Apunte Digital de Sistemas Operativos

Funcionamiento del SO, Procesos, Planificación, Memoria, TLB, Swapping, E/S y Seguridad

José Alfredo Martínez Valdés

Universidad de Antioquia — Facultad de Ingeniería

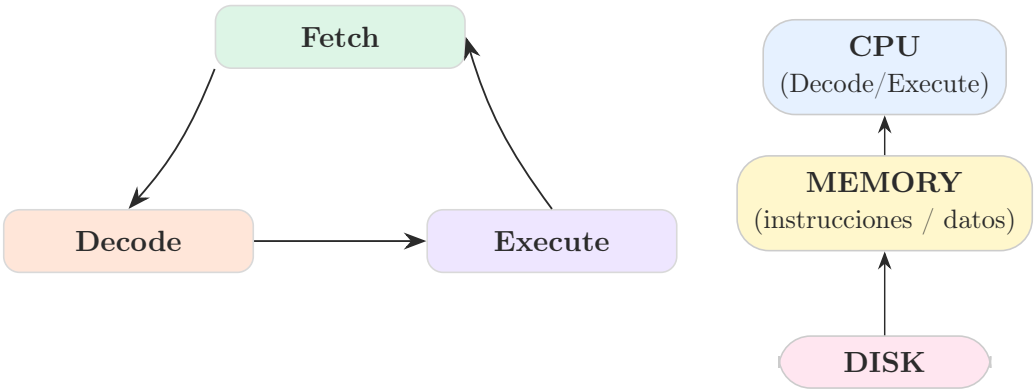
Programa: Ingeniería de Sistemas

Curso: Sistemas Operativos 2025-2

Medellín, Colombia — 2025

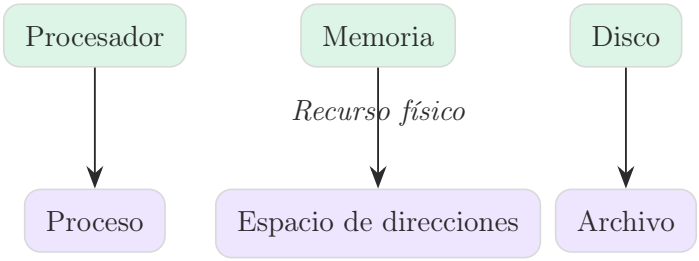
Ejecución de un programa

Ciclo básico (CPU): 1) *Fetch*: obtiene la instrucción desde memoria. 2) *Decode*: averigua el tipo de instrucción. 3) *Execute*: realiza la operación indicada. 4) *Next*: salta a la siguiente instrucción.



Entonces el sistema operativo administra justa y eficientemente los recursos, controla la ejecución de los programas (accesos a *API System*) y brinda la **abstracción** de los recursos para facilitar su uso.

Virtualización de recursos

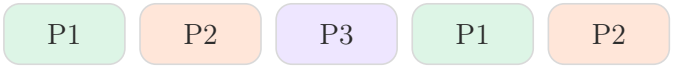


Transformación virtual

Llamadas al sistema permiten que el usuario pida servicios al SO. La API del sistema actúa como interfaz segura entre programas y recursos.

Virtualización de CPU: ilusión de "muchas CPUs"

Idea clave: mediante *time sharing*, el SO multiplexa la CPU y produce la *ilusión* de múltiples procesos ejecutándose en paralelo sobre una única CPU.



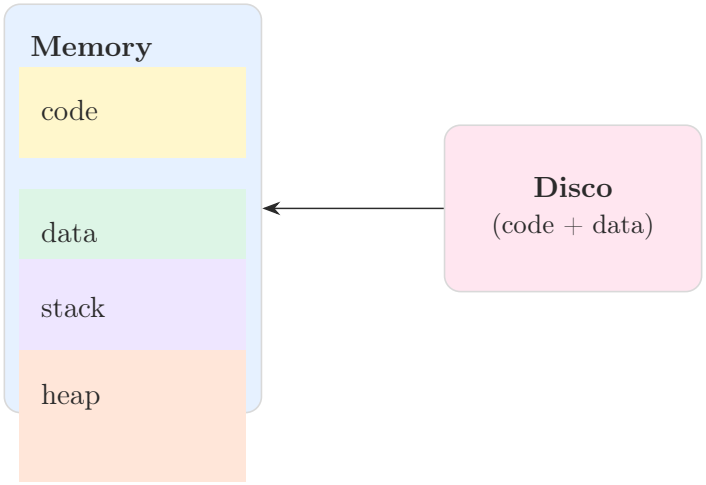
Time sharing (rebanadas de CPU)

Programa: código completo almacenado en un archivo ejecutable (`.exec`) —*entidad pasiva*.
Proceso: instancia de un programa en ejecución —*entidad activa*.

Ahora, los procesos

Proceso = CPU + Memoria + Información de I/O

Conformación de un proceso (memoria)

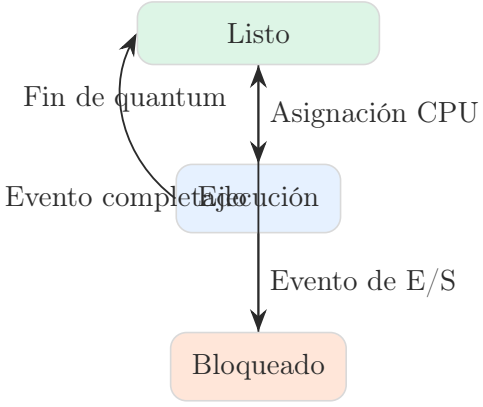


Notas

- La **abstracción** simplifica el uso de los recursos; la **protección** evita interferencias entre procesos; la **multiplexación** comparte recursos de forma eficiente.
- La virtualización de CPU crea la *ilusión* de paralelismo sobre una sola CPU mediante *time sharing*.
- La organización de memoria típica de un proceso incluye: *code*, *data*, *heap* y *stack*.

Multitarea y planificación de la CPU

El **Sistema Operativo** coordina la ejecución de múltiples procesos compartiendo la CPU entre ellos. Cada proceso recibe una *porción de tiempo* denominada **quantum**. Cuando ese tiempo se agota, el SO realiza un **cambio de contexto** (*context switch*) y pasa la CPU a otro proceso.



Estados de un proceso:

- **Listo (Ready)**: el proceso está preparado para ejecutarse.
- **Ejecución (Running)**: ocupa actualmente la CPU.
- **Bloqueado (Waiting)**: espera un evento externo (por ejemplo, E/S).

Planificación de la CPU

El **planificador** (scheduler) decide qué proceso se ejecuta y durante cuánto tiempo. Su política afecta la eficiencia, la equidad y la capacidad de respuesta del sistema.

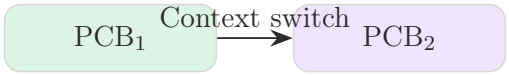
Rebanadas de CPU (time slices)

Algoritmos comunes de planificación:

- **FCFS (First Come, First Served)**: orden de llegada.
- **SJF (Shortest Job First)**: prioriza el proceso más corto.
- **RR (Round Robin)**: asigna turnos iguales (*quantum*).
- **Prioridades**: selecciona según nivel de prioridad.

Cambio de contexto

Cuando el SO interrumpe un proceso, debe guardar su **estado de CPU** (registros, PC, pila) en su **PCB** (*Process Control Block*). Luego, carga el estado del siguiente proceso y continúa la ejecución.



El **PCB** almacena:

- Contador de programa (PC)
- Registros de CPU
- Punteros a memoria
- Estado del proceso

Multitarea: cooperativa y preventiva

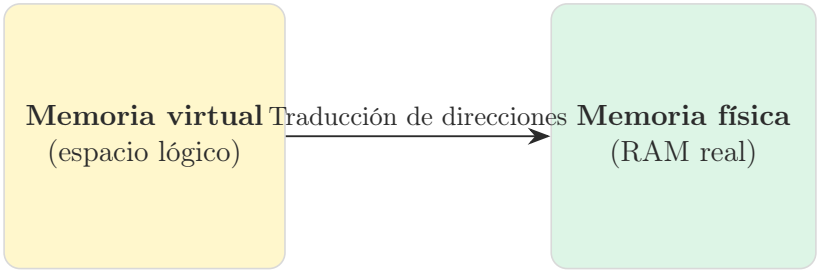
Multitarea cooperativa: cada proceso cede voluntariamente la CPU. **Multitarea preventiva**: el SO fuerza la liberación tras un quantum mediante interrupciones.

Cooperativa *Preventiva*

Los sistemas modernos emplean planificación **preventiva** para asegurar equidad y capacidad de respuesta.

Memoria virtual: concepto general

La **memoria virtual** es una técnica que permite a cada proceso disponer de un **espacio de direcciones lógico** independiente del tamaño real de la memoria física. El Sistema Operativo traduce direcciones virtuales a direcciones físicas mediante estructuras de datos denominadas **tablas de páginas**.



Proceso A *Marcos físicos*

Ventajas principales:

- Permite ejecutar procesos más grandes que la memoria real.
- Aísla a los procesos entre sí, protegiendo sus espacios de memoria.
- Facilita la multiprogramación y la eficiencia global del sistema.

Segmentación de la memoria

En la **segmentación**, el espacio de direcciones se divide en segmentos lógicos: *código, datos, pila, heap*, etc. Cada segmento tiene tamaño variable y puede ubicarse en cualquier lugar de la memoria física.

El **registro de segmento** guarda:

- **Base**: dirección física de inicio del segmento.
- **Límite**: tamaño máximo permitido.

La **unidad de gestión de memoria (MMU)** valida y traduce las direcciones.

Traducción: direcciones lógicas a físicas

Cuando un programa accede a una dirección, esta se divide en dos partes:

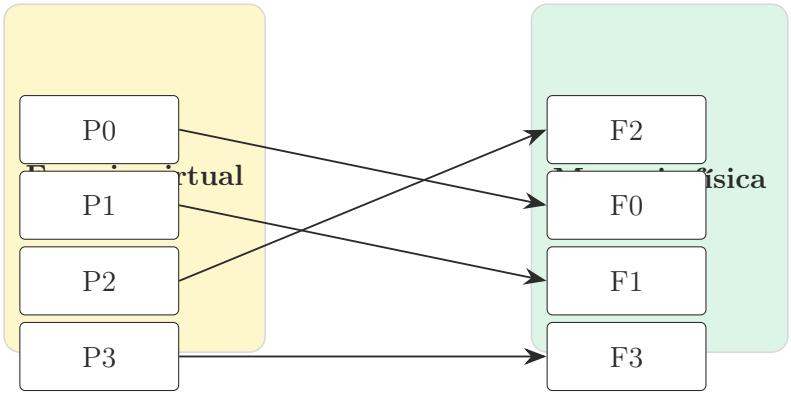
1. **Número de segmento (s)** o de página.
2. **Desplazamiento (d)** dentro del segmento o página.

El SO consulta la tabla correspondiente para obtener la dirección física real.

La traducción de direcciones se realiza **en tiempo de ejecución** mediante hardware especializado (MMU). Esto otorga flexibilidad y aislamiento entre procesos.

Paginación: división uniforme de la memoria

La **paginación** divide el espacio de direcciones lógico en **páginas** de tamaño fijo, y la memoria física en **marcos** del mismo tamaño. Cada página puede almacenarse en cualquier marco, permitiendo un uso más eficiente de la memoria.



Cada proceso posee su propia **tabla de páginas**, que traduce los números de página a los marcos físicos. El hardware utiliza esta tabla para convertir direcciones virtuales en direcciones físicas durante la ejecución.

Tabla de páginas

Cada entrada de la **tabla de páginas** contiene:

- Número de **marco físico** correspondiente.
- **Bit de validez**: indica si la página está cargada en memoria.
- **Bits de protección**: controlan lectura, escritura y ejecución.
- **Bit de uso (referencia)** y **bit de modificación (dirty)**.

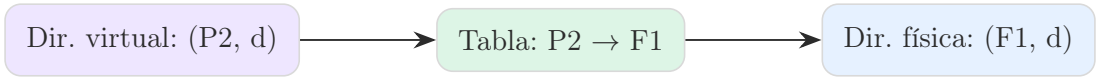
Tabla de páginas

P0	→	F2	V=1
P1	→	F3	V=1
P2	→	F1	V=1
P3	→	—	V=0

Durante un acceso de memoria:

1. La **MMU** toma el número de página de la dirección virtual.
2. Consulta la tabla de páginas.
3. Sustituye el número de página por el número de marco.
4. Forma la dirección física final sumando el desplazamiento.

Ejemplo de traducción



La memoria virtual combina **simplicidad de paginación** con **protección de segmentación**, y permite la ejecución simultánea de muchos procesos sin interferencia.

Apunte digital — Sistemas Operativos. Paginación y Tabla de Páginas.

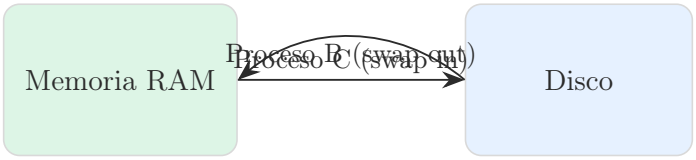
Translation Lookaside Buffer (TLB)

El **TLB (Translation Lookaside Buffer)** es una memoria caché de alta velocidad que almacena un número limitado de traducciones recientes de páginas virtuales a marcos físicos. Reduce el tiempo de acceso a memoria y evita consultar la tabla de páginas en cada acceso.

Si hay acierto (TLB hit): se obtiene la dirección física de inmediato. **Si hay fallo (TLB miss):** se consulta la tabla de páginas y se actualiza el TLB.

Swapping o intercambio de procesos

El **swapping** consiste en trasladar procesos completos entre la memoria principal y el disco (según su prioridad o inactividad) para liberar espacio y permitir la ejecución de otros procesos.



El Sistema Operativo mantiene una **cola de procesos residentes** y una **cola de intercambio**. La política de intercambio depende del nivel de ocupación y prioridad.

Fallos de página

Un **fallo de página (page fault)** ocurre cuando un proceso intenta acceder a una página que no está en memoria física. El SO debe:

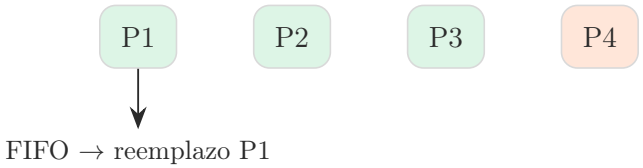
1. Detener el proceso momentáneamente.
2. Cargar la página faltante desde el disco.
3. Actualizar la tabla de páginas.
4. Reanudar la ejecución.

Reemplazo de páginas

Cuando la memoria está llena y se produce un fallo de página, el sistema debe **elegir una página víctima** para reemplazarla.

Algoritmos de reemplazo más comunes:

- **FIFO (First In, First Out)**: elimina la página más antigua.
- **LRU (Least Recently Used)**: elimina la página menos usada recientemente.
- **Óptimo (OPT)**: reemplaza la que no se usará en el futuro cercano (teórico).



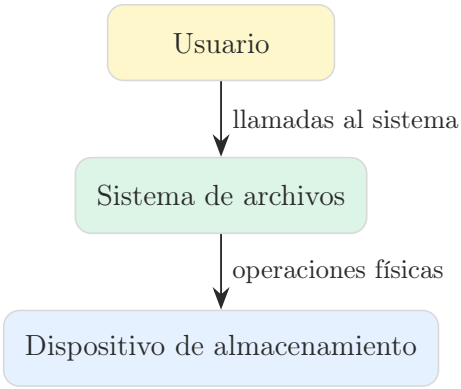
El objetivo de los algoritmos de reemplazo es minimizar los **fallos de página** y optimizar el uso de la memoria física disponible.

Resumen general

- **TLB**: acelera la traducción de direcciones.
- **Swapping**: intercambia procesos completos entre RAM y disco.
- **Reemplazo de páginas**: gestiona qué páginas permanecen en memoria.

Sistema de archivos: organización y estructura

El **sistema de archivos** es el componente del Sistema Operativo que organiza, gestiona y controla el acceso a los datos almacenados en dispositivos de almacenamiento (disco, SSD, etc.). Su función es traducir las operaciones de alto nivel (*abrir, leer, escribir, cerrar*) en operaciones físicas sobre los bloques del dispositivo.



Funciones principales:

- Control de acceso y permisos.
- Gestión del espacio libre y asignación de bloques.
- Mantenimiento de la jerarquía de directorios.
- Protección y recuperación ante fallos.

Estructura de un archivo

Un **archivo** es una colección de información relacionada bajo un nombre único. Puede almacenar texto, código, imágenes o datos binarios. El SO utiliza una estructura de control llamada **FCB (File Control Block)** para mantener información sobre cada archivo.

FCB - File Control Block

Nombre	datos.txt
Tipo	texto
Ubicación	Bloques 14–17
Tamaño	4 KB
Propietario	usuario
Permisos	rw-

El FCB se asocia a cada archivo abierto y permite acceder a su contenido sin tener que recorrer el disco cada vez.

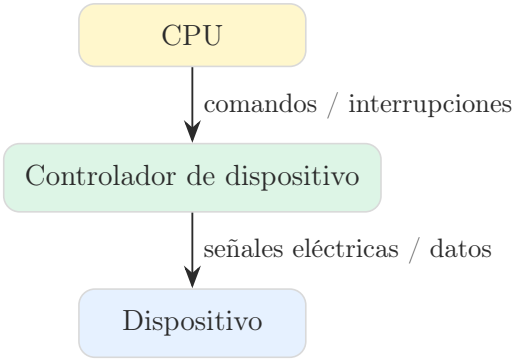
Métodos de acceso a archivos

Existen tres métodos principales:

- **Acceso secuencial**: los datos se leen en orden.
- **Acceso directo (aleatorio)**: se accede a cualquier bloque mediante su dirección.
- **Acceso indexado**: usa un índice que referencia los bloques de datos.

Subsistema de Entrada/Salida (E/S)

El subsistema de **Entrada/Salida (E/S)** coordina la comunicación entre los dispositivos periféricos y la CPU. Incluye **controladores de dispositivos**, **buffers**, **colas de E/S** y **mecanismos de interrupción**.



Los controladores se comunican con el SO a través de **interrupciones**, que notifican la finalización de una operación de E/S. Esto permite que el procesador ejecute otras tareas mientras la E/S está en curso.

Buffering y Spooling

Buffering: usa memoria intermedia para almacenar datos temporalmente mientras se transfieren entre CPU y dispositivos. **Spooling:** gestiona múltiples trabajos de E/S en cola, común en impresión o tareas de red.

El **buffering** y el **spooling** aumentan la eficiencia y evitan bloqueos del sistema por operaciones lentas.

Resumen general

- El sistema de archivos organiza y controla los datos persistentes.
- Los métodos de acceso determinan cómo se recupera la información.
- La E/S coordina la comunicación entre CPU y dispositivos periféricos.
- El buffering y spooling optimizan el rendimiento del sistema.

Seguridad del sistema operativo

La **seguridad del sistema operativo** garantiza la confidencialidad, integridad y disponibilidad de los recursos. El SO protege contra accesos no autorizados, uso indebido de privilegios y ataques externos.

Objetivos fundamentales:
- **Confidencialidad**: solo usuarios autorizados acceden a la información.
- **Integridad**: los datos no pueden modificarse sin permiso.
- **Disponibilidad**: los recursos siempre deben estar accesibles.

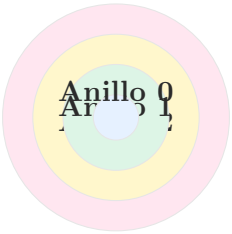
Mecanismos de protección

El sistema operativo utiliza mecanismos de **protección** para controlar el uso de los recursos del sistema por parte de los procesos y usuarios. El objetivo es evitar interferencias o accesos indebidos.

Los mecanismos de protección pueden ser:
- **Autenticación**: verificar identidad del usuario.
- **Autorización**: validar los permisos de acceso.
- **Contabilidad (auditoría)**: registrar actividades del sistema.

Anillos de protección del procesador

Los **anillos de protección** delimitan niveles de privilegio en la CPU. El **anillo 0** posee el máximo privilegio (núcleo del sistema), y el **anillo 3** el mínimo (usuario).



Nivel de privilegio decreciente

Anillo 0: núcleo del sistema operativo. **Anillo 1–2:** controladores, servicios del sistema. **Anillo 3:** aplicaciones de usuario.

Matriz de control de acceso (ACL)

Una **matriz de control de acceso** define qué operaciones puede realizar cada sujeto sobre los objetos del sistema:

black!5 **Usuario / Recurso**	**Archivo A**	**Impresora**	**Base de datos**
Alice	R/W	–	R
Bob	R	W	–
Root	R/W/X	W	R/W

Cada fila corresponde a un usuario (sujeto) y cada columna a un recurso (objeto). Las celdas indican los permisos: lectura (R), escritura (W), ejecución (X).

Políticas de seguridad

Política discrecional (DAC): los usuarios controlan los permisos sobre sus objetos. **Política obligatoria (MAC):** el sistema impone reglas globales (clasificación, niveles). **Política basada en roles (RBAC):** los permisos se asocian a roles predefinidos.

El uso de políticas mixtas (ej. RBAC + MAC) combina flexibilidad y seguridad, adaptando el control a distintos entornos.

Resumen general

- La seguridad del sistema operativo protege la información y los recursos.
- Los anillos de protección definen niveles de privilegio.
- Las políticas de seguridad determinan cómo se gestionan los permisos.
- Los mecanismos de auditoría permiten rastrear acciones y prevenir abusos.

Apunte digital — Sistemas Operativos. Seguridad, Protección y Gestión de Recursos.