Informe de Resultados: Simulación de Asignación de Memoria

David Bayona

October 16, 2025

Contents

1 Síntesis

El propósito es simular y analizar distintos métodos de asignación de memoria en sistemas operativos, comparando el funcionamiento de los algoritmos **First Fit**, **Best Fit** y **Worst Fit**.

Se implementaron y probaron dos modelos de asignación de memoria:

1. **Memoria fija (Fixed Partitioned Memory)**
   - La memoria se divide en bloques de tamaño predefinido.
   - Cada proceso solo puede ocupar un bloque completo, incluso si no lo necesita todo.
   - Produce **fragmentación interna**, pero es más simple de administrar.
2. **Memoria dinámica (Dynamic Partitioned Memory)**
   - La memoria no tiene divisiones fijas.
   - Permite **fragmentación externa**, pero aprovecha mejor el espacio total disponible.

Ambos modelos se implementaron en C++.

2 Descripción general del programa

El programa recibe comandos por consola o archivo:

- `A <proceso> <tamaño>` → Asignar memoria
- `L <proceso>` → Liberar memoria
- `M` → Mostrar estado actual de la memoria
- `P <1/2/3>` → Cambiar política de asignación

Políticas disponibles:

1 → **First Fit**: Asigna al primer hueco libre donde quepa el proceso.

2 → **Best Fit**: Asigna al hueco más pequeño que sea suficiente.

3 → **Worst Fit**: Asigna al hueco más grande disponible.

2.1 Dynamic Partitioned Memory

Memoria inicial:

```
[Libre:100]
```

Ejemplo de entrada:

```
A P1 10
A P2 25
L P1
A P3 8
M
```

Salida:

```
[P2:25][Libre:10][P3:8][Libre:57]
```

3 Resultados con el análisis comparativo de los algoritmos

Se realizaron 5 pruebas con distintas secuencias de asignaciones y liberaciones.

3.1 input1.txt — Caso básico con huecos pequeños

Entrada:

```
P 1
A P1 10
A P2 25
A P3 15
L P2
A P4 5
A P5 20
M
```

Salida:

```
[P1:10][P4:5][P3:15][P5:20][Libre:20][Libre:30]
```

Análisis:

- Todos los procesos asignados correctamente.
- Fragmentación externa moderada (2 huecos).
- First Fit logra ocupación estable sin esperas.

3.2 input2.txt — Fragmentación y recombinación

Entrada:

```
P 3
A A1 20
A A2 30
A A3 10
L A2
A A4 15
A A5 25
L A1
A A6 40
M
```

Salida:

```
A6 must wait
[Libre:20][A5:25][A3:10][A4:15][Libre:25][Libre:5]
```

Análisis:

- Proceso `A6` no encuentra espacio → "must wait".
- Alta fragmentación externa.
- Worst Fit no logró reutilizar bien el espacio.

3.3 input3.txt — Cambiando políticas

Entrada:

```
P 1
A X1 10
A X2 15
P 2
A X3 5
A X4 20
L X2
P 3
A X5 18
A X6 22
M
```

Salida:

```
[X1:10][Libre:15][X3:5][X4:20][X5:18][X6:22][Libre:10]
```

Análisis:

- Secuencia híbrida: se alternaron las tres políticas.
- Ningún proceso quedó en espera.
- Distribución ordenada, huecos manejables.

3.4 input4.txt — Caso con esperas

Entrada:

```
P 2
A B1 40
A B2 30
A B3 25
L B2
A B4 50
A B5 10
A B6 5
M
```

Salida:

```
B4 must wait
[B1:40][B5:10][B3:25][B6:5][Libre:20]
```

Análisis:

- Best Fit no pudo colocar `B4`, a pesar de huecos libres.
- Fragmentación intermedia.
- Espacio total libre: 20 unidades.

3.5 input5.txt — Escenario complejo

Entrada:

```
P 3
A P1 10
A P2 20
A P3 30
A P4 15
L P2
L P4
A P5 5
A P6 12
A P7 25
L P3
A P8 18
A P9 40
M
```

Salida:

```
P7 must wait
P9 must wait
[P1:10][P6:12][P8:18][Libre:15][P5:5][Libre:20][Libre:8][Libre:12]
```

Análisis:

- Dos procesos quedaron sin espacio.
- Alta fragmentación externa (4 huecos).
- Worst Fit deja muchos huecos pequeños.

4 Comparativo general de resultados

Entrada	Política	Procesos en espera	#Huecos finales	Espacio libre total	Fragmentación
input1.txt	First Fit	0	2	50	Moderada
input2.txt	Worst Fit	1 (A6)	3	50	Alta
input3.txt	Mixta	0	2	25	Baja
input4.txt	Best Fit	1 (B4)	1 (+intermedios)	20	Moderada
input5.txt	Worst Fit	2 (P7, P9)	4	55	Alta

5 Conclusiones comparativas

Algoritmo	Ventajas	Desventajas	Comportamiento observado
First Fit	Simple y rápido. Menor tiempo de búsqueda.	Puede dejar huecos pequeños al principio.	Eficiente en cargas moderadas, sin esperas.
Best Fit	Minimiza el desperdicio inmediato.	Genera muchos huecos pequeños.	Bueno para procesos pequeños, pero puede trabarse.
Worst Fit	Conserva huecos grandes para procesos futuros.	Fragmenta mucho y desperdicia espacio.	En la práctica, el menos eficiente.

6 Conclusión general

Durante la simulación de los dos modelos de asignación:

- La **memoria fija (Fixed Partitioned)** mostró simplicidad de gestión, pero desperdicia espacio cuando los procesos no llenan completamente las particiones (*fragmentación interna*).
- La **memoria dinámica (Dynamic Partitioned)** demostró mayor flexibilidad, pero requiere algoritmos de asignación eficientes para minimizar la *fragmentación externa*.

Entre los tres métodos:

- **Best Fit** fue el más eficiente en el caso teórico clásico (bloques grandes), optimizando el espacio.
- **First Fit** tuvo el mejor equilibrio en los escenarios dinámicos reales.
- **Worst Fit** produjo más esperas y fragmentación.

Por lo tanto, **First Fit** es la política más recomendable para entornos con alta variabilidad y asignaciones frecuentes, mientras que **Best Fit** puede ser útil en contextos controlados con procesos de tamaño similar.