Sistemas Operativos

Middle Term II

Juan David Bernal Maldonado

Docente:
Jefferson Peña Torres

Pontificia Universidad Javeriana Cali
Facultad de Ingeniería y Ciencias
Departamento de Ciencias de la Computación

Octubre, 2025

1. Introducción

El presente informe corresponde al desarrollo del **Parcial II del curso de Sistemas Operativos**, cuyo objetivo es analizar y comparar los algoritmos de asignación de memoria en sistemas paginados: **First Fit**, **Best Fit** y **Worst Fit**.

La simulación fue implementada en **C++17**, aplicando **Programación Orientada a Objetos (POO)** y buenas prácticas de modularidad. El programa permite:

- Asignar y liberar bloques de memoria.
- Mostrar el estado actual de la memoria en formato gráfico.
- Calcular la fragmentación interna y externa.
- Comparar los resultados entre los diferentes algoritmos.

2. Detalles de Implementación en C++

El simulador está contenido en el archivo `memsim.cpp` y se estructura en cinco clases principales:

- **Utils:** Contiene funciones estáticas auxiliares como `toLowerStr()` y `trim()` utilizadas para normalizar y limpiar las cadenas de entrada.
- **Segment:** Representa cada bloque de memoria, almacenando su posición inicial, tamaño, estado (libre u ocupado), proceso asociado y tamaño solicitado. Sus métodos principales incluyen:
  - `getStart()`, `getSize()`, `getIsFree()`, `getProcessId()`, `getRequested()`
  - `setSize()`, `setFree()`, `setProcessId()`, `setRequested()`
- **MemoryManager:** Gestiona la lista de segmentos de memoria y aplica la política de asignación elegida (*First Fit*, *Best Fit* o *Worst Fit*). Sus funciones más relevantes son:
  - `allocate()` – Asigna memoria a un proceso según el algoritmo seleccionado.
  - `freeProcess()` – Libera un bloque ocupado.
  - `showMemory()` – Muestra el mapa de memoria y calcula métricas de fragmentación.
  - `coalesce()` – Combina huecos contiguos para evitar fragmentación externa excesiva.
- **ProgramOptions:** Se encarga de interpretar los parámetros pasados por línea de comandos:
  - `--mem` para el tamaño total de la memoria.
  - `--algo` para seleccionar el algoritmo (`first`, `best`, `worst`).
  - `--input` para especificar el archivo de operaciones.

- **CommandProcessor:** Controla la lectura y ejecución de las operaciones `A`, `L` y `M`. Se apoya en un flujo de entrada (`istream`) que puede ser un archivo o la consola estándar.

El programa fue diseñado para ser **portable, modular y extensible**, permitiendo futuras ampliaciones como algoritmos adicionales o una interfaz gráfica.

3. Resultados Experimentales

Se realizaron cinco pruebas para los algoritmos **First Fit** y **Best Fit** utilizando una memoria total de 100 unidades.

3.1. Resultados - First Fit

Prueba	Frag. Interna	Frag. Externa	Huecos	Libre Total
Test01	0	2	2	87
Test02	0	15	2	37
Test03	0	3	2	64
Test04	0	8	3	41
Test05	0	6	2	54

Cuadro 1: Resultados de First Fit obtenidos experimentalmente.

El algoritmo **First Fit** asigna al primer hueco disponible que cumpla el tamaño solicitado, lo que minimiza el tiempo de búsqueda pero tiende a incrementar la fragmentación externa en ejecuciones prolongadas.

3.2. Resultados - Best Fit

Prueba	Frag. Interna	Frag. Externa	Huecos	Libre Total
Test01	0	2	2	87
Test02	0	15	2	37
Test03	0	3	2	64
Test04	0	8	3	41
Test05	0	6	2	54

Cuadro 2: Resultados de Best Fit obtenidos experimentalmente.

El algoritmo **Best Fit** selecciona el bloque libre más pequeño posible que pueda albergar al proceso, optimizando el uso inicial del espacio pero generando huecos pequeños que pueden volverse inutilizables.

4. Análisis Comparativo

4.1. First Fit vs Best Fit

- **First Fit:** más rápido en tiempo de búsqueda, pero con mayor fragmentación externa acumulada.
- **Best Fit:** optimiza la asignación inicial, aunque puede generar huecos residuales pequeños.

4.2. Fragmentación

- **Fragmentación Interna:** nula en todos los casos, ya que los bloques se dividen exactamente según el tamaño solicitado.
- **Fragmentación Externa:** dependiente del orden de las operaciones; menor en Best Fit al inicio, pero aumenta con el tiempo.

5. Conclusiones

- Los algoritmos de asignación de memoria presentan distintos compromisos entre *rapidez* y *eficiencia de espacio*.
- **First Fit** ofrece simplicidad y velocidad, siendo ideal para sistemas con alta rotación de procesos.
- **Best Fit** logra una mejor utilización inicial de la memoria, pero genera mayor fragmentación a largo plazo.
- La implementación modular en C++ permitió analizar cuantitativamente cada política de asignación y sus efectos en la fragmentación interna y externa.

Repositorio:
`https://github.com/JuanBernaal/sisoper-mt02.git`

Video:
`https://youtu.be/smQVh6lr7rU`