

Tema 17

Gestión de Memoria

Sistemas Operativos

 Optimización, protección y asignación eficiente de recursos de memoria

1. Introducción

La **gestión de memoria** es una función crítica del sistema operativo.

🧩 Objetivos principales:

- Eficiencia
- Protección
- Asignación dinámica
- Soporte a multitarea
- Rendimiento óptimo

2. Tipos de Memoria

Tipo	Características
RAM	Principal, volátil, rápida
ROM	No volátil, contiene BIOS/UEFI
Caché	Entre CPU y RAM, muy rápida
Virtual	Usa disco como extensión de RAM
Secundaria	HDD / SSD – almacenamiento permanente
Flash	No volátil, rápida – SSD, USB

3. Funciones del SO en gestión de memoria

3.1 Asignación de memoria

- **Estática:** al inicio del proceso
- **Dinámica:** durante ejecución
- A procesos del usuario y del sistema

3.2 Protección y aislamiento

- Cada proceso tiene su **espacio aislado**
- Previene corrupción de datos o accesos indebidos

3.3 Liberación de memoria

- Al finalizar un proceso
- Automatizada en algunos lenguajes:
 - Garbage Collector (Java, Python...)

3.4 Intercambio (Swapping)

- Mueve procesos entre RAM ↔ Disco
- Libera memoria activa
- Mejora multitarea, aunque reduce rendimiento si es excesivo

4. Técnicas de gestión

4.1 Fragmentación

Tipo	Descripción
Interna	Espacio perdido dentro de bloques
Externa	Huecos dispersos entre bloques

 Técnicas:

- Paginación → reduce fragmentación externa
- Segmentación → reduce interna
- Compactación → alto coste computacional

4.2 Segmentación

- Divide el proceso en **segmentos**: código, pila, datos
- Usa **tabla de segmentos**
- Mejora protección y modularidad

4.3 Paginación

- Divide memoria lógica en **páginas**, física en **frames**
- Traducción de direcciones con **tabla de páginas**
- Usa **TLB (Translation Lookaside Buffer)** para acelerar accesos
- Reduce fragmentación externa

4.4 Segmentación paginada

- Combina segmentación + paginación
- Cada segmento se divide en páginas

 Dirección lógica:

[segmento] → [página] → [desplazamiento]

✓ Muy usada en arquitecturas modernas

4.5 Memoria compartida

- Permite acceso simultáneo de procesos a una región común
- Útil para **IPC (Inter Process Communication)**

5. Aspectos avanzados

5.1 Paginación por demanda

- Las páginas se cargan solo cuando se usan
- Si no está en RAM → **page fault**
- Mejora uso de memoria → penaliza rendimiento si excesivo

5.2 Thrashing



Thrashing = exceso de carga/descarga de páginas



Causa: demasiados procesos con alta demanda



Soluciones:

- Limitar procesos
- Mejorar RAM
- Algoritmos de reemplazo eficientes

5.3 Algoritmos de reemplazo de página

Algoritmo	Descripción
FIFO	Reemplaza la página más antigua
LRU	Reemplaza la menos usada recientemente
Óptimo	Ideal teórico (no implementable)
Clock	Usa bit de uso + puntero circular

5.4 Memoria en sistemas empotrados

- Recursos limitados
- No se usa memoria virtual
- Asignación **estática**, precisa y eficiente

5.5 Herramientas de análisis y depuración

 Herramientas comunes:

- Valgrind (C/C++)
- VisualVM (Java)
- tracemalloc (Python)

 Detectan fugas y uso erróneo de memoria

6. Tendencias modernas

6.1 Memoria persistente (NVDIMM)

- Combina la velocidad de la RAM con la persistencia del disco
- Uso creciente en servidores y bases de datos

6.2 Optimización con IA


- Predicción de uso de memoria para optimizar paginación
- Asignación y liberación adaptativa

6.3 Contenedores y virtualización

 Gestión de memoria en:

- **Docker, Kubernetes**
- **cgroups** → controlan memoria por contenedor
- Aislamiento total del entorno

Conclusión

- ✓ La memoria es el recurso más crítico en sistemas multitarea
- ✓ Su gestión debe ser eficiente, segura y flexible
- ✓ Involucra hardware (RAM, TLB) y software (tablas, algoritmos)
-  Conocerla permite entender cómo se ejecutan realmente los programas

