Link del chatbot de esto: https://chatgpt.com/g/g-6848e953cbf48191a0c53b8540934271-so-master-en-teoria

# Unidad 1: Parte 1

**Resumen del documento: so clase 01 - Introducción**

**Unidad I – Introducción, Historia y Conceptos de los Sistemas Operativos**

**Computadora Actual**

En términos generales, una computadora moderna es considerada un sistema complejo que integra diversos componentes, tanto físicos como lógicos, para su funcionamiento.

**Ubicación del Sistema Operativo**

El sistema operativo se encuentra en una posición clave en la arquitectura informática, actuando como intermediario entre el hardware de la máquina y el resto del software.

**Modo Kernel/Supervisor y Modo Usuario**

Las computadoras contemporáneas operan generalmente en dos modos distintos:

* **Modo Kernel (Supervisor)**: Permite al sistema operativo tener acceso completo y privilegiado al hardware, pudiendo ejecutar cualquier instrucción que soporte el equipo.
* **Modo Usuario**: Aquí se ejecuta el resto del software con restricciones, pudiendo realizar solo operaciones permitidas y limitadas en comparación con el modo kernel.

**Definiciones del Concepto de Sistema Operativo**

* **Stallings (2005)**: Define el sistema operativo como un programa encargado del control en la ejecución de aplicaciones y programas, sirviendo como interfaz entre estos y el hardware.
* **Silva (2010)**: Entiende al sistema operativo como un sistema formado por elementos relacionados entre sí que funcionan coordinadamente para operar o conducir al hardware.
* **Tanenbaum (2009)**: Considera que los sistemas operativos cumplen dos funciones fundamentales: proporcionar abstracciones sencillas a programadores y administrar recursos hardware complejos.

**Sistema Operativo como Máquina Extendida y Administrador de Recursos**

Tanenbaum (2009) plantea dos perspectivas claves sobre el rol del sistema operativo:

* **Máquina Extendida**: El SO proporciona abstracciones sobre el hardware complejo, facilitando la tarea de programación. Ejemplo concreto es el manejo de archivos (crear, abrir, leer/escribir y cerrar).
* **Administrador de Recursos**: El SO asigna de manera ordenada y controlada recursos tales como procesadores, memorias y dispositivos de entrada/salida, a múltiples programas que compiten por ellos. El manejo de recursos se realiza mediante multiplexaje:  
  + **Multiplexaje en tiempo**: Varios programas utilizan secuencialmente un recurso, como la CPU.
  + **Multiplexaje en espacio**: Se asigna a cada programa una parte específica del recurso, como ocurre con la memoria.

**Historia de los Sistemas Operativos**

**Generaciones de Computadoras y Sistemas Operativos:**

* **Prehistoria**:  
  + Charles Babbage (1792-1871): Diseñó la primera computadora digital, que carecía de sistema operativo.
* **Primera Generación (1945-1955)**:  
  + Aparición de computadoras digitales con tubos al vacío, sin lenguajes ni sistemas operativos.
* **Segunda Generación (1955-1965)**:  
  + Nacimiento del transistor.
  + Surgen equipos mainframe y lenguajes como FORTRAN y ensamblador.
  + Aparecen sistemas operativos primitivos como FMS (Fortran Monitor System) e IBSYS de IBM.
* **Tercera Generación (1965-1980)**:  
  + Aparición de circuitos integrados.
  + IBM 360 introduce la multiprogramación, permitiendo compartir memoria entre programas.
  + Implementación del concepto de spooling (acumulación de tareas en cola).
  + Desarrollo del primer sistema de tiempo compartido de propósito general.
  + Crecimiento de minicomputadoras, como DEC PDP-1 (1961).
* **Cuarta Generación (1980-actualidad)**:  
  + Surge la computadora personal (microcomputadora).
  + Aparece el sistema operativo CP/M basado en disco.
  + IBM lanza IBM PC; Bill Gates adquiere DOS (Disk Operating System).
  + En 1983, IBM PC/AT con CPU Intel 80286 populariza MS-DOS.

**Bibliografía citada en el documento:**

* Tanenbaum, A. S. (2009). *Sistemas operativos modernos* (3ª ed.). México: Pearson Educación.
* Stallings, W. (2005). *Sistemas operativos* (5ª ed.). Madrid: Pearson Educación.

# Unidad 1: Parte 2

**Repaso del Hardware de una Computadora**

El SO está estrechamente relacionado con el hardware, extendiendo instrucciones y gestionando recursos. El hardware básico incluye:

* **Procesadores**: La CPU ejecuta instrucciones en ciclos continuos (obtener, decodificar y ejecutar). Modernamente se utiliza la canalización para ejecutar múltiples instrucciones simultáneamente.
* **Memoria**: Está organizada en jerarquías según velocidad, capacidad y costo (caché, RAM, ROM, EEPROM, Flash y CMOS).
* **Discos y cintas**: Discos (acceso más lento pero económico y grande) y cintas magnéticas (respaldo para grandes cantidades de datos).
* **Dispositivos de E/S y Buses**: Incluyen controladores y medios para transferir datos entre componentes.
* **Arranque**: El BIOS verifica hardware básico y carga el sistema operativo desde el dispositivo activo.

**Tipos de Sistemas Operativos**

* **Mainframes**: Sistemas de gran capacidad y E/S (OS/390, S/360, UNIX, Linux).
* **Servidores**: Brindan servicios compartidos a través de redes (Solaris, FreeBSD, Linux, Windows Server).
* **Multiprocesadores**: Gestionan sistemas con múltiples CPUs o núcleos.
* **Personales**: Orientados a un usuario único (Linux, Windows, Macintosh).
* **De Bolsillo (PDAs)**: Compactos y funcionales (Palm OS, Symbian, iOS, Android).
* **Integrados**: Controlan dispositivos específicos (Linux embebido, QNX, VxWorks).
* **Nodos Sensores**: Redes inalámbricas pequeñas para monitoreo remoto (TinyOS).
* **Tiempo Real**: Operan bajo restricciones temporales estrictas (eCos, FreeRTOS).
* **Tarjetas Inteligentes**: Limitados en recursos y funciones específicas (Java Virtual Machine).

**Bibliografía citada:**

* Tanenbaum, A. S. (2009). *Sistemas operativos modernos* (3ª ed.). México: Pearson Educación.
* Stallings, W. (2005). *Sistemas operativos* (5ª ed.). Madrid: Pearson Educación.

# Unidad 1: Clase 3

**Procesos**

Un proceso es un programa en ejecución, asociado con un espacio de direcciones que contiene ubicaciones de memoria específicas que puede leer y escribir. Cuando se suspende un proceso temporalmente, el sistema operativo debe guardar el estado exacto del proceso en la tabla de procesos, que incluye todos los datos necesarios para reanudar su ejecución posteriormente. Los procesos pueden crear procesos hijos, generando así estructuras jerárquicas.

**Memoria y Espacios de Direcciones**

La memoria principal almacena los programas en ejecución. Los sistemas operativos avanzados permiten múltiples programas simultáneamente mediante técnicas como la memoria virtual, que permite manejar partes del espacio de direcciones en memoria principal y otras en disco. Esta técnica desacopla el espacio lógico del proceso de la memoria física.

**Archivos y Sistema de Archivos**

El sistema operativo abstrae el manejo de archivos de los detalles físicos de los dispositivos. Los archivos se agrupan en directorios, creando estructuras jerárquicas. Se utilizan llamadas al sistema para manipular archivos y directorios (crear, abrir, leer/escribir, cerrar).

En UNIX, el concepto de montaje permite integrar sistemas de archivos externos al sistema principal. Además, existen archivos especiales (de bloque y carácter) que permiten representar dispositivos de entrada/salida como archivos estándar.

**Canales (Pipes)**

Son pseudoarchivos que facilitan la comunicación directa entre procesos.

**Entrada/Salida (E/S)**

El subsistema de E/S del SO maneja dispositivos físicos como teclados, monitores y periféricos. Existen drivers específicos para cada dispositivo y componentes genéricos que aplican a múltiples dispositivos.

**Protección y Seguridad**

Los sistemas operativos administran la seguridad y confidencialidad mediante mecanismos de protección, como los permisos de archivos en UNIX (lectura, escritura y ejecución, identificados por los bits rwx).

**El Shell**

El shell actúa como intermediario entre el usuario y el sistema operativo, facilitando el uso de llamadas al sistema mediante comandos directos. Ejemplos de shells incluyen sh, csh, ksh y bash.

**Llamadas al Sistema**

Son interfaces entre los programas y el sistema operativo, permitiendo operaciones específicas como lectura (read), creación de procesos (fork), administración de archivos (open, close) y directorios (mkdir, rmdir, link, mount).

**Estructura de los Sistemas Operativos**

* **Monolíticos**: Todo el sistema se ejecuta como un solo programa en modo kernel.
* **De Capas**: Estructura jerárquica organizada en capas con distintos niveles de privilegios.
* **Microkernels**: Núcleo pequeño con módulos adicionales ejecutados en modo usuario.
* **Cliente-servidor**: Procesos servidor ofrecen servicios a clientes mediante mensajes.
* **Máquinas Virtuales**: Proveen múltiples ambientes aislados sobre un solo hardware físico.
* **Exokernels**: Particionan el hardware físico asignando recursos específicos a diferentes usuarios o aplicaciones.

**Clasificación de Sistemas Operativos**

* Por administración de tareas: Monotarea vs. Multitarea.
* Por administración de usuarios: Monousuario vs. Multiusuario.
* Por administración de sesiones: Monosesión vs. Multisesión.
* Por manejo de recursos: Centralizado vs. Distribuido.

# Unidad 2: Procesos Clase 4

**Concepto de Proceso**

Un proceso es una abstracción fundamental que representa un programa en ejecución. Es el concepto clave sobre el que se basan todas las demás funcionalidades del sistema operativo, permitiendo la ejecución pseudoconcurrente mediante la conmutación rápida de la CPU entre procesos, conocida como multiprogramación.

**Modelo del Proceso**

Cada proceso es una instancia independiente de un programa que incluye su propio contador de programa, registros y variables. Aunque cada proceso simula tener su propia CPU virtual, en realidad la CPU física conmuta rápidamente entre ellos.

**Creación de Procesos**

Existen cuatro eventos principales que desencadenan la creación de procesos:

1. **Arranque del sistema**: se crean procesos en primer plano (interacción directa con usuarios) y procesos en segundo plano (daemon o servicios).
2. **Llamadas al sistema**: procesos existentes pueden crear nuevos procesos para realizar tareas específicas.
3. **Petición de usuario**: usuarios interactúan iniciando procesos mediante comandos o clics.
4. **Trabajo por lotes**: sistemas batch crean procesos para manejar trabajos en cola.

**Creación de Procesos en UNIX y Windows**

* **UNIX**: Utiliza la llamada al sistema fork, que duplica exactamente al proceso padre. Luego, el hijo usualmente cambia su imagen de memoria con una llamada adicional execve.
* **Windows**: Emplea la función CreateProcess, que gestiona directamente la creación y ejecución del nuevo proceso, definiendo múltiples parámetros de configuración.

En ambos sistemas, los procesos tienen espacios de direcciones independientes tras su creación, aunque pueden compartir ciertos recursos.

**Terminación de Procesos**

Los procesos pueden terminar por las siguientes razones:

1. **Salida normal**: El proceso completa su tarea normalmente.
2. **Salida por error**: El proceso encuentra un error que no puede manejar.
3. **Error fatal**: Un error grave, como acceso inválido a memoria.
4. **Eliminación por otro proceso**: Otro proceso fuerza la terminación mediante llamadas al sistema específicas (kill en UNIX, TerminateProcess en Windows).

**Jerarquías de Procesos**

Algunos sistemas operativos mantienen una jerarquía clara de procesos. UNIX organiza procesos en un árbol, con init como raíz, mientras que Windows no mantiene jerarquías estrictas, permitiendo una administración más flexible de los procesos creados.

**Estados y Transiciones de un Proceso**

Un proceso puede estar en distintos estados según su ejecución:

* **Ejecución**: Activo y utilizando la CPU.
* **Listo**: Preparado para ejecutarse, esperando asignación de CPU.
* **Bloqueado**: No puede ejecutarse hasta que ocurra cierto evento externo.

Las transiciones entre estos estados ocurren mediante acciones del sistema operativo, eventos externos, o decisiones del planificador del sistema operativo.

**Implementación de los Procesos**

El SO mantiene una tabla de procesos que registra la información crítica necesaria para manejar las suspensiones y reanudaciones de los procesos, incluyendo su estado, registro de memoria, archivos abiertos y planificación de tareas.

**Hilos (Threads)**

Los hilos representan múltiples secuencias de ejecución dentro del mismo espacio de direcciones de un proceso, lo que facilita tareas concurrentes, mejora la eficiencia en operaciones de E/S y permite explotar sistemas multiprocesador. A diferencia de los procesos completos, los hilos comparten la memoria, lo que simplifica ciertas aplicaciones pero requiere coordinación explícita para evitar conflictos.

# Unidad 2: Procesos - Clase 5

**Planificación**

La planificación en sistemas operativos es fundamental para decidir qué proceso o hilo debe ejecutarse cuando múltiples tareas compiten simultáneamente por la CPU. Esta función es desempeñada por el planificador utilizando algoritmos específicos para optimizar el rendimiento y la equidad en el uso de recursos.

**Introducción a la Planificación**

Históricamente, la planificación era sencilla en sistemas de procesamiento por lotes, pero se ha vuelto compleja debido a la multiprogramación y sistemas interactivos modernos. Actualmente, aunque en sistemas personales la planificación es menos crítica, en servidores y sistemas interactivos sigue siendo fundamental para mantener el rendimiento y la equidad.

**Comportamiento de un Proceso**

Los procesos alternan ráfagas de cálculos con esperas por operaciones de entrada/salida (E/S). Algunos procesos son limitados por cálculos (con largas ráfagas de CPU) mientras que otros son limitados por E/S (con ráfagas cortas de CPU y frecuentes esperas).

**Cuándo Planificar Procesos**

La planificación ocurre cuando:

* Se crea un nuevo proceso.
* Un proceso termina.
* Un proceso se bloquea por E/S u otra razón.
* Se produce una interrupción que requiere atención inmediata.

**Tipos de Algoritmos de Planificación**

* **No apropiativos:** El proceso se ejecuta hasta bloquearse o finalizar voluntariamente.
* **Apropiativos:** La CPU puede ser retirada al proceso tras un intervalo definido.

**Categorías de Algoritmos de Planificación**

1. **Procesamiento por lotes:** Favorecen la reducción en cambios de contexto, típicamente no apropiativos.
2. **Interactivos:** Requieren apropiación para garantizar tiempo de respuesta rápido y evitar monopolización de la CPU.
3. **Tiempo Real:** Prioridad a tiempos límite estrictos o suaves, dependiendo del grado de precisión requerida.

**Metas de los Algoritmos de Planificación**

* **Equidad:** Distribuir equitativamente el tiempo de CPU.
* **Aplicación de Políticas:** Cumplir con reglas definidas previamente.
* **Balance:** Optimizar el uso de todos los recursos del sistema.
* **Rendimiento:** Maximizar trabajos procesados y minimizar tiempos de respuesta.

**Planificación en Sistemas de Procesamiento por Lotes**

* **Primero en entrar, primero en ser atendido (FCFS):** Procesos atendidos en orden de llegada.
* **El trabajo más corto primero (SJF):** Prioriza procesos cortos, minimizando el tiempo promedio.
* **El menor tiempo restante a continuación (SRTN):** Versión apropiativa del SJF, enfocada en tiempo restante.

**Planificación en Sistemas Interactivos**

* **Turno circular (Round-Robin):** Cada proceso recibe un tiempo fijo (quántum) cíclicamente.
* **Planificación por prioridad:** Procesos ejecutados según prioridad asignada.
* **Múltiples colas:** Prioridades organizadas en clases, procesos cambian de clase según uso.
* **El proceso más corto a continuación:** Adaptación del SJF para entornos interactivos basado en estimaciones previas.
* **Planificación garantizada:** Proporciona tiempo de CPU proporcional garantizado.
* **Planificación por sorteo:** Procesos reciben boletos de lotería y se selecciona aleatoriamente el ganador.
* **Planificación por partes equitativas:** Tiempo de CPU distribuido equitativamente entre usuarios.

**Planificación en Sistemas de Tiempo Real**

En sistemas de tiempo real es crucial respetar tiempos límite. Existen:

* **Tiempo real duro:** Estrictos tiempos límite obligatorios.
* **Tiempo real suave:** Tiempos límite deseables pero no críticos.

**Política contra Mecanismo**

Separar la política (qué procesos priorizar) del mecanismo (cómo hacerlo técnicamente) permite flexibilidad. Por ejemplo, el kernel proporciona mecanismos mientras los usuarios definen las políticas.

**Planificación de Hilos (Threads)**

* **Hilos a nivel usuario:** Planificados por la aplicación sin intervención directa del kernel, eficientes pero menos robustos.
* **Hilos a nivel kernel:** Planificados por el kernel, permitiendo mejor administración de recursos y estabilidad, aunque más lentos debido al mayor coste de conmutación de contexto.

# Unidad 2: Procesos - Clase 6

**Concurrencia, Exclusión Mutua y Sincronización**

La concurrencia es esencial en sistemas operativos debido a la necesidad de gestionar múltiples procesos e hilos simultáneamente, ya sea mediante multiprogramación, multiprocesamiento o procesamiento distribuido. Los sistemas concurrentes requieren técnicas especiales para asegurar la correcta ejecución y sincronización de procesos, manejo de recursos compartidos y prevención de conflictos.

**Principios de la Concurrencia**

La concurrencia implica entrelazamiento temporal en sistemas monoprocesador o solapamiento en sistemas multiprocesador, presentando problemas similares en ambos casos debido a la impredecibilidad de las velocidades relativas de ejecución de procesos.

**Términos Clave**

* **Sección crítica:** Código que requiere acceso exclusivo a recursos compartidos.
* **Interbloqueo:** Situación en la que varios procesos se bloquean mutuamente.
* **Círculo vicioso (livelock):** Cambio continuo de estado sin progreso efectivo.
* **Exclusión mutua:** Garantía de acceso exclusivo a recursos.
* **Condición de carrera:** Resultado depende del orden de ejecución.
* **Inanición:** Proceso queda indefinidamente excluido del acceso a recursos.

**Competencia entre Procesos por Recursos**

Los procesos en competencia pueden enfrentarse a problemas como exclusión mutua, interbloqueo e inanición, especialmente cuando varios procesos necesitan acceso simultáneo a recursos únicos y críticos.

**Cooperación entre Procesos**

* **Vía compartición:** Procesos comparten objetos sin conocimiento explícito del otro, exigiendo mecanismos que garanticen la integridad de los datos y la exclusión mutua durante operaciones de escritura.
* **Vía comunicación:** Procesos interactúan explícitamente mediante paso de mensajes, siendo cruciales la sincronización y prevención de interbloqueos.

**Exclusión Mutua: Soporte Hardware**

Las soluciones hardware para exclusión mutua incluyen deshabilitación de interrupciones e instrucciones máquina especiales como Test-and-Set o Exchange, útiles para mantener acceso atómico a recursos compartidos pero susceptibles a problemas como espera activa, inanición e interbloqueo.

**Semáforos**

Introducidos por Dijkstra, los semáforos son mecanismos que utilizan señales (semSignal y semWait) para sincronizar procesos. Existen semáforos binarios y semáforos generales (con contador), siendo crucial la implementación de colas para gestionar procesos bloqueados.

**Monitores**

Los monitores encapsulan la exclusión mutua mediante procedimientos y datos locales, asegurando que solo un proceso acceda simultáneamente al recurso protegido. Simplifican la programación concurrente al ofrecer estructuras claras y manejables.

**Paso de Mensajes**

El paso de mensajes permite a los procesos sincronizarse y comunicarse enviando y recibiendo mensajes. Pueden utilizarse primitivas bloqueantes o no bloqueantes, direccionamiento directo o indirecto (a través de buzones), y mensajes de longitud fija o variable.

**Características de Diseño en Sistemas de Mensajes**

* **Sincronización:** Determina el comportamiento del emisor y receptor (bloqueante/no bloqueante).
* **Direccionamiento:** Especificación directa o indirecta del receptor del mensaje.
* **Formato del mensaje:** Mensajes cortos (fijos) o variables según el contexto.

**Exclusión Mutua mediante Paso de Mensajes**

Los procesos pueden implementar exclusión mutua utilizando mensajes mediante un sistema de buzones compartidos. La técnica garantiza que un único proceso acceda a su sección crítica por vez, manejando adecuadamente bloqueos y desbloqueos.

# Unidad 3: Memoria Clase 7

**Gestión de la Memoria**

La gestión de la memoria es una tarea crítica del sistema operativo, especialmente en sistemas multiprogramados, donde la memoria debe subdividirse para alojar múltiples procesos simultáneamente. Esta gestión es esencial para garantizar que el procesador esté siempre ocupado y no ocioso debido a procesos esperando por operaciones de entrada/salida (E/S).

**Requisitos para la Gestión de la Memoria**

* **Reubicación:** Capacidad de mover programas en la memoria principal de forma flexible y dinámica para optimizar el uso del procesador y la memoria disponible.
* **Protección:** Los procesos deben estar protegidos contra interferencias accidentales o intencionadas por parte de otros procesos, utilizando mecanismos de hardware para verificar en tiempo de ejecución todas las referencias de memoria.
* **Compartición:** Posibilidad de permitir acceso controlado a regiones comunes de memoria por múltiples procesos, esencial para eficiencia y cooperación entre procesos.
* **Organización lógica:** Tratamiento modular de los programas para permitir compilación independiente, protección selectiva, y compartición eficiente de módulos mediante segmentación.
* **Organización física:** Separación en memoria principal (rápida y volátil) y memoria secundaria (más lenta, económica y no volátil), gestionando eficazmente el flujo de información entre ambas.

**Particionamiento de la Memoria**

Las técnicas básicas para asignar memoria incluyen particionamiento fijo y dinámico, aunque actualmente se prefieren métodos más avanzados como paginación y segmentación (memoria virtual).

**Particionamiento Fijo**

Se asigna la memoria en particiones de tamaño fijo, pudiendo tener particiones iguales o variables:

* **Particiones iguales:** Pueden causar fragmentación interna y desperdicio de memoria.
* **Particiones variables:** Flexibilidad limitada, con ventajas en la reducción de fragmentación interna, pero aún así presentan restricciones en cantidad y tamaño definidos en tiempo de generación del sistema operativo.

**Particionamiento Dinámico**

En este método, las particiones tienen tamaños variables adaptados exactamente a la necesidad de cada proceso, lo que puede llevar a fragmentación externa debido a pequeños espacios libres dispersos. La compactación periódica es una solución costosa en tiempo de CPU.

**Algoritmos de Ubicación**

* **Mejor-ajuste (best-fit):** Selecciona el bloque más cercano en tamaño a la necesidad del proceso.
* **Primer-ajuste (first-fit):** Elige el primer bloque disponible suficientemente grande desde el principio.
* **Siguiente-ajuste (next-fit):** Similar al primer-ajuste, pero comienza desde el último punto de asignación.

**Algoritmos de Reemplazo**

En situaciones en que todos los procesos están bloqueados y falta espacio, el sistema operativo debe intercambiar procesos entre la memoria principal y el disco, escogiendo procesos adecuados para el reemplazo.

**Sistema Buddy**

Combina ventajas del particionamiento fijo y dinámico asignando memoria en bloques potencias de dos. Cuando se solicita memoria, se divide el bloque hasta alcanzar el tamaño adecuado. Este método gestiona eficientemente la memoria y facilita su asignación y liberación rápida.

# Unidad 3: Clase 8

**Unidad III – Administración de la Memoria**

**Reubicación**

La reubicación es un mecanismo necesario cuando los procesos pueden ocupar diferentes áreas de la memoria en distintos momentos. Esto ocurre debido al intercambio (swap) y compactación. Existen tres tipos de direcciones relevantes en este contexto:

* **Dirección lógica:** Referencia independiente de la ubicación física actual del proceso en memoria.
* **Dirección relativa:** Tipo particular de dirección lógica referenciada desde un punto conocido.
* **Dirección física o absoluta:** Ubicación específica en la memoria principal.

Se requiere soporte hardware especial para traducir direcciones lógicas a físicas en tiempo de ejecución, facilitando la reubicación dinámica y efectiva.

**Paginación**

La paginación resuelve problemas de fragmentación interna y externa dividiendo tanto la memoria principal como los procesos en unidades de tamaño fijo denominadas marcos (memoria) y páginas (procesos). Esta técnica elimina la fragmentación externa y limita la interna a una fracción de una página.

Cada proceso tiene una tabla de páginas que indica dónde se encuentra cada página en memoria principal. El sistema operativo mantiene una lista de marcos libres para asignar nuevas páginas. La dirección lógica en paginación está compuesta por un número de página y un desplazamiento dentro de dicha página. La traducción de dirección lógica a física es realizada por el hardware mediante la tabla de páginas.

**Segmentación**

La segmentación organiza programas y datos en segmentos de tamaño variable. Una dirección lógica en segmentación incluye un número de segmento y un desplazamiento dentro del segmento. La segmentación es similar al particionamiento dinámico, con la ventaja de permitir que los segmentos no sean contiguos, eliminando la fragmentación interna pero manteniendo cierta fragmentación externa.

Esta técnica permite que los programas se dividan en segmentos según criterios lógicos o funcionales, facilitando la programación modular. La tabla de segmentos almacena la dirección inicial y la longitud del segmento, garantizando que las direcciones lógicas sean válidas. La traducción de direcciones lógicas a físicas implica verificar el desplazamiento frente a la longitud del segmento y sumar la dirección inicial del segmento al desplazamiento.

**Resumen General**

La gestión de memoria es crítica en sistemas operativos, encargándose de distribuir eficientemente la memoria principal entre múltiples procesos. La paginación utiliza unidades fijas (páginas y marcos), reduciendo significativamente la fragmentación externa. La segmentación, por otro lado, maneja unidades variables (segmentos), adecuándose más a la lógica de los programas y facilitando la modularidad. Ambos enfoques pueden combinarse para optimizar la gestión de la memoria.

# Unidad 3: Clase 9

**Memoria Virtual**

La memoria virtual permite ejecutar programas más grandes que la memoria física disponible y múltiples programas simultáneamente, al dividir los programas en partes más pequeñas llamadas páginas, almacenadas en marcos de página en memoria física. Cuando un programa accede a una página que no está en memoria, el sistema operativo carga dicha página desde el disco.

**Paginación**

La paginación utiliza direcciones virtuales divididas en número de página virtual y desplazamiento. La MMU (Unidad de Administración de Memoria) realiza la traducción de direcciones virtuales a físicas mediante tablas de páginas, asociando cada página virtual a un marco físico. Las tablas de páginas almacenan entradas que incluyen el número del marco, bits de presencia/ausencia, protección, modificación, referencia y caché.

**Estructuras para Optimización de Paginación**

* **Búfer de traducción adelantada (TLB)**: Es un hardware especial que acelera la paginación almacenando un número limitado de asociaciones página-marco para acceso rápido.
* **Tablas de páginas multinivel**: Reducen la memoria requerida al cargar sólo las páginas necesarias, organizando la tabla en múltiples niveles.
* **Tablas de páginas invertidas**: Ahorran espacio almacenando solo las entradas de páginas que están en memoria física, aunque hacen más compleja la traducción que requiere búsqueda adicional mediante hashing.

**Algoritmos de Reemplazo de Páginas**

* **Óptimo**: Ideal pero imposible en la práctica, reemplaza la página que no se utilizará por más tiempo.
* **No usadas recientemente (NRU)**: Usa bits de referencia (R) y modificación (M) para clasificar páginas según su uso reciente.
* **FIFO (Primera en entrar, primera en salir)**: Remueve la página más antigua, pero podría eliminar páginas aún necesarias.
* **Segunda oportunidad**: Versión mejorada de FIFO que da otra oportunidad a páginas referenciadas recientemente.
* **Reloj**: Similar al anterior, pero más eficiente al usar una lista circular.
* **Menos usadas recientemente (LRU)**: Remueve la página que no ha sido usada por más tiempo.
* **No utilizadas frecuentemente (NFU)**: Contadores aproximan frecuencia de uso, aunque nunca "olvida".
* **Envejecimiento (aging)**: Variante de NFU que reduce contadores periódicamente, aproximándose mejor al LRU.
* **Conjunto de trabajo**: Mantiene páginas recientemente usadas para reducir fallos de página, aunque es costoso.
* **WSClock**: Combina el reloj con conjunto de trabajo para un rendimiento eficiente y práctico.

**Resumen General**

La memoria virtual es una solución clave para gestionar eficazmente grandes programas y múltiples procesos simultáneos. Utiliza técnicas como paginación y algoritmos específicos para decidir qué páginas mantener en memoria física, buscando un equilibrio entre eficiencia y rendimiento del sistema operativo.

# Unidad 4: Clase 10 Archivos

## Sistemas de Archivos

Todas las aplicaciones de computadora requieren almacenar y recuperar información.

La capacidad de almacenamiento está restringida por el tamaño del espacio de direcciones virtuales.

Cuando un proceso termina, su info se pierde.

Varios procesos deben poder acceder a partes de la información al mismo tiempo.

3 Requerimientos esenciales para almacenar info a largo plazo:

* Almacenar mucha info
* La información debe sobrevivir a la terminación del proceso que la use.
* Múltiples procesos deben ser capaces de acceder a la información de forma concurrente.

## Archivos

Unidad logica de inforamcion creada por los procesos.

Los procesos pueden leer archivos existentes y crear nuevos de ser necesario.

La informacion debe ser persistente, no debe ser afectada por la terminacion y creacion de procesos. Los archivos son administrados por el sistema operativo.

Desde el punto de vista del usuario, lo mas importante del archivo es su apariencia; de que se constituye, como se llaman y cómo se protegen, que operaciones se pueden hacer con ellos, etc.

Realmente, los archivos son **un mecanismo de abstracción**, ya que nos permiten una forma de guardar información en el disco y después la posibilidad de acceder a ella.

UNIX es case sensitive respecto al nombre de los archivos, mientras que MS DOS no.

Un archivo puede tener más de 1 extensión: paginaweb.html.zip quiere decir que el archivo es un html (pagina) que después se comprime usando zip.

Las extensiones de los archivos indican algo sobre la naturaleza del mismo, pero no son más que convenciones y no son impuestas por el SO.

Todo archivo tiene nombres y datos. Todos los sistemas operativos asocian informacion con cada archivo, como la fecha y hora de la ultima modificacion, o su tamaño. Estos elementos se le llama “Atributos de los archivos” o “Metadatos”.

## Formas de estructurar archivos

Primer opcion, es una secuencia de bytes sin estructura. El SO no sabe ni le importa que tiene el archivo. Todas las versiones de UNIX, MS-DOS y Windows usan esto.

Segunda, secuencia de registros. Tiene estructura. Los registros tienen longitud fija, el usuario agrega o modifica registros.

Tercera: Arbol (probably B). Tiene estructura. Se compone de registros y campos llave en posiciones fijas. Se ordena en base al campo llave para permitir una busqueda rapida.

## Tipos de archivos

Windows: Tiene archivos y directorios regulares.

UNIX: Tiene los de windows mas archivos especiales de caracteres y de bloques.

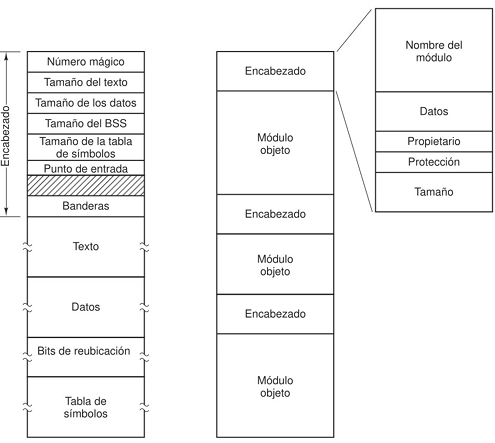
Archivos regulares: Contienen info del usuario.

Directorios: Sistemas de archivos para mantener la estructura del sistema de archivos (jijo)

Archivos especiales de caracteres: Modelan los dispositivos de E/S en serie.

Archivos especiales de bloque: Se usan para modelar discos.

Existen archivos tipo ASCII que se pueden mostrar e imprimir como estan, y despues estan los binarios.



Antes los SO solo podian acceder a los archivos de forma secuencial, porque usaban cintas magneticas. Cuando vinieron los discos duros, se podia acceder a los registros mediante llaves. Los famosos archivos de acceso aleatorio.

## Operaciones de archivos

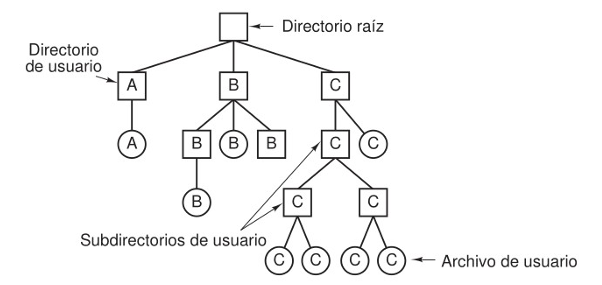
Create Delete Open Close Read Write Append Seek Get attributes Set attributes Rename

CDOCRWASGSR -> no es como que funcione muy bien la regla nemotecnica

## Organizacion de directorios

La forma mas simple de organizarlos es con un directorio que contenga todos los archivos. Se usaba antes, pero cuando tenes muchos archivos es horrible.

La otra forma es con una jerarquia de directorios, donde tenes un directorio raiz base, y de ahi desprenden distintos directorios para cada usuario, y los subdirectorios que tenga el mismo junto con sus archivos.



Las rutas absolutas siempre arrancan por el directorio raiz y son unicas.

En UNIX se separa usando / cada componente, en Windows \ y en MULTICS >.

Lo otro son las rutas relativas, que parten desde el directorio que se este usando actualmente. “.” sirve para referenciar el directorio actual, “..” para referenciar el padre, excepto en el directorio raiz que se refiere a si mismo.

## Operaciones de directorios en UNIX:

Create Delete Opendir Closedir Readdir Rename Link Unlink

CDOCRRLU -> tampoco funca la nemotecnica.

Readdir devuelve a siguiente entrada en un directorio abierto, Link sirve para vincular un archivo a mas de un directorio.

# Unidad 4: Clase 11 Implementación de un sistema de archivos

Los discos se pueden dividir en varias particiones, donde cada particion puede tener su propio sistema de archivos. Me voy a ahorrar la explicacion de MBR aca, ya que la vimos en la practica y esta en su propio resumen.

El superbloque del principio, del MBR, contiene los parametros clave acerca del sistema de archivos, y se lee al principio cuando se arranca la computadora, o cuando se entra en contacto con el sistema de archivos por primera vez.

## Implementación de archivos

### Asignación continua

Es el esquema mas simple, almacena cada archivo como una serie contigua de bloques de disco. Por ejemplo, cada bloque de 1 kb, si tengo un archivo de 50kb se le asigna 50 bloques. Los nuevos bloques se ubican al final del disco.

Ventajas:

> Es simple de implementar, ya que para ubicar un registro solo hace falta saber la direccion en el disco del primer bloque, y el numero de bloques del archivo.

> El rendimiento de lectura es excelente ya que se puede leer el archivo completo en una sola operacion.

**Desventajas:**

> Los discos se fragmentan muy facilmente.

> Cada tanto hay que compactar el disco, es decir, reubicar los bloques, para evitar la fragmentacion. Esto es sumamente costoso.

**Ejemplo de uso:** CD-ROMs, porque el tamaño es fijo y nunca cambia. Los DVDs usan formato UDF, que como maximo le deja alojar archivos de maximo 1 gb. Aproximadamente pueden guardar hasta 4.5 GBs, por lo que se usan varios archivos.

### Asignación de lista enlazada

Se mantiene cada archivo como una lista enlazada de bloques en el disco. La primer palabra de cada bloque se usa para apuntar al siguiente.

**Ventajas**:

* No se pierde espacio por fragmentacion.
* La lectura secuencial del archivo es directa.

**Desventajas**

* El acceso aleatorio es extremadamente lento. Para llegar al bloque N necesita pasar por los N-1 anteriores.
* El apuntador ocupa unos bytes al principio, reduciendo ligeramente la cantidad de almacenamiento del bloque.
* En funcion de como se arme el tamaño de este apuntador, leer el tamaño del bloque completo requiere adquirir y concatenar informacion de dos bloques completos. Lo cual genera un gasto adicional en procesamiento debido a la copia.

### Asignación de lista entrelazada, usando una tabla en memoria.

Es una forma de eliminar las desventajas de la lista entrelazada. Donde ponemos en una tabla las partes donde arranca cada bloque de cada archivo dentro del a memoria PRINCIPAL (ram).

**Ventaja:**

* El acceso aleatorio es mas sencillo.
* Los bloques completos quedan disponibles para los datos.

**Desventaja**:

* Como metes tantas boludeces en la ram????????????
* digo, necesitas tener la tabla en memoria todo el tiempo para que funcione, y en funcion del tamaño de los bloques, la cantidad de RAM que necesitas dedicarle escala monstruosamente.

### Nodos-i (iNodos)

# Unidad 4: Clase 12