este espacio se ubica por encima de la sección de datos, y crece hacia arriba. Este espacio es conocido en inglés como el Heap. Cuando el programa es escrito en lenguajes que requieren manejo dinámico manual de la memoria (como C), esta área es la que se maneja mediante las llamadas de la familia de malloc y free. En lenguajes con gestión automática, esta área es monitoreada por los recolectores de basura.
- **Pila de llamadas** Consiste en un espacio de memoria que se usa para almacenar la secuencia de funciones que han sido llamadas dentro del proceso, con sus parámetros,

direcciones de retorno, variables locales, etc. La pila ocupa la parte más alta del espacio en memoria, y crece hacia abajo.

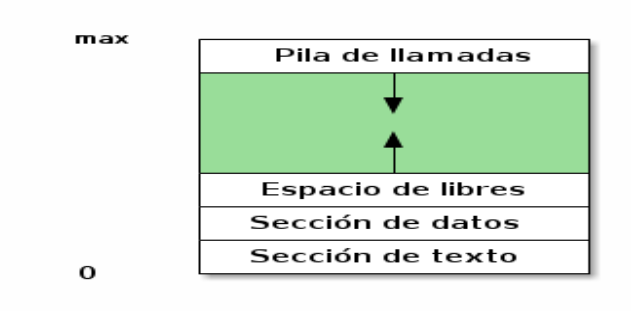

Figura 3.2: Regiones de la memoria para un proceso.

3.4.1 Fases en la generación de un ejecutable

En general, una aplicación estará compuesta por un conjunto de módulos de código fuente que deberán ser procesados para obtener el ejecutable de la aplicación. Este procesamiento típicamente costa de dos fases: **compilación**, que genera el código máquina correspondiente a cada módulo fuente de la aplicación; y **enlace,** que genera un ejecutable agrupando todos los archivos objeto y resolviendo las referencias entre módulos.

Además de referencias entre módulos, pueden existir referencias a símbolos definidos en otros archivos objeto previamente compilados agrupados normalmente en bibliotecas.

La manera de generar el ejecutable, consiste en compilar los módulos fuente de la aplicación y enlazar los módulos objeto resultantes junto con los extraídos de las bibliotecas correspondientes.

3.4.2 Mapa de memoria de un proceso

El mapa de memoria de un proceso está formado por distintas regiones o segmentos. Cuando se activa la ejecución de un programa, se crean varias regiones dentro del mapa a partir de la información del ejecutable. Las regiones iniciales del proceso se van a corresponder básicamente con las distintas secciones del ejecutable.

Cada región es una zona contigua que está caracterizada por la dirección dentro del mapa del proceso donde comienza y por su tamaño. Además, tendrá asociada una serie de propiedades y características específicas, tales como las siguientes:

- **Soporte de la región:** donde está almacenado el contenido inicial de la región. Se presentan dos posibilidades: soporte en archivo (está almacenado en un archivo o en parte del mismo); sin soporte (no tiene un contenido inicial).
- **Tipo de uso compartido:** puede ser privada (el contenido de la región sólo es accesible al proceso que la contiene), compartida (puede ser compartida por varios procesos).
- **Protección:** tipo de acceso a la región permitido. Se proporcionan tres tipos, de lectura, ejecución y escritura.
- **Tamaño fijo o variable:** en el caso de regiones de tamaño variable, se suele distinguir si la región crece hacia direcciones de regiones menores o mayores.

Las regiones que presentan el mapa de memoria inicial del proceso se corresponden básicamente con las secciones del ejecutable más la pila inicial del proceso, a saber:

- **Código:** se trata de una región compartida de lectura/ejecución, es de tamaño fijo.
- **Datos con valor inicial:** se trata de una región privada, ya que cada proceso que ejecuta un determinado programa necesita una copia propia de las variables del mismo. Es de lectura y escritura y de tamaño fijo.
- **Datos sin valor inicial:** región privada de tamaño fijo, contenido inicial irrelevante.
- **Pila:** servirá de soporte para almacenar los registros de activación de las llamadas a funciones; es de tamaño variable que crecerá cuando se produzcan llamadas y decrecerá cuando se retorne de las mismas.

Los sistemas operativos modernos ofrecen un modelo de memoria dinámico en el que el mapa de un proceso está formado por un número variable de regiones que pueden añadirse o eliminarse durante la ejecución del mismo.

3.4.3 Operaciones sobre regiones

El espacio de direccionamiento de un proceso se compone de varias regiones de memoria y cada región de memoria se caracteriza por varios atributos: sus direcciones de inicio y fin; los derechos de acceso que tiene asociados; el objeto asociado. Las regiones de memoria contenidas en el espacio de direccionamiento de un proceso pueden determinarse mostrando el contenido del archivo maps, situado en el directorio de cada proceso en el sistema de archivos/proc.

Las operaciones que el kernel del sistema operativo puede realizar sobre las regiones de memoria de un proceso son las siguientes:

- **Creación de región:** al crear mapa inicial o por solicitud posterior.
- **Liberación de región:** al terminar el proceso o por solicitud posterior.
- **Cambio de tamaño de región:** del heap o de la pila.
- **Duplicado de región**: operación requerida por el servicio fork.

3.5 Partición estática y dinámica

- **Partición estática**

En la mayoría de los esquemas de gestión de memoria, se puede suponer que el sistema operativo ocupa una parte fija de memoria principal y que el resto de la memoria está disponible para ser usado por varios procesos. El esquema más sencillo de gestión de la memoria disponible es dividirla en regiones con límites fijos.

Una posibilidad es emplear particiones de igual tamaño. En este caso, cualquier proceso cuyo tamaño sea menor o igual que el tamaño de la partición puede cargarse en cualquier partición libre. Si todas las particiones están ocupadas y no hay procesos residentes en estado Listo o Ejecutando, el sistema operativo puede sacar un proceso de alguna de las particiones y cargar otro proceso de forma que haya trabajo para el procesador.

Las particiones fijas de igual tamaño plantean dos dificultades:

- Un programa puede ser demasiado grande para caber en la partición. En este caso, el programador debe diseñar el programa mediante superposiciones, para que sólo una parte del programa esté en memoria principal en cada instante. Cuando se necesita un módulo que no está presente, el programa de usuario debe cargar dicho módulo en la partición del programa, superponiéndose a los programas y datos que se encuentren en ella.
- El uso de memoria principal es extremadamente ineficiente. Cualquier programa, sin importar lo pequeño que sea, ocupará una partición completa. En el ejemplo, puede haber un programa que ocupe menos de 128Kb de memoria y, aun así, ocuparía una partición de 512Kb cada vez que se cargase. Este fenómeno, en el que se malgasta el espacio interno

de una partición cuando el bloque de datos cargado sea más pequeño que la partición, se denomina **fragmentación interna.**

Pueden reducirse, aunque no solventarse, ambos problemas, por medio del empleo de particiones de tamaños distintos.

- **Partición Dinámica**

Para superar algunas de las dificultades de la partición estática, se desarrolló una solución denominada *partición dinámica.* Otra vez, este enfoque ha sido superado de largo por técnicas de gestión de memoria más sofisticadas.

Con la partición dinámica, las particiones son variables en número y longitud. Cuando se trae un proceso a memoria principal, se le asigna exactamente tanta memoria como necesita y no más. Por ejemplo, una tarea que usa 1MB de memoria principal. Al principio, la memoria principal está vacía, exceptuando el sistema operativo. Se cargan los tres primeros procesos, empezando en donde acaba el sistema operativo y ocupando sólo un espacio suficiente para cada proceso. Esto deja un "hueco" al final de la memoria demasiado pequeño para un cuarto proceso. En algún momento, ninguno de los procesos en memoria está listo. Así pues, el sistema operativo saca al proceso 2, que deja sitio suficiente para cargar un nuevo proceso, el proceso 4. Puesto que el proceso 4 es más pequeño que el proceso 2, se crea otro hueco pequeño. Más tarde, se alcanza un punto en el que ninguno de los procesos que están en memoria principal están listos y el proceso 2, que está en estado Listo, pero suspendido, está disponible. Puesto que no hay suficiente sitio en memoria para el proceso 2, el sistema operativo expulsa al proceso 1 y carga de nuevo el proceso 2 . Como se muestra en el ejemplo, este método comienza bien, pero, finalmente, desemboca en una situación en la que hay un gran número de huecos pequeños en memoria. Conforme pasa el tiempo, la memoria comienza a estar más fragmentada y su rendimiento decae. Este fenómeno se denomina fragmentación externa y se refiere al hecho de que la memoria externa a todas las particiones se fragmenta cada vez más. Una técnica para superar la fragmentación externa es la **compactación:** De vez en cuando, el sistema operativo desplaza los procesos para que estén contiguos de forma que toda la memoria libre quede junta en un bloque. Por ejemplo, una compactación produce un bloque de memoria libre de 256K. Este hueco puede ser suficiente para cargar un proceso adicional. La dificultad de la compactación está en que es un procedimiento

que consume tiempo, por lo que desperdicia tiempo del procesador. La compactación necesita de la capacidad de reubicación dinámica. Es decir, se debe poder mover un programa de una región a otra de memoria principal sin invalidar las referencias a memoria del programa.

3.6 Esquemas de Memoria basado en Asignación Contigua

En los sistemas de ejecución en lotes, así como en las primeras computadoras personales, sólo un programa se ejecutaba a la vez. Por lo que, más allá de la carga del programa y la satisfacción de alguna eventual llamada al sistema solicitando recursos, el sistema operativo no tenía que ocuparse de la asignación de memoria.

Al nacer los primeros sistemas operativos multitarea, se hizo necesario resolver cómo asignar el espacio en memoria a diferentes procesos.

La respuesta más sencilla es la de asignar a cada programa a ser ejecutado un bloque contiguo de memoria de un tamaño fijo. En tanto los programas permitieran la resolución de direcciones en tiempo de carga o de ejecución, podrían emplear este esquema. El sistema operativo emplearía una región específica de la memoria del sistema (típicamente la región baja, desde la dirección en memoria 0x00000000 hacia arriba), y una vez terminado el espacio necesario para el núcleo y sus estructuras, el sistema asigna espacio a cada uno de los procesos. Si la arquitectura en cuestión permite limitar los segmentos disponibles a cada uno de los procesos, esto sería suficiente para alojar en memoria varios procesos y evitar que interfieran entre sí.

Desde la perspectiva del sistema operativo, cada uno de los espacios asignados a un proceso es una partición. Cuando el sistema operativo inicia, toda la memoria disponible es vista como un sólo bloque, y conforme se van ejecutando procesos, este bloque va siendo subdividido para satisfacer sus requisitos. Al cargar un programa el sistema operativo calcula cuánta memoria va a requerir a lo largo de su vida prevista. Esto incluye el espacio requerido para la asignación dinámica de memoria con la familia de funciones malloc y free.

- **Fragmentación**

Es un fenómeno que se manifiesta a medida que los procesos terminan su ejecución, y el sistema operativo libera la memoria asignada a cada uno de ellos. Si los procesos se encontraban en regiones de memoria, apartadas entre sí, comienzan a aparecer regiones de memoria disponible, interrumpidas por regiones de memoria usada por los procesos que aún se encuentran activos.

Si la computadora no tiene hardware específico que permita que los procesos resuelvan sus direcciones en tiempo de ejecución, el sistema operativo no puede reasignar los bloques existentes, y aunque pudiera hacerlo, mover un proceso entero en memoria puede resultar una operación costosa en tiempo de procesamiento.

Al crear un nuevo proceso, el sistema operativo tiene tres estrategias según las cuales podría asignarle uno de los bloques disponibles:

1. Primer ajuste El sistema toma el primer bloque con el tamaño suficiente para alojar el nuevo proceso. Este es el mecanismo más simple de implementar y el de más rápida ejecución. No obstante, esta estrategia puede causar el desperdicio de memoria, si el bloque no es exactamente del tamaño requerido.

1. **Mejor ajuste** El sistema busca entre todos los bloques disponibles cuál es el que mejor se ajusta al tamaño requerido por el nuevo proceso. Esto implica la revisión completa de la lista de bloques, pero permite que los bloques remanentes, una vez que se ubicó al nuevo proceso, sean tan pequeños como sea posible (esto es, que haya de hecho un mejor ajuste).
2. **Peor ajuste** El sistema busca cuál es el bloque más grande disponible, y se lo asigna al nuevo proceso. Empleando una estructura de datos como un montículo, esta operación puede ser incluso más rápida que la de primer ajuste. Con este mecanismo se busca que los bloques que queden después de otorgarlos a un proceso sean tan grandes como sea posible, de cierto modo balanceando su tamaño.

La fragmentación externa se produce cuando hay muchos bloques libres entre bloques asignados a procesos; la fragmentación interna se refiere a la cantidad de memoria dentro de un bloque que nunca se usará

- **Compactación**

Un problema importante que va surgiendo como resultado de esta fragmentación es que el espacio total libre de memoria puede ser mucho mayor que lo que requiere un nuevo proceso, pero al estar fragmentada en muchos bloques, éste no encontrará una partición contigua donde ser cargado. Si los procesos emplean resolución de direcciones en tiempo de ejecución, cuando el sistema operativo comience a detectar un alto índice de fragmentación, puede lanzar una operación de compresión o compactación. Esta operación consiste en mover los contenidos en

memoria de los bloques asignados para que ocupen espacios contiguos, permitiendo unificar varios bloques libres contiguos en uno solo.
La compactación tiene un costo alto, involucra mover prácticamente la totalidad de la memoria (probablemente más de una vez por bloque).

3.7 Zona de Intercambio

Siguiendo de cierto modo la lógica requerida por la compactación se encuentran los sistemas que utilizan intercambio (swap) entre la memoria primaria y secundaria. En éstos, el sistema operativo puede comprometer más espacio de memoria del que tiene físicamente disponible. Cuando la memoria se acaba, el sistema suspende un proceso (usualmente un proceso "bloqueado") y almacena una copia de su imagen en memoria en almacenamiento secundario para luego poder restaurarlo.

Hay algunas restricciones que observar previo a suspender un proceso. Por ejemplo, se debe considerar si el proceso tiene pendiente alguna operación de entrada/salida, en la cual el resultado se deberá copiar en su espacio de memoria. Si el proceso resultara suspendido (retirándolo de la memoria principal), el sistema operativo no tendría dónde continuar almacenando estos datos conforme llegaran. Una estrategia ante esta situación podría ser que todas las operaciones se realicen únicamente a buffers (regiones de memoria de almacenamiento temporal) en el espacio del sistema operativo, y éste las transfiera el contenido del buffer al espacio de memoria del proceso suspendido una vez que la operación finalice.

Esta técnica se popularizó en los sistemas de escritorio hacia finales de los 1980 y principios de los 1990, en que las computadoras personales tenían típicamente entre 1 y 8 MB de memoria.

Se debe considerar que las unidades de disco son del orden de decenas de miles de veces más lentas que la memoria, por lo que este proceso resulta muy caro. Por ejemplo, si la imagen en memoria de un proceso es de 100 MB, bastante conservador hoy en día, y la tasa de transferencia sostenida al disco de 50 MB/s, intercambiar un proceso al disco toma dos segundos. Cargarlo de vuelta a la memoria toma otros dos segundos, y a esto debe sumarse el tiempo de posicionamiento de la cabeza de lectura/escritura, especialmente si el espacio a emplear no es

secuencial y contiguo. Resulta obvio por qué esta técnica ha caído en desuso conforme aumenta el tamaño de los procesos.

3.8 Memoria Virtual

En un sistema que emplea paginación, un proceso no conoce su dirección en memoria relativa a otros procesos, sino que trabajan con una idealización de la memoria, en la cual ocupan el espacio completo de direccionamiento, desde el cero hasta el límite lógico de la arquitectura, independientemente del tamaño físico de la memoria disponible.

Y si bien en el modelo mencionado de paginación los diferentes procesos pueden compartir regiones de memoria y direccionar más memoria de la físicamente disponible, no se ha presentado aún la estrategia que se emplearía cuando el total de páginas solicitadas por todos los procesos activos en el sistema superara el total de espacio físico. Es ahí donde entra en juego la memoria virtual: para ofrecer a los procesos mayor espacio en memoria de con el que se cuenta físicamente, el sistema emplea espacio en almacenamiento secundario (típicamente, disco duro), mediante un esquema de intercambio (swap) guardando y trayendo páginas enteras.

Es importante apuntar que la memoria virtual es gestionada de forma automática y transparente por el sistema operativo. No se hablaría de memoria virtual, por ejemplo, si un proceso pide explícitamente intercambiar determinadas páginas. Puesto de otra manera: del mismo modo que la segmentación permitió hacer mucho más cómodo y útil al intercambio permitiendo que continuara la ejecución del proceso, incluso con ciertos segmentos intercambiados a disco, la memoria virtual lo hace aún más conveniente al aumentar la granularidad del intercambio:

ahora ya no se enviarán a disco secciones lógicas completas del proceso (segmentos), sino que se podrá reemplazar página por página, aumentando significativamente el rendimiento resultante. Al emplear la memoria virtual, de cierto modo la memoria física se vuelve sólo una proyección parcial de la memoria lógica, potencialmente mucho mayor a ésta.

Técnicamente, cuando se habla de memoria virtual, no se está haciendo referencia a un intercambiador (swapper), sino al paginador.

3.8.1 Paginación

La fragmentación externa y, por tanto, las necesidades de compactación pueden evitarse por completo empleando la paginación. Ésta consiste en que cada proceso está dividido en varios bloques de tamaño fijo (más pequeños que los segmentos) llamados páginas, dejando de requerir que la asignación sea de un área contigua de memoria. Claro está, esto requiere de mayor soporte por parte del hardware, y mayor información relacionada a cada uno de los procesos: no basta sólo con indicar dónde inicia y termina el área de memoria de cada proceso, sino que se debe establecer un mapeo entre la ubicación real (física) y la presentada a cada uno de los procesos (lógica). La memoria se presentará a cada proceso como si fuera de su uso exclusivo.

La memoria física se divide en una serie de marcos (frames), todos ellos del mismo tamaño, y el espacio para cada proceso se divide en una serie de páginas (pages), del mismo tamaño que los marcos. La MMU se encarga del mapeo entre páginas y marcos mediante tablas de páginas.

Cuando se trabaja bajo una arquitectura que maneja paginación, las direcciones que maneja el CPU ya no son presentadas de forma absoluta. Los bits de cada dirección se separan en un identificador de página y un desplazamiento, de forma similar a lo presentado al hablar de resolución de instrucciones en tiempo de ejecución. La principal diferencia con lo entonces abordado es que cada proceso tendrá ya no un único espacio en memoria, sino una multitud de páginas.

El tamaño de los marcos (y, por tanto, las páginas) debe ser una potencia de dos, de modo que la MMU pueda discernir fácilmente la porción de una dirección de memoria que se refiere a la página del desplazamiento. El rango varía, según el hardware, entre los 512 bytes (29) y 16 MB (224); al ser una potencia de dos, la MMU puede separar la dirección en memoria entre los primeros m bits (referentes a la página) y los últimos n bits (referentes al desplazamiento).

Para poder realizar este mapeo, la MMU requiere de una estructura de datos denominada tabla de páginas (page table), que resuelve la relación entre páginas y marcos, convirtiendo una dirección lógica (en el espacio del proceso) en la dirección física (la ubicación en que realmente se encuentra en la memoria del sistema).

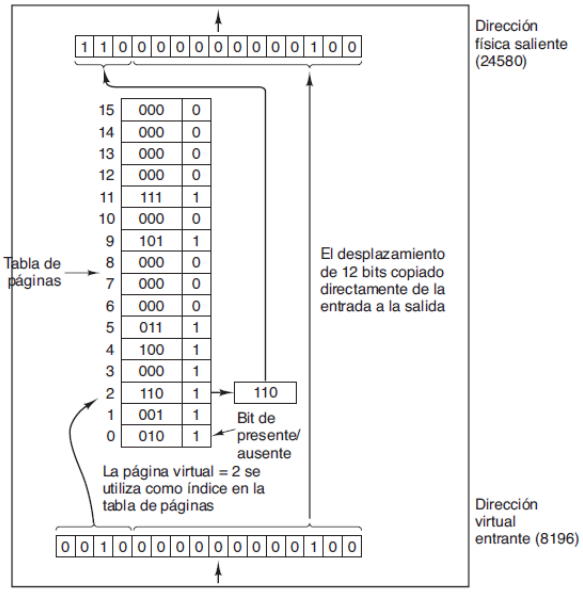

Figura 3.3: Modelo de paginación

3.8.1.1 Paginación por demanda

La memoria virtual entra en juego desde la carga misma del proceso. Se debe considerar que hay una gran cantidad de código durmiente o inalcanzable: aquel que sólo se emplea eventualmente, como el que responde ante una situación de excepción o el que se emplea sólo ante circunstancias particulares (por ejemplo, la exportación de un documento a determinados formatos, o la verificación de que no haya tareas pendientes antes de cerrar un programa). Y si bien a una computadora le sería imposible ejecutar código que no esté cargado en memoria, éste sí puede comenzar a ejecutarse sin estar completamente en memoria: basta con haber cargado la página donde están las instrucciones que permiten continuar con su ejecución actual.

La paginación sobre demanda significa que, para comenzar a ejecutar un proceso, el sistema operativo carga solamente la porción necesaria para comenzar la ejecución (posiblemente una página o ninguna), y que, a lo largo de la ejecución, el paginador es flojo: sólo carga a memoria las páginas cuando van a ser utilizadas. Al emplear un paginador flojo, las páginas que no sean requeridas nunca serán siquiera cargadas a memoria.

La estructura empleada por la MMU para implementar un paginador flojo es mediante una tabla de páginas que incluirá un bit de validez, indicando para cada página del proceso si está presente

o no en memoria. Si el proceso intenta emplear una página que esté marcada como no válida, esto causa un fallo de página, que lleva a que el sistema operativo lo suspenda y traiga a memoria la página solicitada para luego continuar con su ejecución:

1. Verifica en el PCB si esta solicitud corresponde a una página que ya ha sido asignada a este proceso.

2. En caso de que la referencia sea inválida, se termina el proceso.

3. Procede a traer la página del disco a la memoria. El primer paso es buscar un marco disponible (por ejemplo, por medio de una tabla de asignación de marcos).

4. Solicita al disco la lectura de la página en cuestión hacia el marco especificado.

Una vez que finaliza la lectura de disco, modifica tanto al PCB como al TLB para indicar que la tabla está en memoria.

6. Termina la suspensión del proceso, continuando con la instrucción que desencadenó al fallo. El proceso puede continuar sin notar que la página había sido intercambiada.

Llevando este proceso al extremo, se puede pensar en un sistema de paginación puramente sobre demanda (pure demand paging): en un sistema así, ninguna página llegará al espacio de un proceso si no es mediante de un fallo de página.

Un proceso, al iniciarse, comienza su ejecución sin ninguna página en memoria, y con el apuntador de siguiente instrucción del procesador apuntando a una dirección que no está en memoria (la dirección de la rutina de inicio). El sistema operativo responde cargando esta primera página, y conforme avanza el flujo del programa, el proceso irá ocupando el espacio real que empleará.

3.8.1.2 Políticas

3.8.1.2.1 De asignación de marcos de página

¿Cómo se asignan los marcos existentes a los procesos del sistema? Esto es, ¿qué esquemas se pueden definir para que la asignación inicial (y, de ser posible, en el transcurso de la ejecución) sea adecuada?

Por ejemplo, usando esquema sencillo: un sistema con 1 024 KB de memoria, compuesta de 256 páginas de 4096 bytes cada una, y basado en paginación puramente sobre demanda. Si el sistema operativo ocupa 248 KB, el primer paso será reservar las 62 páginas que éste requiere, y destinar las 194 páginas restantes para los procesos a ejecutar. Conforme se van lanzando y comienzan a ejecutar los procesos, cada vez que uno de ellos genere un fallo de página, se le irá asignando uno de los marcos disponibles hasta causar que la memoria entera esté ocupada. Claro está, cuando un proceso termine su ejecución, todos los marcos que tenía asignados volverán a la lista de marcos libres.

Una vez que la memoria esté completamente ocupada (esto es, que haya 194 páginas ocupadas por procesos), el siguiente fallo de página invocará a un algoritmo de reemplazo de página, que elegirá una de las 194.20

Este esquema, si bien es simple, al requerir una gran cantidad de fallos de página explícitos puede penalizar el rendimiento del sistema, el esquema puede resultar demasiado flojo, no le vendría mal ser un poco más ansioso y asignar, de inicio, un número determinado como mínimo utilizable de marcos**.**

3.8.1.2.2 Lectura

Están relacionadas con la decisión de cuándo se debe cargar una página en memoria principal. Con paginación por demanda, se trae una página a memoria principal sólo cuando se hace referencia a una posición en dicha página. Cuando un proceso se ejecute por primera vez, se presentarán muchos fallos de página. A medida que se traigan a memoria más páginas, el principio de cercanía hará que la mayoría de las futuras referencias estén en páginas que se han

cargado hace poco. Es decir, después de un tiempo, la situación se estabilizará y el número de fallos de páginas se reducirá considerablemente.

Con paginación previa se cargan adicionalmente otras páginas distintas de las demandadas debido a un fallo de página. El principal atractivo de esta estrategia es que la mayoría de los dispositivos de memoria secundaria, como los discos, tienen un tiempo de búsqueda y una latencia de giro. Es más eficiente traer a memoria un número de páginas contiguas que ir trayéndolas de una en una durante un período largo de tiempo.

La paginación previa no debe confundirse con el intercambia.

Cuando un proceso se descarga de memoria y pasa al estado suspendido, todo su conjunto residente se lleva también afuera. Cuando se reanuda el proceso, todas las páginas que estaban antes en memoria principal se devuelven a la misma.

3.8.1.2.3 De Ubicación

Tienen que ver con dónde va a residir una parte de un proceso en memoria principal. En un sistema de segmentación las posibles alternativas son: primer ajuste, siguiente ajuste y peor ajuste. Pero para un sistema que usa paginación o paginación combinada con segmentación, la ubicación carece de importancia puesto que el hardware de traducción de direcciones y el hardware de acceso a memoria principal pueden desarrollar sus funciones para cualquier combinación de marco de página con idéntica eficiencia.

3.8.1.2.4 De Reemplazo

Si se aprovechan las características de la memoria virtual para aumentar el grado de multiprogramación, se presenta un problema: al sobre-comprometer memoria, en determinado momento, los procesos que están en ejecución pueden caer en un patrón que requiera cargarse a memoria física páginas por un mayor uso de memoria que el que hay físicamente disponible.

Y si se tiene en cuenta que uno de los objetivos del sistema operativo es otorgar a los usuarios la ilusión de una computadora dedicada a sus procesos, no sería aceptable terminar la ejecución de un proceso ya aceptado y cuyos requisitos han sido aprobados, porque no hay suficiente

memoria. Se vuelve necesario encontrar una forma justa y adecuada de llevar a cabo un reemplazo de páginas que permita continuar satisfaciendo sus necesidades.

El reemplazo de páginas es una parte fundamental de la paginación, ya que es la pieza que posibilita una verdadera separación entre memoria lógica y física. El mecanismo básico a ejecutar es simple: si todos los marcos están ocupados, el sistema deberá encontrar una página que pueda liberar (una página víctima) y llevarla al espacio de intercambio en el disco. Luego, se puede emplear el espacio recién liberado para traer de vuelta la página requerida, y continuar con la ejecución del proceso.

Esto implica una doble transferencia al disco (una para grabar la página víctima y una para traer la página de reemplazo), y, por tanto, a una doble demora. Se puede, con un mínimo de burocracia adicional (aunque requiere de apoyo de la MMU): implementar un mecanismo que disminuya la probabilidad de tener que realizar esta doble transferencia: agregar un bit de modificación o bit de página sucia (dirty bit) a la tabla de páginas. Este bit se marca como apagado siempre que se carga una página a memoria, y es automáticamente encendido por hardware cuando se realiza un acceso de escritura a dicha página.

Cuando el sistema operativo elige una página víctima, si su bit de página sucia está encendido, es necesario grabarla al disco, pero si está apagado, se garantiza que la información en disco es idéntica a su copia en memoria, y permite ahorrar la mitad del tiempo de transferencia.

Ahora bien, ¿cómo decidir qué páginas reemplazar marcándolas como víctimas cuando hace falta? Para esto se debe implementar un algoritmo de reemplazo de páginas. La característica que se busca en este algoritmo es que, para un patrón de accesos dado, permita obtener el menor número de fallos de página.

De la misma forma como se realizó la descripción de los algoritmos de planificación de procesos, para analizar los algoritmos de reemplazo se usará una cadena de referencia, esto es, una lista de referencias a memoria. Estas cadenas modelan el comportamiento de un conjunto de procesos en el sistema, y, obviamente, diferentes comportamientos llevarán a resultados distintos.

Hacer un volcado y trazado de ejecución en un sistema real puede dar una enorme cantidad de información, del orden de un millón de accesos por segundo. Para reducir esta información en un

número más tratable, se puede simplificar basado en que no interesa cada referencia a una dirección de memoria, sino cada referencia a una página diferente.

Además, varios accesos a direcciones de memoria en la misma página no causan efecto en el estado. Se puede tomar como un sólo acceso a todos aquellos que ocurren de forma consecutiva (esto es, sin llamar a ninguna otra página, no es necesario que sean en instrucciones consecutivas) a una misma página.

Para analizar a un algoritmo de reemplazo, si se busca la cantidad de fallos de página producidos, además de la cadena de referencia, es necesario conocer la cantidad de páginas y marcos del sistema que se está modelando

3.8.1.2.5 gestión del conjunto residente

•**Ubicación fija:** se da al proceso un número fijo de página en las que ejecutarse; cuando ocurre un fallo de página, una de las páginas de ese proceso se debe reemplazar.

•**Ubicación variable**: el número de páginas asignadas a un proceso varía durante el tiempo de vida del proceso.

3.8.1.2.6 de vaciado

Son lo contrario de una política de lectura; se preocupan por determinar el momento en que hay que escribir en memoria secundaria una página modificada. Es posible dos formas de hacerlo:

- **Por demanda:** se escribirá sólo cuando haya sido elegida para reemplazarse.
- **Vaciado previo:** escribe las páginas antes de que se necesiten sus marcos, de forma que las páginas puedan escribirse por lotes.

3.8.1.2.7 control de carga

Aun con el mejor algoritmo de reemplazo de páginas y una asignación global óptima de marcos de página a los procesos, puede ocurrir que el sistema se sobrepagine. De hecho, cada vez que los conjuntos de trabajo combinados de todos los procesos exceden a la capacidad de la memoria, se puede esperar la sobrepaginación. Un síntoma de esta situación es que el algoritmo PFF indica que algunos procesos necesitan más memoria, pero ningún proceso necesita menos memoria. En

este caso no hay forma de proporcionar más memoria a esos procesos que la necesitan sin lastimar a algún otro proceso. La única solución real es deshacerse temporalmente de algunos procesos.

Una buena forma de reducir el número de procesos que compiten por la memoria es intercambiar algunos de ellos enviándolos al disco y liberar todas las páginas que ellos mantienen. Por ejemplo, un proceso puede intercambiarse al disco y sus marcos de página dividirse entre otros procesos que están sobrepaginando. Si el sobrepaginado se detiene, el sistema puede operar de esta forma por un tiempo. Si no se detiene hay que intercambiar otro proceso y así en lo sucesivo hasta que se detenga el sobrepaginado. Por ende, incluso hasta con la paginación se sigue necesitando el intercambio, sólo que ahora se utiliza para reducir la demanda potencial de memoria, en vez de reclamar páginas.

El proceso de intercambiar procesos para liberar la carga en la memoria es semejante a la planificación de dos niveles, donde ciertos procesos se colocan en disco y se utiliza un planificador de corto plazo para planificar los procesos restantes. Sin duda se pueden combinar las dos ideas donde se intercambien, fuera de memoria, sólo los procesos suficientes para hacer que la proporción de fallo de páginas sea aceptable. Periódicamente, ciertos procesos se traen del disco y otros se intercambian hacia el mismo.

Sin embargo, otro factor a considerar es el grado de multiprogramación. Cuando el número de procesos en la memoria principal es demasiado bajo, la CPU puede estar inactiva durante largos periodos.

Esta consideración sostiene que no sólo se debe tomar en cuenta el tamaño del proceso y la proporción de paginación al decidir qué proceso se debe intercambiar, sino también sus características, tal como si está ligado a la CPU o a la E/S, así como las características que tienen los procesos restantes.

3.81.3 Hiperpaginación

Es un fenómeno que se puede presentar por varias razones: cuando (bajo un esquema de reemplazo local) un proceso tiene asignadas pocas páginas para llevar a cabo su trabajo, y genera fallos de página con tal frecuencia que le imposibilita realizar trabajo real. Bajo un esquema de

reemplazo global, cuando hay demasiados procesos en ejecución en el sistema y los constantes fallos y reemplazos hacen imposible a todos los procesos involucrados avanzar, también se presenta hiperpaginación.

Hay varios escenarios que pueden desencadenar la hiperpaginación, y su efecto es tan claro e identificable que prácticamente cualquier usuario de cómputo lo sabrá reconocer. A continuación, se presentará un escenario ejemplo en que las malas decisiones del sistema operativo pueden conducirlo a este estado.

Suponga un sistema que está con una carga media normal, con un esquema de reemplazo global de marcos. Se lanza un nuevo proceso, que como parte de su inicialización requiere poblar diversas estructuras a lo largo de su espacio de memoria virtual. Para hacerlo, lanza una serie de fallos de página, a las que el sistema operativo responde reemplazando a varios marcos pertenecientes a otros procesos.

Casualmente, a lo largo del periodo que toma esta inicialización (que puede parecer una eternidad: el disco es entre miles y millones de veces más lento que la memoria) algunos de estos procesos solicitan los espacios de memoria que acaban de ser enviados a disco, por lo cual lanzan nuevos fallos de página.

Cuando el sistema operativo detecta que la utilización del procesador decrece, puede aprovechar la situación para lanzar procesos de mantenimiento. Se lanzan estos procesos, reduciendo aún más el espacio de memoria física disponible para cada uno de los procesos preexistentes.

Se ha formado ya toda una cola de solicitudes de paginación, algunas veces contradictorias. El procesador tiene que comenzar a ejecutar NOOP (esto es, no tiene trabajo que ejecutar), porque la mayor parte del tiempo lo pasa en espera de un nuevo marco por parte del disco duro. El sistema completo avanza cada vez más lento.

Los síntomas de la hiperpaginación son muy claros, y no son difíciles de detectar. ¿Qué estrategia puede emplear el sistema operativo una vez que se da cuenta que se presentó esta situación?

Una salida sería reducir el nivel de multiprogramación, si la paginación se presentó debido a que los requisitos de memoria de los procesos actualmente en ejecución no pueden ser satisfechos con

la memoria física disponible, el sistema puede seleccionar uno (o más) de los procesos y suspenderlos por completo hasta que el sistema vuelva a un estado normal. Podría seleccionarse, por ejemplo, al proceso con menor prioridad, al que esté causando más cantidad de fallos, o al que esté ocupando más memoria.

3.8.2 Segmentación

Un programa de usuario se puede dividir utilizando segmentación, en la cual el programa y sus datos ahocicados se dividen en un número de segmentos. No se requiere que todos los programas sean de la misma longitud, aunque hay una longitud máxima de segmento. Como en el caso de la paginación, una dirección lógica utilizando segmentación está compuesta por dos partes, en este caso un número de segmento y un desplazamiento.

Debido al uso de segmentos de distinto tamaño, la segmentación es similar al particionamiento dinámico. En la ausencia de un esquema overlays o el uso de la memoria virtual, se necesitaría que todos los segmentos de un programa se cargaran en la memoria para su ejecución. La diferencia, comparada con el particionamiento dinámico, es que con la segmentación un programa podría ocupar más de una partición, y estas particiones no necesitan ser contiguas. La segmentación elimina la fragmentación interna, pero, al igual que el particionamiento dinámico, sufre de fragmentación externa. Sin embargo, debido a que el proceso se divide en varias piezas más pequeñas, la fragmentación externa debería ser menor.

En la segmentación normalmente el programador o compilador asignara programas y datos a diferentes segmentos de distinto tamaños. El inconveniente principal de este servicio es que el programador debe ser consciente de la limitación de tamaño de segmento máximo.

Otra consecuencia de utilizar segmentos de distintos tamaños es que no hay una relación simple entre direcciones lógicas y direcciones físicas. Un esquema de segmentación sencillo haría uso de una tabla de segmentos por cada proceso y una lista de bloques libre de memoria principal. Cada entrada de la tabla de segmentos tendría que proporcionar la dirección inicial de la memoria principal del correspondiente segmento. La entrada también debería proporcionar la longitud del segmento, para asegurar que no se utilizan direcciones no válidas. Cuando un proceso entra en

estado ejecutando, la dirección de su tabla de segmentos se carga en un registro especial utilizado por el hardware de gestión de memoria.

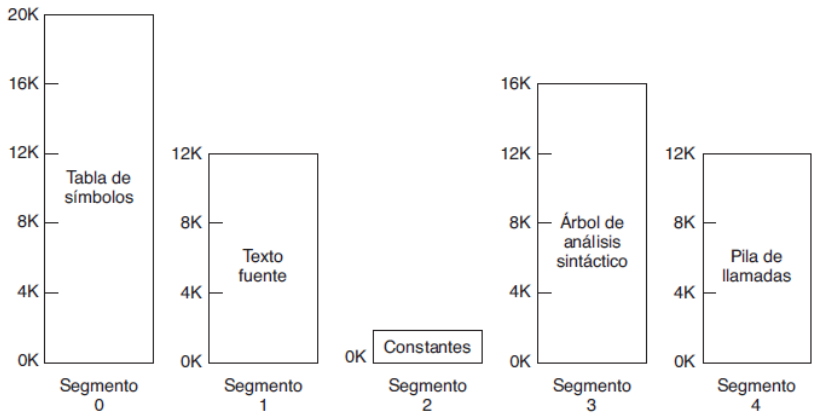

Figura 3.4: Una memoria segmentada permite que cada tabla crezca o se reduzca de manera independiente a las otras tablas.

3.8.2.1 Segmentación por demanda

Para esto es necesario: Hardware de segmentación que indique si el segmento está presente en memoria o no de manera que:

1. Si se referencia un segmento que está en memoria: se accede normalmente
2. Se referencia un segmento que no está en memoria se produce una excepción.

- Fallo del segmento:
- ✓ Se mira si hay espacio suficiente en memoria, y si no lo hay se hace una compactación
- ✓ Si no hay espacio en memoria se toma el primer segmento de la lista y si es necesario lo escribe en el intercambio.
- ✓ Si ahora hay espacio suficiente, se carga el segmento en memoria, se actualiza la tabla de segmentos y se pone al final de la lista.

3.8.3 Segmentación y Paginación combinada

Tanto la paginación como la segmentación tienen sus ventajas. La paginación, que es transparente al programador, elimina la fragmentación externa y, de este modo, aprovecha la memoria principal de forma eficiente. Además, puesto que los fragmentos que se cargan y descargan de memoria principal son de tamaño constante e igual para todos, es posible construir

algoritmos de gestión de memoria sofisticados que aprovechen mejor el comportamiento de los programas, tal y como se verá. La segmentación, que es visible para el programador, tiene las ventajas antes citadas, incluida la capacidad de manejar estructuras de datos que puedan crecer, el modularidad y el soporte de la compartición y la protección. Para combinar las ventajas de ambas, algunos sistemas están equipados con hardware del procesador y software del sistema operativo que las permiten.

En un sistema con paginación y segmentación combinadas, el espacio de direcciones de un usuario se divide en varios segmentos según el criterio del programador. Cada segmento se vuelve a dividir en varias páginas de tamaño fijo, que tienen la misma longitud que un marco de memoria principal. Si el segmento tiene menor longitud que la página, el segmento ocupará sólo una página. Desde el punto de vista del programador, una dirección lógica también está formada por un número de segmento y un desplazamiento en el segmento. Desde el punto de vista del sistema, el desplazamiento del segmento se ve como un número de página dentro del segmento y un desplazamiento dentro de la página.

La figura 3.5 propone una estructura para soportar la combinación de paginación y segmentación. Asociada con cada proceso existe una tabla de segmentos y varias tablas de páginas, una por cada segmento del proceso. Cuando un proceso determinado está ejecutándose, un registro contendrá la dirección de comienzo de la tabla de segmentos para ese proceso. Dada una dirección virtual, el procesador emplea la parte de número de segmento como índice en la tabla de segmentos del proceso para encontrar la tabla de páginas de dicho segmento. Entonces, la parte de número de página de la dirección virtual se usará como índice en la tabla de páginas para localizar el número de marco correspondiente. Este se combina con la parte de desplazamiento de la dirección virtual para generar la dirección real deseada.

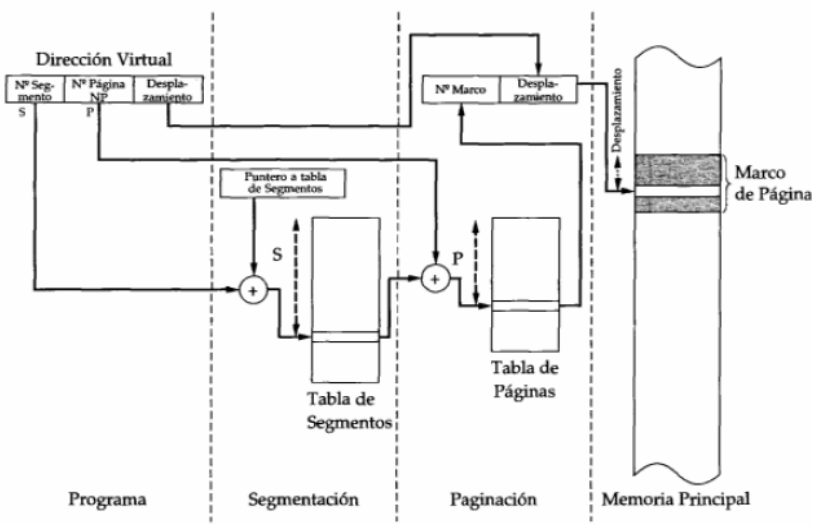

Figura 3.5 Estructura para soportar la combinación de paginación y segmentación.

La figura propone los formatos de la entrada de la tabla de segmentos y de la entrada de la tabla de páginas. Como antes, la entrada de la tabla de segmentos contiene la longitud del segmento. También contiene un campo base, que ahora se refiere a una tabla de páginas. Los bits de presencia y modificación no son necesarios, puesto que estos elementos se manejan a nivel de página. Pueden usarse otros bits de control para compartición y protección. La entrada de la tabla de páginas es, básicamente, la misma que se usa en un sistema de paginación pura. Cada número de página se convierte en el número de marco correspondiente si la página está presente en memoria. El bit de modificación indica si se necesita escribir la página a disco cuando se asigna el marco a otra página. Además, puede haber otros bits de control para ocuparse de la protección y de otros aspectos de la gestión de memoria.

3.8 Gestión de memoria en Linux

Linux usa una estructura de tablas de páginas de tres niveles, consiste en los siguientes tipos de tablas (cada tabla en particular tiene el tamaño de una página):

- **Directorio de páginas:** un proceso activo tiene un directorio de páginas único que tiene el tamaño de una página. Cada entrada en el directorio de páginas apunta a una página en el directorio intermedio de páginas. El directorio de páginas debe residir en la memoria principal para todo proceso activo.

- **Directorio intermedio de páginas:** este se expande a múltiples páginas Cada entrada en el directorio intermedio paginas apunta a una página que contiene una tabla de páginas.
- **Tabla de páginas:** la tabla también se puede expandir a múltiples páginas. Cada entrada de la tabla de páginas hace referencia a una página virtual del proceso.

La estructura de la tabla de páginas de Linux es independiente de plataforma y se diseñó para acomodarse al procesador Alpha de 64 bits, que proporciona soporte hardware para tres niveles de páginas. La arquitectura Pentium/x86 de 32 bits tiene un hardware de paginación de dos niveles. El software de Linux se acomoda al esquema de dos niveles definiendo el tamaño del directorio intermedio de páginas como 1. Todas las referencias a ese nivel extra de indirección se eliminan en la optimización realizada en el momento de la compilación, no durante la ejecución. Por tanto, no hay sobrecarga de rendimiento por la utilización de un diseño genérico de tres niveles en plataformas que solo soportan dos niveles.

Para mejorar la eficiencia de la lectura y escritura de páginas en la memoria principal, Linux define un mecanismo para manejar bloques de páginas contiguas que se proyectaran sobre bloques de marcos de páginas también contiguos. El núcleo mantiene una lista de marcos de páginas contiguos por grupos de un tamaño fijo. A lo largo del uso, las páginas se asignan y liberan de la memoria principal, los grupos disponibles se dividen y se juntan utilizando un algoritmo llamado buddy.

La gestión de la memoria del núcleo se realiza en base a los marcos de páginas de la memoria principal. Su función básica es asignar y liberar marcos para los diferentes usos. Los posibles propietarios de un marco incluyen procesos en espacio de usuario, datos del núcleo reservados dinámicamente, código estático del núcleo, y la cache de páginas.

Los fundamentos de la reserva de memoria de núcleo para Linux son los mecanismos de reserva de páginas ya usados para la gestión de la memoria virtual del usuario. Debido a que el tamaño mínimo de memoria que se pueda reservar de esta forma es una página, la reserva de páginas únicamente sería insuficiente debido a que el núcleo requiere pequeños fragmentos que se utilizarán durante un corto periodo de tiempo y que son de diferentes tamaños. Para ajustarse a estos pequeños tamaños sutiliza un esquema conocido como asignación de láminas dentro de una página ya reservada. Esto es, que Linux mantiene un

conjunto de listas enlazadas, una para cada tamaño del fragmento. Todos los fragmentos se pueden dividir, agregar o mover entre las diferentes listas de la forma correspondiente.

3.9 Gestión de memoria en Windows

El gestor de memoria virtual en Windows controla la forma en la que se reserva la memoria y cómo se realiza la paginación. El gestor de memoria se ha diseñado para funcionar sobre una variada gama de plataformas.

Cada proceso de usuario en Windows puede ver un espacio de direcciones independiente de 32 bits, permitiendo 4 Gbytes de memoria por proceso. Existe opciones que permiten aumentar el espacio de direcciones, esta característica se incluyó para dar soporte a aplicaciones que requieren un uso intensivo de grandes cantidades de memoria en servidores con memorias de bajos gigabytes, en los cuales un espacio de direcciones mayor puede mejorar drásticamente el rendimiento de aplicaciones de soporte como la decisión o data minig.

Una página se puede encontrar, a efectos de gestión, en los presentes estados:

- **Disponible:** páginas que no están actualmente usadas por este proceso.
- **Reservada:** conjunto de páginas contiguas que el gestor de memoria virtual separa para un proceso pero que no se cuentan para la cuota de memoria usada por dicho proceso. Cuando un proceso necesite escribir en la memoria, parte de esta memoria reservada se asigna al proceso.
- **Asignada:** las páginas para las cuales el gestor de la memoria virtual ha reservado espacio en el fichero de paginación (por ejemplo, el fichero de disco donde se escribirían las páginas cuando se eliminen de la memoria principal).

La distinción entre la memoria reservada y asignada es muy útil debido a que minimiza la cantidad de espacio de disco que debe guardarse para un proceso en particular, manteniendo espacio libre en disco para otros procesos; y permite que un hilo o un proceso declare una petición de una cantidad de memoria que puede proporcionarse rápidamente si se necesita.

El esquema de gestión del conjunto residente que utiliza Windows es de asignación variable, con ámbito local. Cuando se activa un proceso por primera vez, se le asigna un cierto número

de marcos de páginas de la memoria principal como conjunto de trabajo. Cuando un proceso hace referencia a una página que no está en la memoria, una de las páginas residentes de dicho proceso se expulsa, trayéndose la nueva página. Los conjuntos de trabajo de los procesos activos se ajustan usando las siguientes condiciones:

- Cuando hay memoria principal disponible, el gestor de la memoria virtual permite que los conjuntos residentes de los procesos activos crezcan. Para hacer esto, cuando ocurre un fallo página, se trae la nueva página a la memoria sin expulsar una página antigua, haciendo que se incremente el conjunto residente de proceso en una página.
- Cuando la memoria empieza escasear, el gestor de la memoria virtual recupera la memoria de sistema moviendo las páginas que se han utilizado hace más tiempo de cada uno de los procesos hacia swap, reduciendo el tamaño de esos conjuntos residentes.

BIBLIOGRAFÍA

- Stallings William. (2006). Sistemas Operativos Aspectos Internos y Principios de Diseño (Quinta Edición). (pág. 872). Pearson Educación, S.A., Madrid.
- Tanenbaum S. Andrews. (2009). Sistemas Operativos Modernos. (tercera edición). (pág. 1104). Pearson Educación, México.
- Meza E., Ruiz E., Wolf G. (2015). Fundamentos de Sistemas Operativos. Universidad Nacional Autónoma de México. México.
- Stallings William. (1997). Sistemas Operativos. (segunda edición). (pág. 732). Prentice Hall. Madrid
- Martínez David. (2001). Sistemas Operativos. (pág. 926). Universidad Nacional del Nordeste. Argentina.

IV. SISTEMA DE ARCHIVOS

Objetivos:

- Describir el sistema de archivos y su forma de organización.
- Conocer la estructura de directorios en un sistema de archivos.
- Analizar las técnicas empleadas en un servidor de archivos.
- Describir el manejo de archivos en los sistemas operativos actuales

¿De qué trata esta sesión de aprendizaje?

En esta sesión de aprendizaje se presentan las características generales de los archivos; su forma de organización y las técnicas empleadas por los servidores de archivos para lograr una eficiente gestión en el procesamiento de éstos.

4.1 Archivos

4.1.1 Concepto de archivo

Los **archivos** son unidades lógicas de información creada por los procesos. En general, un disco contiene miles o incluso millones de archivos independientes. De hecho, si concibe a cada archivo como un tipo de espacio de direcciones, no estará tan alejado de la verdad, excepto porque se utilizan para modelar el disco en vez de modelar la RAM.

Los procesos pueden leer los archivos existentes y crear otros si es necesario. La información que se almacena en los archivos debe ser **persistente**, es decir, no debe ser afectada por la creación y terminación de los procesos. Un archivo debe desaparecer sólo cuando su propietario lo remueve de manera explícita. Aunque las operaciones para leer y escribir en archivos son las más comunes, existen muchas otras.

"Un archivo es un conjunto de registros relacionados". Tanenbaum.

4.1.2 Nombres de archivos

Los archivos son un mecanismo de abstracción. Proporcionan una manera de almacenar información en el disco y leerla después. Esto se debe hacer de tal forma que se proteja al usuario de los detalles acerca de cómo y dónde se almacena la información y cómo funcionan los discos en realidad.

Probablemente, la característica más importante de cualquier mecanismo de abstracción sea la manera en que los objetos administrados son denominados.

Cuando un proceso crea un archivo le proporciona un nombre. Cuando el proceso termina, el archivo continúa existiendo y puede ser utilizado por otros procesos mediante su nombre.

Las reglas exactas para denominar archivos varían un poco de un sistema a otro, pero todos los sistemas operativos actuales permiten cadenas de una a ocho letras como nombres de archivos legales.

Por ende, *andrea*, *bruce* y *cathy* son posibles nombres de archivos. Con frecuencia también se permiten dígitos y caracteres especiales, por lo que nombres como *2*, *urgente!* y *Fig.2-14* son a menudo válido también. Muchos sistemas de archivos admiten nombres de hasta 255 caracteres.

Muchos sistemas operativos aceptan nombres de archivos en dos partes, separadas por un punto, como en *prog.c*. La parte que va después del punto se conoce como la **extensión del archivo** y por lo general indica algo acerca de su naturaleza. Por ejemplo, en MS-DOS, los nombres de archivos son de 1 a 8 caracteres, más una extensión opcional de 1 a 3 caracteres. En UNIX el tamaño de la extensión (si la hay) es a elección del usuario y un archivo puede incluso tener dos o más extensiones, como en *paginainicio.html.zip*, donde *.html* indica una página Web en HTML y *.zip* indica que el archivo se ha comprimido mediante el programa *zip.*

4.1.3 Estructura de un archivo

Los archivos se pueden estructurar en una de varias formas. Tres posibilidades comunes se describen en la figura 4.1. El archivo en la figura 4.1(a) es una secuencia de bytes sin estructura: el sistema operativo no sabe, ni le importa, qué hay en el archivo. Todo lo que ve son bytes. Cualquier significado debe ser impuesto por los programas a nivel usuario. Tanto Linux como Windows utilizan esta metodología.

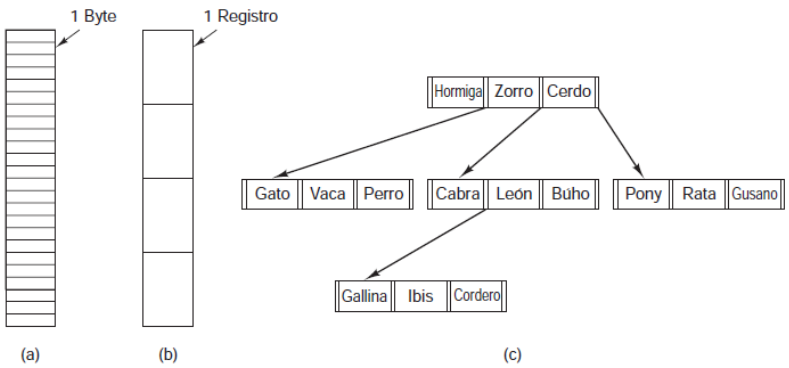

Figura 4.1: Estructuras de archivos.

Hacer que el sistema operativo considere los archivos sólo como secuencias de bytes provee la máxima flexibilidad. Los programas de usuario pueden colocar cualquier cosa que quieran en sus archivos y denominarlos de cualquier manera conveniente. El sistema operativo no ayuda, pero tampoco estorba. Para los usuarios que desean realizar cosas inusuales, esto último puede ser muy importante. Todas las versiones de Linux, MS-DOS y Windows utilizan este modelo de archivos.

La primera configuración en la estructura se muestra en la figura 4.1(b). En este modelo, un archivo es una secuencia de registros de longitud fija, cada uno con cierta estructura interna. El concepto central para la idea de que un archivo sea una secuencia de registros es la idea de que la operación de lectura devuelva un registro y la operación de escritura sobrescriba o agregue un registro.

Como nota histórica, hace algunas décadas, cuando reinaba la tarjeta perforada de 80 columnas, muchos sistemas operativos de mainframes basaban sus sistemas de archivos en archivos consistentes de registros de 80 caracteres; es decir, en imágenes de la tarjeta. Estos sistemas también admitían archivos con registros de 132 caracteres, que fueron destinados para la impresora de línea (que en esos días eran grandes impresoras de cadena con 132 columnas). Los programas leían la entrada en unidades de 80 caracteres y la escribían en unidades de 132 caracteres, aunque los últimos 52 podían ser espacios, desde luego. Ningún sistema de propósito general de la actualidad utiliza ya este modelo como su sistema de archivos primario, pero en aquellos días de las tarjetas perforadas de 80 columnas y del papel de impresora de línea de 132 caracteres, éste era un modelo común en las computadoras mainframe.

El tercer tipo de estructura de archivo se muestra en la figura 4.1(c). En esta organización, un archivo consiste de un árbol de registros, donde no todos son necesariamente de la misma longitud; cada uno de ellos contiene un campo **llave** en una posición fija dentro del registro. El árbol se ordena con base en el campo llave para permitir una búsqueda rápida por una llave específica.

La operación básica aquí no es obtener el "siguiente" registro, aunque eso también es posible, sino obtener el registro con una llave específica. Para el archivo del zoológico de la figura 4.0(c), podríamos pedir al sistema que, por ejemplo, obtenga el registro cuya llave sea *pony*, sin preocuparnos acerca de su posición exacta en el archivo. Además, se pueden agregar nuevos registros al archivo, con el sistema operativo, y no el usuario, decidiendo dónde colocarlos. Evidentemente, este tipo de archivos es bastante distinto de los flujos de bytes sin estructura que se usan en Linux y Windows, pero se utiliza de manera amplia en las grandes computadoras mainframe que aún se emplean en algún procesamiento de datos comerciales.

4.1.4 Métodos de acceso

Los primeros sistemas operativos proporcionaban sólo un tipo de acceso: **acceso secuencial**. En estos sistemas, un proceso podía leer todos los bytes o registros en un archivo en orden, empezando desde el principio, pero no podía saltar algunos y leerlos fuera de orden. Sin embargo, los archivos secuenciales podían rebobinarse para poder leerlos todas las veces que fuera necesario. Los archivos secuenciales eran convenientes cuando el medio de almacenamiento era cinta magnética en vez de disco.

Cuando se empezó a usar discos para almacenar archivos, se hizo posible leer los bytes o registros de un archivo fuera de orden, pudiendo acceder a los registros por llave en vez de posición.

Los archivos cuyos bytes o registros se pueden leer en cualquier orden se llaman **archivos de acceso aleatorio**. Son requeridos por muchas aplicaciones.

Los archivos de acceso aleatorio son esenciales para muchas aplicaciones, como los sistemas de bases de datos. Si el cliente de una aerolínea llama y desea reservar un asiento en un vuelo específico, el programa de reservación debe poder tener acceso al registro para ese vuelo sin tener que leer primero los miles de registros de otros vuelos.

Es posible utilizar dos métodos para especificar dónde se debe empezar a leer. En el primero, cada operación read da la posición en el archivo en la que se va a empezar a leer. En el segundo se provee una operación especial (seek) para establecer la posición actual. Después de una operación seek, el archivo se puede leer de manera secuencial desde la posición actual. Este último método se utiliza en Linux y Windows.

4.1.5 Semánticas de coutilización

Cualquier forma de acceso tiene problemas cuando varios usuarios trabajan con el archivo simultáneamente. La semántica de coutilización especifica el efecto de varios procesos accediendo de forma simultánea al mismo archivo y cuando se hacen efectivas las modificaciones.

Tipos de semánticas:

- **Semántica UNIX (POSIX):** Las escrituras son inmediatamente visibles para todos los procesos con el archivo abierto. Los procesos pueden compartir archivos. Si existe relación de parentesco pueden compartir el puntero. La coutilización afecta también a los metadatos. Costoso computacionalmente. SO ha de secuenciar accesos.
- **Semántica de sesión**: Las escrituras que hace un proceso no son inmediatamente visibles para los demás procesos con el archivo abierto. Cuando se cierra el archivo los cambios se hacen visibles para las futuras sesiones. Un archivo puede asociarse temporalmente a varias imágenes. Se hace necesario sincronizar los procesos explícitamente.
- **Semántica de versiones**: las actualizaciones se hacen sobre copias con nº versión. Sólo son visibles cuando se consolidan versiones. Sincronización explícita si se requiere actualización inmediata.
- **Semántica de archivos inmutables:** una vez creado el archivo sólo puede ser compartido para lectura y no cambia nunca. Implementado con técnica tipo copy-on-write. Atributo de lectura por bloque.

4.1.6 Comportamiento de archivos

Podemos pensar en un archivo como en un tipo abstracto de datos, al que se pueden aplicar las siguientes operaciones: Crear, Abrir, Cerrar, Leer, Escribir, Añadir, Buscar, Obtener Atributos, Cambiar de nombre.

4.2 Directorios

Asociado con cualquier sistema de gestión de archivos o cualquier colección de archivos suele haber un directorio de archivos. El directorio contiene información sobre los archivos, incluyendo atributos, ubicación y propietario. Gran parte de esta información, especialmente la relativa al almacenamiento, la gestiona el sistema operativo.

4.2.1 Concepto de directorio

EL directorio es propiamente un archivo, poseído por el sistema operativo y accesible a través de diversas rutinas de gestión de archivos. Aunque parte de la información de los directorios está disponible para los usuarios y aplicaciones, en general, la información se proporciona indirectamente, a través de rutinas del sistema.

De este modo, los usuarios no pueden acceder directamente al directorio, incluso en modo de sólo lectura.

4.2.2 Estructuras de directorio

Antes de acceder a un archivo, éste tiene que ser abierto. En la apertura, el S.O. carga en memoria central la información que permite localizar los bloques de disco asignados al archivo.

Los archivos se organizan en directorios, siendo un directorio una tabla o archivo que contiene información acerca de los archivos contenidos en el mismo.

Ejemplos de estructuras de directorios:

- **MS-DOS**: Los directorios son archivos que contienen un número arbitrario de entradas. Cada entrada tiene 32 bytes repartidos de la siguiente forma: nombre del archivo (8 bytes); extensión del archivo (3 bytes); atributos (archivo de lectura, oculto, del sistema, etiqueta de volumen, subdirectorio, ...1 byte); reservado (10 bytes); hora (2 bytes); fecha (2 bytes); nº del primer bloque se usa como índice de la FAT para localizar los demás bloques de disco (2 bytes); tamaño en bytes (4 bytes).

- **UNIX :** Los directorios son archivos y en cada entrada al directorio, que tiene un tamaño de 16 bytes, se almacena la siguiente información: Nº del nodo-i(2 bytes) ; Nombre del archivo(14 bytes).

  Toda la información del tipo de archivo, tamaño, propietario, bloques de disco, ... está contenida en el nodo-i.

4.2.3 Nombres jerárquicos

Una posibilidad es que el directorio contenga por cada archivo referenciado.

- El nombre.
- Sus atributos.
- Las direcciones en disco donde se almacenan los datos.

Otra posibilidad es que cada entrada del directorio contenga.

- El nombre del archivo.
- Un apuntador a otra estructura de datos donde se encuentran los atributos y las direcciones en disco.

Al abrir un archivo el S.O:

- Busca en su directorio el nombre del archivo.
- Extrae los atributos y direcciones en disco.
- Graba esta información en una tabla de memoria real.
- Todas las referencias subsecuentes al archivo utilizarán la información de la memoria principal.

El número y organización de directorios varía de sistema en sistema:

- **Directorio único:** el sistema tiene un solo directorio con todos los archivos de todos los usuarios.
- **Un directorio por usuario:** el sistema habilita un solo directorio por cada usuario.
- Un árbol de directorios por usuario: el sistema permite que cada usuario tenga tantos directorios como necesite, respetando una jerarquía general.

4.2.4 Construcción de la jerarquía de directorios

El directorio jerárquico o estructurado en árbol consiste en un directorio maestro que contiene un número determinado de directorios de usuario. Cada uno de estos directorios puede tener a su vez subdirectorios y archivos como entradas. Esto se cumple en cualquier nivel. Es decir, en cualquier nivel, un directorio puede constar de entradas para subdirectorios y/o entradas para archivos. Queda comentar cómo se organiza cada directorio y subdirectorio. El método más simple es, por supuesto, almacenar cada directorio como un archivo secuencial. Cuando los directorios contengan un número muy grande de entradas, tal organización puede conducir a tiempos de búsqueda innecesariamente grandes. En tal caso, se prefiere una estructura de dispersión.

Los usuarios deben poder referirse a un archivo por medio de un nombre simbólico. Evidentemente, cada

archivo del sistema debe tener un nombre único para que las referencias al archivo no sean ambiguas. Por otra parte, proporcionar nombres únicos es una carga inaceptable para los usuarios, especialmente en un sistema compartido.

El uso de directorios estructurados en árbol minimiza la dificultad de asignar nombres únicos.

Cualquier archivo del sistema puede ser localizado siguiendo un camino desde el directorio raíz o maestro, descendiendo por varias ramas hasta que se alcance el archivo.

La serie de nombres de directorios, terminados con el propio nombre del archivo, constituye el nombre de camino del archivo. Es perfectamente aceptable tener varios archivos con el mismo nombre de archivo mientras tengan nombres de camino únicos.

Aunque el nombre de camino facilita la elección de los nombres de archivo, para un usuario sería incómodo tener que deletrear el nombre de camino entero cada vez que haga una referencia a un archivo.

Normalmente, cada usuario interactivo o proceso tiene asociado un directorio actual, conocido como directorio de trabajo. Las referencias a los archivos son entonces relativas al directorio de trabajo.

Cuando un usuario interactivo se conecte o cuando se cree un proceso, el valor por defecto para el directorio de trabajo será el directorio del usuario. Durante la ejecución, el usuario puede navegar por árbol y así definir directorios de trabajo diferentes.

4.3 Estructura y Almacenamiento del archivo y del directorio

El sistema de archivos debe llevar registro no sólo de los bloques de disco libres sino también de los bloques asignados a cada archivo. Un archivo se divide en bloques de tamaño fijo que se almacenan en bloques de disco no necesariamente contiguos.

El sistema de archivos debe registrar la dirección de disco donde se encuentra almacenado cada bloque lógico de cada archivo.

Algunas estrategias posibles son las que se indican a continuación:

- **Lista encadenada de bloques:** En la entrada del directorio correspondiente al archivo, se guarda la dirección del primer bloque de disco. Cada bloque contiene (al final) la dirección del siguiente bloque de la cadena o una marca de fin de archivo (EOF).
- **Tabla de asignación de archivos (FAT):** A cada disco se asocia una tabla (FAT) con una entrada por cada bloque de disco. Cada entrada de la FAT puede contener: una dirección de disco, una marca de bloque libre (indicativo de que el bloque no está asignado), una marca de bloque defectuoso o una marca de fin de archivo (EOF).
- **Bloques de índices (nodos-índices):** A diferencia de la FAT, se guardan las listas de bloques de los diferentes archivos en estructuras de datos independientes. Esta estrategia es la que se utiliza en el S.O. UNIX, donde a cada archivo se le asigna una tabla llamada nodo índice (nodo-i) que contiene la siguiente información: tipo de nodo, número de enlaces al archivo, identificación de usuario y de grupo del propietario, tamaño del archivo, fecha/hora de creación, del último acceso y de la última modificación, las 10 primeras direcciones de bloques de disco, y punteros indirectos simple, doble y triple.

Para un archivo que ocupe menos de 10 bloques, no son necesarios los punteros indirectos, puesto que en el nodo-i se registran las direcciones de disco de todos los bloques de datos. Si el archivo crece por encima de los 10 bloques, se adquiere un bloque de disco (bloque indirecto simple) al cual apunta el P.I.S. Dicho bloque contiene direcciones de disco de bloques de datos.

4.4 Sistema de Archivos y el servidor de archivos

El sistema de archivos permite a los usuarios crear colecciones de datos, llamados archivos, con propiedades deseables, tales como:

- **Existencia a largo plazo:** los archivos se almacenan en disco u otro almacenamiento secundario y no desaparece cuando un usuario se desconecta.
- **Compartible entre procesos:** los archivos tienen nombres y pueden tener permisos de acceso asociados que permitan controlar la compartición.
- **Estructura:** dependiendo del sistema de archivo, un archivo puede tener una estructura interna que es conveniente para aplicaciones particulares

El servidor de archivos es aquel conjunto de software de sistema que proporciona servicios a los usuarios y aplicaciones en el uso de archivos. Sus principales objetivos son:

- Satisfacer las necesidades de gestión de datos y requisitos del usuario, lo que incluye el almacenamiento de datos y la capacidad de llevar a cabo las operaciones requeridas.
- Garantizar, hasta donde sea posible, que los datos del fichero son válidos.
- Optimizar el rendimiento, desde el punto de vista del sistema en términos de productividad y desde el punto de vista del usuario en términos de tiempo de respuesta.
- Proporcionar soporte de e/s a una variedad de tipos de dispositivos de almacenamiento.
- Minimizar o eliminar la potencial pérdida de datos.
- Proporcionar un conjunto estándar de rutinas de interfaces de e/s a los procesos.
- Dar soporte de e/s a múltiples usuarios, en el caso de sistemas multiusuarios.

4.5 Servicios de archivos y directorios

Un servicio de directorio es una aplicación o un conjunto de aplicaciones que almacena y organiza la información sobre los usuarios de una red de ordenadores y sobre los recursos de red que permite a los administradores gestionar el acceso de usuarios a los recursos sobre dicha red. Además, los servicios de directorio actúan como una capa de abstracción entre los usuarios y los recursos compartidos.

Un servicio de directorio no debería confundirse con el repositorio de directorio, que es la base de datos la que contiene la información sobre los objetos de nombrado gestionada por el servicio de

directorio. En el caso del modelo de servicio de directorio distribuido en X.500, se usa uno o más espacios de nombre (árbol de objetos) para formar el servicio de directorio. El servicio de directorio proporciona la interfaz de acceso a los datos que se contienen en unos o más espacios de nombre de directorio. La interfaz del servicio de directorio es la encargada de gestionar la autenticación de los accesos al servicio de forma segura, actuando como autoridad central para el acceso a los recursos de sistema que manejan los datos del directorio.

Como base de datos, un servicio del directorio está altamente optimizado para lecturas y proporciona alternativas avanzadas de búsqueda en los diferentes atributos que se puedan asociar a los objetos de un directorio. Los datos que se almacenan en el directorio son definidos por un esquema extensible y modificable. Los servicios de directorio utilizan un modelo distribuido para almacenar su información y esa información generalmente está replicada entre los servidores que forman el directorio

4.6 Gestión de archivos en Linux

Linux incluye un versátil y potente sistema de gestión de archivos, diseñado para soportar una gran variedad de sistemas de gestión de archivos y estructuras de archivos; esto lo hace utilizando un sistema de archivos virtual (VFS), que presenta una única y uniforme interfaz de sistema de archivos para los procesos de usuario. Define un modelo de archivos común que es capaz de representar cualquier característica general y comportamiento de un sistema de archivo concebible. VFS asume que los ficheros son objetos de un sistema de almacenamiento masivo del computador que comparte propiedades básicas sin tener en cuenta el sistema de archivo concreto o el hardware subyacente. Para cualquier sistema de archivo específico, se necesita un módulo de proyección que transforme las características del sistema de archivo real a las características esperadas por el sistema de ficheros virtual.

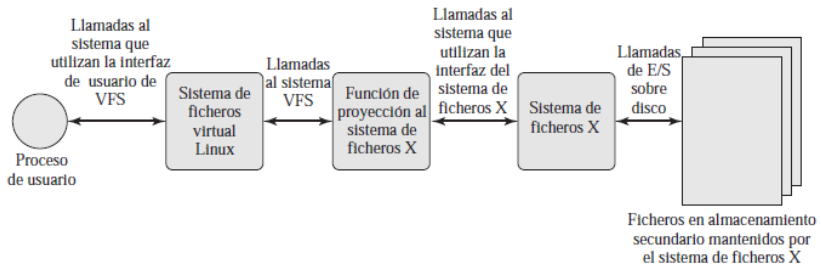

Figura 4.2: Sistema de archivo virtual de Linux

Como se muestra en la figura 4.2, cuando un proceso inicia una llamada al sistema orientada a archivos (por ejemplo, lectura), el núcleo llama a una función de VFS. Esta función gestiona los aspectos independientes del sistema de archivos e inicia una llamada a una función en el código del sistema de archivos destino. Esta llamada pasa a través de una función de proyección debe ser parte de la implementación de un sistema de archivos en Linux. El sistema de archivo destino convierte la petición del sistema de ficheros en instrucciones orientadas a dispositivo que se pasa al controlador del dispositivo mediante funciones cache de páginas.

4.7 Gestión de archivos en Windows

Windows utiliza un sistema de gestión de archivos llamado NTFS, que está pensado para alcanzar requisitos de altas prestaciones en estaciones de trabajo y servidores.

Entre sus características están.

- **Recuperación:** capacidad de recuperarse frente a errores en el sistema; es capaz de reconstruir volúmenes de disco y devolverlos a un estado consistente.
- **Seguridad:** utiliza un modelo de objetos para forzar la seguridad, un archivo abierto se implementa como un objeto archivo con un descriptor de seguridad que define sus atributos de seguridad.
- **Discos y ficheros grandes:** soporta discos y archivos grandes de forma más eficiente en comparación con el resto.
- **Múltiples flujos de datos:** los contenidos se tratan como un flujo de bytes y es posible definir múltiples flujos de datos para un único archivo.
- **Facilidad general de indexación**: asocia una colección de atributos con cada archivo y se organiza como una base de datos relacional de forma que los archivos se pueden indexar por cualquier atributo.

BIBLIOGRAFÍA

- J. Carretero, F. García, P. de Miguel, F. Pérez. Sistemas operativos: una visión aplicada"
- Stallings William. (2006). Sistemas Operativos Aspectos Internos y Principios de Diseño (Quinta Edición). (pág. 872). Pearson Educación, S.A., Madrid.
- Tanenbaum S. Andrews. (2009). Sistemas Operativos Modernos. (tercera edición). (pág. 1104). Pearson Educación, México.
- Martínez David. (2001). Sistemas Operativos. (pág. 926). Universidad Nacional del Nordeste. Argentina.
- Deitel, H.M. Sistemas Operativos. Segunda edición. Editorial Addison Wesley.

V. GESTIÓN DE ENTRADA/SALIDA

Objetivos:

- Describir la caracterización de los dispositivos de E/S.
- Conocer la organización del sistema de E/S.
- Identificar las interfaces de aplicación.
- Señalar el funcionamiento del almacenamiento secundario y terciario.
- Describir el manejo y uso del reloj y la terminal.
- Analizar el manejo de entrada y salida en los sistemas operativos actuales

¿De qué trata esta sesión de aprendizaje?

En esta sesión de aprendizaje se describirán y conocerán las características del sistema de E/S, así como también los dispositivos de almacenamiento secundario y terciario Se identifican los elementos básicos como las interfaces para las aplicaciones; y el modo en el que el software de E/S ejecuta las peticiones emitidas por el usuario y el sistema operativo.

5.1 Introducción

Una de las funciones principales de un S.0. es el control de todos los dispositivos de e/s de la computadora.

Las principales funciones relacionadas son:

- Enviar comandos a los dispositivos.
- Detectar las interrupciones.
- Controlar los errores.
- Proporcionar una interfaz entre dispositivos y el resto del sistema; la cual debe ser sencilla y fácil de usar y debe ser misma para todos los dispositivos.

El código de e/s representa una fracción significativa del S.O. Y el uso inapropiado de los dispositivos de e/s frecuentemente genera ineficiencias del sistema, lo que afecta el desarrollo de tareas.

5.2 Caracterización de los dispositivos de E/S

A medida que los sistemas informáticos han evolucionado, se ha producido una tendencia creciente en la complejidad y satisfacción de cada componente individual. En ningún punto es

más evidente que en la función de entrada/salida. En cada etapa se destacan diferentes características que se pueden resumirse como sigue:

- Se añade un controlador o módulo de entrada/salida programada sin interrupciones. El CPU se aísla de los detalles específicos de las interfaces en dispositivos externos.
- Se emplean interrupciones. Ahora el procesador no tiene que desperdiciar tiempo esperando a que se realice una operación de entrada/salida, incrementando así la eficiencia.
- El módulo de entrada/salida recibe el control directo de la memoria a través de DMA (Direct Access Memory). Ahora puede mover un bloque de datos de la memoria o desde la misma sin que intervenga el procesador, excepto al principio y al final de la transferencia.
- Se mejora el módulo de entrada/salida hasta llegar a ser un procesador separado con un conjunto de instrucciones especializadas para entrada/salida. El CPU ordena al procesador de entrada/salida la ejecución de un programa de E/S en la memoria principal. El procesador de entrada/salida lee y ejecuta estas instrucciones sin intervención del procesador. Esto permite al procesador especificar una secuencia de actividades de e/s e interrumpirla sólo cuando haya terminado la secuencia entera.
- El módulo de entrada/salida posee su propia memoria local y es, de hecho, un computador independiente. Con esta arquitectura se pueden controlar un gran número de dispositivos de entrada/salida, con una participación mínima del procesador.

A medida que se sigue en esta evolución, una mayor parte de las funciones de entrada/salida se realiza sin la participación del procesador. El procesador central se ve liberado cada vez más de las tareas relacionadas con la entrada/salida, mejorando así el rendimiento.

5.2.1 Conexión de un dispositivo de E/S a una computadora

Los dispositivos de e/s se conectan a la CPU a través de grupos de hilos que se conoce como buses. En el interior del computador el bus transmite la información de los datos en paralelo. El bus que conecta la CPU con los otros elementos del procesador se conoce como bus local o bus de la CPU. Es un bus muy rápido y conecta la CPU con las tarjetas de la placa base y los

controladores de los dispositivos externos. Las conexiones entre los periféricos y los controladores o tarjetas de la placa base se realizan a través de un bus más general llamado bus del sistema. También suele conectar algunas ampliaciones de memoria. Algunos dispositivos requieren un bus especializado que se adapte a su velocidad de transferencia, sus niveles de tensión, la naturaleza de sus señales de control y otros requerimientos. A estos buses se les llama bus de entrada/salida o bus de expansión. Por todo lo vistos, los computadores grandes al disponer de varios tipos de buses requieren de dispositivos adaptadores o de interconexión entre buses. Los procesadores suelen tener unas ranuras de expansión (6 normalmente) sobre la placa base que están conectadas al bus del sistema y que permiten conectar una serie de dispositivos a este bus a través de tarjetas de circuito integrado y que permiten conectar varios dispositivos a la CPU, como por ejemplo tarjetas digitalizadoras de imágenes, aceleradores gráficos con FPGAs, etc. Todos los buses poseen unas especificaciones normalizadas, como son: - protocolos de transmisión de datos, velocidades y temporización de las transferencias, anchuras de los sub-buses, - y sistema físico de conexión (conectores estandarizados).

5.2.2 Dispositivos conectados por puertos o proyectados en memoria

Para comenzar una operación de e/s, la cpu tiene que escribir sobre los registros anteriores los datos de la operación a través de una dirección de e/s o de memoria asignada únicamente al controlador. Según se haga de una u otra forma, se distingue entre dispositivos conectados por puertos o proyectados en memoria.

- **Dispositivo por puertos:**

  En este modelo cunado se instala un dispositivo, a su controlador se le asigna un puerto e/s, una interrupción de hardware y un vector de interrupción.

  Para efectuar una operación de e/s la cpu ejecuta operaciones por el puerto de salida con la dirección del puerto del dispositivo y con parámetros para indicar que registro se quiere manipular.

- **Dispositivo proyectado en memoria:**

  Este método asigna a cada dispositivo de e/s un rango de direcciones de memoria a través de las cuales se escribe sobre los registros del controlador. En este modelo no hay instrucciones específicas de e/s, sino que las operaciones se llevan a cabo mediante instrucciones máquina de manejo de memoria, lo que permite gestionar un mapa único de

direcciones de memoria. Sin embargo, para no tener conflictos con otros accesos a la memoria y optimizar las operaciones, se reserva una zona de memoria física para asignar las direcciones de e/s.

5.2.3Dispositivos de bloque y caracteres

Las principales características de los dispositivos por bloque son:

- La información se almacena en bloques de tamaño fijo.
- Cada bloque tiene su propia dirección.
- Los tamaños más comunes de los bloques van desde los 128 bytes hasta los 1024.
- Se puede leer o escribir en un bloque de forma independiente de los demás, en cualquier momento.
- Un ejemplo de dispositivos de bloque son los discos.

Las principales características de los dispositivos de carácter son:

- La información se transfiere como un flujo de caracteres, sin sujetarse a una estructura de bloques.
- No se pueden utilizar direcciones.
- No tiene una operación de búsqueda.
- Un ejemplo de este tipo de dispositivo son las impresoras de línea, terminales, interfaces de una red, ratones, etc.

5.2.4 E/S programada o por interrupciones

- **E/S programada:** El procesador emite una orden de e/s de parte de un proceso a un módulo de e/s; el proceso espera entonces a que termine la operación, antes de seguir.
- **E/S por interrupciones:** el procesador emite una orden de e/s de parte de un proceso, continúa la ejecución de las instrucciones siguientes y es interrumpido por el módulo de e/s cuando este ha completado su trabajo. Las instrucciones siguientes pueden ser del mismo proceso, sino es necesario para este esperar la terminación de la e/s. En otro caso, el proceso se ve suspendido a la espera de la interrupción, mientras se realiza otro trabajo.

5.2.5 Mecanismos de incremento de prestaciones

Los mecanismos de incremento de prestaciones se presentan de las siguientes maneras:

- **Acceso Directo a Memoria (DMA, Direct Memory Access):**

  Cuando se utiliza el método de Acceso Directo a Memoria, es el controlador el que se encarga directamente de transferir los datos entre el periférico y la memoria principal, sin requerir intervención alguna por parte del procesador. Esto funciona de la siguiente manera: cuando el procesador desea un bloque de datos, envía una orden al controlador indicándole la siguiente información:

- ✓ Tipo de Operación: Lectura o escritura.
- ✓ Periférico involucrado en la operación.
- ✓ La dirección de memoria desde la que se va a leer o a la que va a escribir directamente el controlador de dispositivo (dirección).
- ✓ El número de bytes a transferir.

Una vez emitida la orden el procesador continúa realizando otro trabajo sin necesidad de transferir el bloque de datos. Es el propio controlador el que se encarga de transferir el bloque de datos del periférico a memoria.

Los pasos que sigue el método en una operación de e/s con DMA son los siguientes:

- ✓ Programación de la operación de E/S. Se indica al controlador la operación, los datos a transferir y la dirección de memoria sobre la que se efectuara la operación.
- ✓ El controlador contesta aceptando la petición de E/S.
- ✓ El controlador le ordena al dispositivo que lea (para operación de lectura) una cierta cantidad de datos desde una posición determinada del dispositivo a su memoria interna.
- ✓ Cuando los datos están listos, el controlador los copia a la posición de memoria que tiene en sus registros, incrementa dicha posición de memoria y decrementa el contador de datos pendientes de transferir.
- ✓ Los pasos 3 y 4 se repiten hasta que no quedan más datos por leer.
- ✓ Cuando el registro de contador está en cero, el controlador interrumpe a la UCP para indicar que la operación DMA ha terminado.

- **Caches de Disco en el Controlador**

  La idea de este método es aprovechar la memoria interna de los controladores para leer los datos por adelantado, evitando muchas operaciones de búsqueda en el disco y sobre todo los tiempos de latencia necesarios para esperar a que los datos pasen de nuevo bajo las cabezas del disco.

  La proximidad espacial permite optimizar la E/S en el ámbito de controlador, ya que, en lugar de leer un sector, o un grupo de ellos, se leen pistas enteras en cada vuelta de disco, lo que permite traer múltiples bloques de datos en una única operación. En los canales de E/S, donde suele haber mucha memoria interna, se guardan en memoria varias pistas por cada dispositivo E/S.

  Estos mecanismos permiten optimizar mucho la E/S, especialmente en operaciones de lectura con un comportamiento conocido. Para evitar afectar el rendimiento de las operaciones que no responden a patrones de proximidad espacial predecibles, los controladores incluyen instrucciones para desactivar este mecanismo, siempre que el sistema operativo lo crea conveniente.

Solapamiento de búsquedas y transferencias

Los controladores de disco actuales permiten la conexión de varios dispositivos de E/S y tiene un canal de comunicaciones con ellos de varios MB.

Para optimizar el uso del conjunto de los dispositivos, muchos controladores actuales programan las operaciones de búsqueda en los dispositivos y mientras reciben la respuesta transfieren datos de otros dispositivos listos para leer o escribir. De esta forma existe paralelismo real entre los dispositivos, lo que permite explotar al máximo el canal de comunicaciones.

5.3 Arquitectura del sistema de E/S

El sistema de entrada y salida está construido como un conjunto de manejadores apilados, cada uno de los cuales está asociado a un dispositivo de entrada/salida (archivos, red, etc.).

Ofrece a las aplicaciones y entornos de ejecución servicios genéricos que permiten manejar los objetos de e/s del sistema. A través de ellos se puede acceder a todos los manejadores de archivos y de dispositivos tales como: discos, redes, consola, tarjetas de sonido, etc.

La arquitectura de e/s, es compleja y está estructurada en capas, cada una de las cuales tiene una funcionalidad bien definida.

- **Interfaz del sistema operativo para e/s**: proporciona servicios de e/s síncrona y asíncrona a las aplicaciones y una interfaz homogénea para poderse comunicar con los manejadores de dispositivo ocultando los detalles de bajo nivel a la vez pueden proporcionar una interfaz para que el usuario pueda comunicarse entre sí y controla los manejadores que no son comunes entre otros
- **Sistema de archivos:** proporcionan una interfaz homogénea, a través del sistema de archivos virtuales, para acceder a todos los sistemas de archivos que proporciona el sistema operativo (FFS, SV, NTFS, FAT, etc).
- **Gestor de redes:** proporciona una interfaz homogénea para acceder a todos los sistemas de red que proporciona el sistema operativo (TCP/IP, Novell, etc.). Además, permite acceder a los manejadores de cada tipo de red particular de forma transparente.
- **Gestor de bloques:** los sistemas de archivos y otros dispositivos lógicos con acceso a nivel de bloques se suelen limitar a traducir las operaciones del formato del usuario de bloques que entiende el dispositivo y se las pasan a este gestor de bloques.
- **Gestor de caché:** optimiza las e/s mediante la gestión de almacenamiento intermedio de memoria para dispositivos de tipo bloques. El tamaño de la cache de bloques varía dinámicamente en función de la memoria RAM disponible, y los bloques se escriben a los dispositivos según la política que se tenga definida.
- **Manejadores de dispositivo:** proporcionan operaciones de alto nivel sobre los dispositivos y las traducen en su ámbito interno a operaciones de control de cada dispositivo particular.

Cada uno de estos componentes se considera un objeto del sistema, por lo que habitualmente todos los sistemas operativos permiten modificar el sistema operativo de forma estática o dinámica para reemplazar, añadir o quitar manejadores de dispositivos. Sin embargo, por razones de seguridad no se permite a las aplicaciones de usuario acceder directamente a los dispositivos, sino a través de la interfaz de llamadas al sistema.

5.3.1 Estructura y componentes del sistema de E/S

El sistema de entrada y salida está construido como un conjunto de manejadores apilados, cada uno de los cuales está asociado a un dispositivo de entrada/salida (archivos, red, etc.).

Ofrece a las aplicaciones y entornos de ejecución servicios genéricos que permiten manejar los objetos de E/S del sistema. A través de ellos se puede acceder a todos los manejadores de archivos y de dispositivos tales como: discos, redes, consola, tarjetas de sonido, etc., Los dispositivos periféricos solicitan recursos del sistema por medio de interrupciones.

La arquitectura de E/S, es compleja y está estructurada en capas, cada una de las cuales tiene una funcionalidad bien definida.

- Interfaz del sistema operativo para E/S.
- Sistema de archivos.
- Gestor de redes.
- Gestor de bloques.
- Gestor de caché.
- Manejadores de dispositivo.

El sistema de E/S está compuesto por un sistema de almacenamiento temporal (caché), una interfaz de controladores de dispositivos y otra interfaz para dispositivos específicos.

El sistema operativo gestiona el almacenamiento temporal de entrada/salida y las interrupciones de los dispositivos de entrada/salida.

5.3.2 Software de E/S

El software de e/s está organizado como una serie de capas donde:

- Las capas inferiores se encarguen de ocultar las peculiaridades del hardware a las capas superiores.
- Las capas superiores deben presentar una interfaz agradable, limpia y regular a los usuarios.

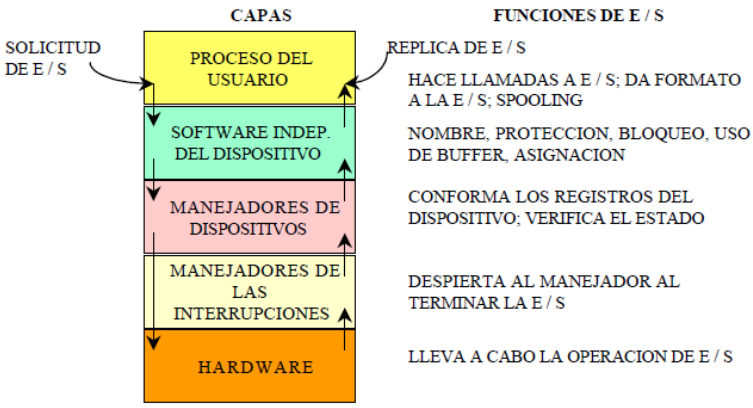

Figura 5.1: Capas del sistema de entrada/salida y las principales funciones de cada capa

El objetivo principal del software de e/s es la independencia del dispositivo, incluyendo las sigfuientes caracteristicas:

- Debe ser posible escribir programas que se puedan utilizar con archivos en distintos dispositivos, sin tener que modificar los programas para cada tipo de dispositivo.
- Todos los archivos y dispositivos adquieren direcciones de la misma forma, es decir mediante el nombre de su ruta de acceso.

Otro aspecto importante del software es el manejo de errores de e/s:

- Generalmente los errores deben manejarse lo más cerca posible del hardware.
- Solo si los niveles inferiores no pueden resolver el problema, se informa a los niveles superiores.
- En su mayoría, la recuperación se puede hacer en un nivel inferior y de forma transparente.

Otro aspecto clave son las transferencias síncronas (por bloques) o asíncronas (controladas por interruptores):

- La mayoría de la e/s es asíncrona: la cpu inicia la transferencia y realiza otras tareas hasta una interrupción.
- La programación es más fácil si la e/s es síncrona: el programa se suspende automáticamente hasta que los datos estén disponibles en el buffer.

Generalmente el software de e/s se estructura en capas:

- Manejadores de interrupciones.
- Directivas de dispositivos.

- Software de S.O. independiente de los dispositivos-
- Software a nivel usuario.

5.4 Mecanismos y funciones de los manejadores de dispositivos (device drivers)

Todo el código que depende de los dispositivos aparece en los manejadores de dispositivos. Cada controlador posee uno o más registros de dispositivos, los cuales se utilizan para darle los comandos y verificar su ejecución adecuada.

La labor principal para un manejador de dispositivo es la de aceptar las solicitudes abstractas que le hace el software independiente del dispositivo y a su vez verificarlas.

Si al recibir una solicitud el manejador está ocupado con otra solicitud, agregara la nueva solicitud a una cola de solicitudes pendientes.

5.5 Interfaz de aplicaciones

- Las llamadas de E/S encapsulan el comportamiento de los dispositivos en clases genéricas.
- La capa del manejador esconde las diferencias entre los controladores de E/S del núcleo.
- Los dispositivos pueden variar en muchas dimensiones:

•Nombres independientes de dispositivo.

•E/S bloqueante y no bloqueante.

•Control de acceso a dispositivos compartidos y dedicados.

•Indicaciones de error.

•Uso de estándares.

5.6 Almacenamiento secundario

El almacenamiento secundario es un medio de almacenamiento definitivo (no volátil como el de la memoria RAM, sino DVD, CDs, por ejemplo).

El proceso de transferencia de datos a un equipo de cómputo se le llama procedimiento de lectura. El proceso de transferencia de datos desde la computadora hacia el almacenamiento se denomina procedimiento de escritura.

En la actualidad se pueden usar principalmente dos tecnologías para almacenar información: Magnético (ej. disco duro, diskette); óptico. (ej. Algunos dispositivos combinan ambas tecnologías.

5.6.1 Discos

Los discos están organizados en cilindros, pistas y sectores. El número típico de sectores por vista varía entre 8 y 32 (o más); todos los sectores tienen igual número de bytes.
Entre sus principales ventajas están:

- Mucho mayor capacidad de espacio de almacenamiento.
- Menor precio por bit.
- La información no se pierde al apagar la computadora

5.6.2 El manejador de disco

Un manejador de disco es el conjunto de circuitos integrados que tienen como función organizar la lectura y escritura en las unidades de disco en una computadora. Este dispositivo envía la información que necesita la computadora para interpretar los comandos que se soliciten. Se utilizan con ambas unidades de disquetes y con los discos duros; en otros casos, está insertado en la placa madre.

Esta transferencia de información que recibe y transmite a la unidad de disco consiste en diversos comandos, basados en los caracteres de control ASCII. Hace la conversión entre los patrones magnéticos de la superficie del disco en movimiento y los bits del buffer del dispositivo; estos patrones indican acciones como mover el cabezal de lectura/escritura, controlar la transferencia de información y fungir de intermediario entre la unidad de disco y el microprocesador. También con esta información, el disco debe ser capaz de mover radialmente el brazo hacia dentro y hacia afuera sobre la superficie del disco.

Los controladores de disco más conocidos son el IDE, SATA, EIDE y SCSI.

5.6.3 Discos en memoria

Utilizan una parte de la memoria principal asignada con anterioridad para almacenar los bloques. Entre sus ventajas están:

- Acceso instantáneo.
- No hay demora rotacional o debida a las búsquedas.
- Son adecuados para el almacenamiento de programas o datos con acceso muy frecuentes.

Los bloques de almacenamiento tienen el mismo tamaño que en discos reales.

Cuando el manejador debe leer o escribir en un bloque de un disco en memoria, calcula el lugar de la memoria donde se encuentra el bloque solicitado y lee o escribe en él mismo.

5.6.4 Fiabilidad y tolerancia a fallos

La tolerancia a fallos determina la capacidad de un sistema de almacenamiento de acceder a información o al recurso aún en caso de producirse algún fallo. Esta falla puede deberse a daños físicos (mal funcionamiento) en uno o más componentes de hardware lo que produce la pérdida de información almacenada. La tolerancia a fallos requiere para su implementación que el sistema de almacenamiento guarde la misma información en más de un componente de hardware o en una máquina o dispositivo externos a modo de respaldo. De esta forma, si se produce alguna falla con una consecuente pérdida de datos, el sistema debe ser capaz de acceder a toda la información recuperando los datos faltantes desde algún respaldo disponible.

Los dispositivos de almacenamiento secundario son una buena opción que permite contrarrestar estos posibles fallos, y, o, a estar preparados en caso de que ocurran.

5.7 Almacenamiento terciario

El almacenamiento terciario o también llamado memoria terciaria es un sistema en el que un brazo robótico montará (conectará) o desmontará (desconectará) un medio de almacenamiento masivo fuera de línea según lo solicite el sistema operativo de la computadora.

La memoria terciaria se usa en el área del almacenamiento industrial, la computación científica en grandes sistemas informáticos y en redes empresariales.

5.7.1 Tecnología para almacenamiento terciario

Cuando una computadora debe leer la información del almacenamiento terciario, primero consultará un catálogo base de datos para determinar cuál cinta o el disco contiene la información. A continuación, el ordenador le instruirá un brazo robótico para buscar el medio y colocarlo en una unidad. Cuando el ordenador haya terminado de leer la información, el brazo robótico volverá el medio a su lugar en la biblioteca.

5.7.2 Estructura y componentes de un sistema de almacenamiento terciario

- Sistemas de almacenamiento removibles.
- Tiempos de acceso aún mayores, pero coste por GB menor.
- Empleados para almacenamiento “long term”.
- Cintas (desde los 50s) de acceso secuencial.
- Pueden almacenar centenares de petabytes (y hasta exabytes)

5.8 El reloj

Los relojes son esenciales para la operación de sistemas de tiempo compartido. Registran la hora del día y evitan que un proceso monopolice la cpu., entre otras cosas. El software de reloj puede tomar la forma de un software controlador de dispositivo, aun y cuando un reloj no es un dispositivo de bloque (como un disco) ni un dispositivo de carácter (como un ratón).

5.8.1 El hardware del reloj

Hay dos tipos de relojes de uso común en las computadoras, y ambos son bastante distintos de los relojes que utilizan las personas. Los relojes más simples están enlazados a la línea de energía de 110 o 220 voltios y producen una interrupción en cada ciclo de voltaje, a 50 o 60 Hz. Estos relojes solían dominar el mercado, pero ahora son raros.

El otro tipo de reloj se construye a partir de tres componentes: un oscilador de cristal, un contador y un registro contenedor. Cuando una pieza de cristal de cuarzo se corta en forma apropiada y se monta bajo tensión, puede generar una señal periódica con una precisión muy grande, por lo general en el rango de varios cientos de megahertz, dependiendo del cristal elegido. Mediante el uso de componentes electrónicos, esta señal base puede multiplicarse por un pequeño entero para

obtener frecuencias de hasta 1000 MHz o incluso más. Por lo menos uno de esos circuitos se encuentra comúnmente en cualquier computadora, el cual proporciona una señal de sincronización para los diversos circuitos de la misma. Esta señal se alimenta al contador para hacer que cuente en forma descendente hasta cero. Cuando el contador llega a cero, produce una interrupción de la CPU.

5.8.2 El software del reloj

Las principales funciones del software manejador del reloj son:

- Mantener la hora del día o tiempo real.
- Evitar que los procesos se ejecuten durante más tiempo del permitido.
- Llevar un registro del uso de la cpu.
- Controlar llamadas al sistema tipo "alarm" por parte de los procesos del usuario.
- Proporcionar cronómetros guardianes de partes del propio sistema.
- Realizar resúmenes, monitoreo y recolección de estadísticas.
- El software de reloj puede tener que simular varios relojes virtuales con un único reloj físico.

Por lo general, los relojes programables tienen varios modos de operación. En el **modo de una instancia**, cuando se inicia el reloj copia el valor del registro contenedor en el contador y después decrementa el contador en cada pulso del cristal. Cuando el contador llega a cero, produce una interrupción y se detiene hasta que vuelve a ser iniciado en forma explícita mediante el software.

En el **modo de onda cuadrada**, después de llegar a cero y producir la interrupción, el registro contenedor se copia automáticamente en el contador y todo el proceso se repite de nuevo en forma indefinida. Estas interrupciones periódicas se conocen como **pulsos de reloj**.

La ventaja del reloj programable es que su frecuencia de interrupción se puede controlar mediante software. Si se utiliza un cristal de 500 MHz, entonces se aplica un pulso al contador cada 2 nseg. Con registros de 32 bits (sin signo), se pueden programar interrupciones para que ocurran a intervalos de 2 nseg hasta 8.6 seg. Los chips de reloj programables por lo general contienen dos o tres relojes que pueden programarse de manera independiente, y tienen muchas otras opciones también (por ejemplo, contar en forma ascendente o descendente, deshabilitar interrupciones, y más).

5.9 La terminal

Son dispositivos electrónicos utilizados para ingresar datos, emitir resultados dentro de un sistema de cómputo.

Comparados con las tarjetas perforadas o las cintas de papel, los primeros terminales eran dispositivos baratos pero muy lentos para la entrada de datos, sin embargo, a medida que la tecnología mejoró, ya que fueron introducidas las pantallas de video, los terminales sacaron de la industria a estas viejas formas de interacción. Un desarrollo relacionado fueron los sistemas de tiempo compartido, que se desarrollaron en paralelo y compensaron cualquier ineficacia en la habilidad de mecanografiado del usuario con la capacidad de soportar a múltiples usuarios conectados a la misma máquina, cada uno de ellos con su propio terminal.

La función de un terminal está confinada a la exhibición y entrada de datos; un dispositivo con una significativa capacidad local programable de procesamiento de datos puede ser llamado un "terminal inteligente" o cliente pesado. Un terminal que depende del computador huésped para su capacidad de procesamiento es llamado cliente ligero. Un computador personal puede correr un software que emule la función de un terminal, permitiendo a veces el uso concurrente de programas locales y el acceso a un distante sistema huésped de terminal.

5.9.1 Modo de operación del terminal

- Exposición se centra en información alfanumérica.
- Modo de operación similar en todos los tipos de terminales.
- Entrada: código de tecla o caracteres ASCII(se tiene en cuenta teclas modificadoras Control, Alt, ...)
- Salida: pantalla (matriz de píxeles con memoria de vídeo asociada)
- Escritura en pantalla requiere escritura en memoria de vídeo.
- Secuencias de escape: operaciones especiales.

5.9.2 El hardware del terminal

- Teclado
- Pantalla
- Memoria de video

5.9.3 El software del terminal

- Dirigido por interrupciones (uso de int. software)
- Terminal proyectado: conversión a ASCII por manejador
- Software de salida más sencillo para terminales serie: hardware del terminal se encarga de todo el procesamiento.
- En terminales proyectados más trabajo para el manejador: copia/procesa caracteres, de proceso a memoria de vídeo
- Trata caracteres con presentación especial y secuencias escape.

5.10 E/S en Linux

El núcleo de Linux asocia un fichero especial con cada manejador de dispositivo de e/s, distinguiéndose entre dispositivos de bloques, de caracteres y de red.

Linux utiliza un planificador llamado ascensor, incluye también dos algoritmos adicionales: el planificador de e/s basado en plazos y el planificador de e/s previsor.

El planificador del ascensor mantiene una única cola con las peticiones de lectura y escritura en el disco, realizando operaciones de ordenamiento y agrupamiento sobre la cola, es decir, mantiene la lista de peticiones ordenadas por el número de bloque. De esta manera, cuando se manejan las peticiones de disco, el dispositivo se mueve en una única dirección satisfaciendo cada petición según la encuentra.

El planificador basado en plazos no intenta servir peticiones en un plazo de tiempo determinado, sino que deja de insertar las peticiones en orden después de un plazo conveniente.

Esto lo hace usando tres colas, cada nueva petición se incluye en la cola ordenada de ascensor, esa misma petición se sitúa al final de una cola FIFO de lectura en el caso de que sea así o de escritura; por tanto, las colas de lectura y escritura almacenan una lista de peticiones en el orden en qué estas se hicieron. Asociado con cada petición hay un tiempo de expiración, con un valor por defecto de 0,5 segundos en caso de una petición de lectura y de 5 segundos en el de una de escritura. Cuando se completa una petición, se elimina de la cabeza de la cola ordenada y también de la cola FIFO correspondiente. Sin embargo, cuando se cumple el tiempo de expiración de elemento de la cabeza de una de las colas FIFO, el planificador pasa a dar servicio de esa cola

FIFO, extrayendo la petición expirada, junto con algunas de las siguientes peticiones de la cola. Según se sirve cada petición, se borra de la cola ordenada.

En tanto, el planificador de e/s previsor está superpuesto sobre el planificador basado en plazos; cuando se sirve una petición de lectura, el planificador previsor causa que el sistema de planificación se retrase has 6 milisegundos, dependiendo de la configuración. Durante este pequeño retardo, hay una oportunidad apreciable de que la aplicación que solicitó la última petición de lectura genere otra petición de lectura en la misma región del disco. En caso de que sea así, esa petición se servirá inmediatamente. Sino se produce esa petición de lectura, el planificador continúa utilizando el algoritmo de planificación basado en plazos.

Viendo desde el punto de cache de páginas en Linux 2.4 en adelante, hay una única cache de páginas unificada que está involucrada en todo el tráfico entre el disco y la memoria principal.

La cache de páginas conlleva dos beneficios. El primero, cuando llega el momento de escribir en el disco las páginas modificadas, se puede agrupar un conjunto de las mismas ordenándolas adecuadamente y escribiéndolas, por tanto, eficientemente. En segundo lugar, gracias al principio de la proximidad temporal, las páginas incluidas en la cache de páginas se accederán probablemente de nuevo antes de ser expulsadas de la cache, evitando de esta una operación e/s de disco.

Las páginas modificadas se escriben en el disco en dos situaciones:

- Cuando la cantidad de memoria libre llega a ser menor que un determinado umbral, el núcleo reduce el tamaño de la cache de páginas liberando memoria que va a añadirse al conjunto de memoria libre disponible en el sistema.
- Cuando las páginas modificadas envejecen más allá de un determinado umbral, se escriben en disco varias páginas modificadas.

5.11 E/S en Windows

Windows cuenta con un gestor de e/s que es responsable de todo el sistema de e/s del sistema operativo y proporciona una interfaz uniforme a la que todos los tipos de manejadores pueden llamar.

- **Módulos de e/s básicos:**
  - **Gestor de cache:** maneja la gestión de la cache para todo el subsistema de e/s proporcionando un servicio de cache memoria principal para todos los componentes

del sistema de ficheros y de red. Se puede incrementar y decrementar dinámicamente el tamaño de la cache dedicada a una determinada actividad según varié la cantidad de la memoria física disponible.

- **Manejadores de sistemas de ficheros:** encamina los mensajes destinados a determinados volúmenes al manejador software correspondiente a ese adaptador de dispositivo.
- **Manejadores de red:** Windows incluye una gestión de red integrada y proporciona soporte para aplicaciones distribuidas.
- **Manejadores de dispositivos hardware:** estos manejadores acceden a los registros hardware de los dispositivos periféricos a través de puntos de entrada de bibliotecas dinámicamente enlazadas del Ejecutivo de Windows.

- **E/S síncrona y asíncrona**

  Windows ofrece dos modos de operación para la e/s, asíncrono y síncrono. El modo asíncrono se utiliza para optimizar el rendimiento de la aplicación, siempre que sea posible; una aplicación inicia una operación e/s prosiguiendo su ejecución mientras se lleva a cabo la petición de e/s. Con la e/s síncrona, la aplicación se bloquea hasta que se completa la operación de e/s.

  La e/s asíncrona necesita una manera de determinar cuándo se completa la operación, esto lo hace Windows proporcionando cuatro técnicas diferentes para la notificación de la finalización de la operación de e/s:

  - **Activación de un objeto dispositivo del núcleo:** se activa un indicador asociado a un objeto dispositivo cuando se completa una operación en ese objeto. Esta tecina es sencilla, pero no es apropiada para manejar múltiples peticiones.
  - **Activación de un objeto evento del núcleo:** permite que haya múltiples peticiones de e/s simultáneas sobre un dispositivo o fichero. El hilo crea un evento por cada petición, y puede esperar por una sola de estas peticiones o por todas ellas.
  - **E/S con alerta:** utiliza una cola asociada a un hilo, el hilo realiza peticiones de e/s y el gestor de e/s sitúa los resultados de estas peticiones en la cola del hilo solicitante.

- **Puertos de finalización de e/s:** se utiliza en un servidor de Windows para optimizar el uso de los hilos. Está disponible un conjunto de hilos para su uso, de manera que no es necesario crear un nuevo hilo para manejar una nueva petición.

BIBLIOGRAFÍA

- Llaven. Daniel S. (2015). Sistemas Operativos, Panorama para Ingeniería en Computación.(1ra Edición). Universidad Nacional Autónoma de México. Grupo Editorial Patria, S.A. de CV. México.
- Juan P.C., Juan M.M. (2002). Concepto de Sistemas Operativos. Universidad Pontificia Comillas. Amábar S.L. España.
- Stallings William. (2006). Sistemas Operativos Aspectos Internos y Principios de Diseño (Quinta Edición). (pág. 872). Pearson Educación, S.A., Madrid.
- Tanenbaum S. Andrews. (2009). Sistemas Operativos Modernos. (tercera edición). (pág. 1104). Pearson Educación, México.
- Martínez David. (2001). Sistemas Operativos. (pág. 926). Universidad Nacional del Nordeste. Argentina.
- Harvey M. Deitel. Sistemas Operativos. 2da. Edición. Editorial Addison-Wesley.