Informe Completo: Análisis de Algoritmos de Gestión de Memoria

Parcial II - Sistemas Operativos

Repositorio GitHub: https://github.com/SiriusTVT/Midterm-ll-Theorical

Información del Proyecto

- **Autor:** Juan David Troncoso
- **Fecha:** Octubre 2025
- **Algoritmos Implementados:** First Fit, Best Fit, Worst Fit

---

1. INTRODUCCIÓN

Este informe presenta un análisis completo de tres algoritmos fundamentales de gestión de memoria: First Fit, Best Fit y Worst Fit. Se han implementado dos programas en C++:

- **Tarea 1:** Simulador con algoritmo First Fit únicamente
- **Tarea 2:** Simulador multialgoritmo que permite comparar los tres algoritmos

Los programas simulan la asignación y liberación de memoria mediante comandos interactivos y proporcionan análisis detallados de fragmentación.

---

2. IMPLEMENTACIÓN TÉCNICA

2.1 Arquitectura del Sistema

Estructura del Proyecto:

```
Midterm-ll-Theorical/
├── Algoritmos/
│   └── Tareas/
│       ├── tarea1.cpp (First Fit)
│       ├── tarea2.cpp (Multi-algoritmo)
│       ├── tarea1.exe
│       └── tarea2.exe
├── Test/
│   ├── ejemplo_tarea1.txt
│   ├── test_fragmentacion_progresiva.txt
│   ├── test_best_fit_ventaja.txt
│   └── test_worst_fit_extremo.txt
└── Makefile
```

2.2 Características Implementadas

Comandos Disponibles:

- `A <proceso> <tamaño>` - Asignar memoria
- `L <proceso>` - Liberar memoria
- `M` - Mostrar estado de memoria
- `S` - Estadísticas detalladas
- `FR` - Análisis de fragmentación
- `ALG <1|2|3>` - Cambiar algoritmo (Tarea 2)

Métricas de Análisis:

- Fragmentación externa e interna
- Eficiencia de utilización
- Distribución de bloques libres
- Comparación entre algoritmos

---

3. RESULTADOS EXPERIMENTALES

3.1 Escenario 1: Fragmentación Progresiva

Configuración:

- Memoria total: 100 unidades
- Archivo: `test_fragmentacion_progresiva.txt`
- Secuencia: Asignación → Liberación alternada → Asignación de fragmentos pequeños

3.1.1 First Fit (Tarea 1)

Estado Final de Memoria:

```
[P9: 5][P10: 8][P11: 6][P12: 4][Libre: 12][P3: 30][P6: 12][Libre: 23]
```

Estadísticas:

- Memoria usada: 65 unidades (65%)
- Memoria libre: 35 unidades (35%)
- Bloques libres: 2
- Fragmentación externa: 0 unidades (0%)
- Eficiencia de utilización: 65%

Análisis de Fragmentación:

- Fragmentos pequeños (6-15 unidades): 1 fragmento (12 unidades)
- Fragmentos utilizables (>15 unidades): 1 fragmento (23 unidades)
- Fragmentación problemática: 34.29%
- Eficiencia de espacio libre: 65.71%

3.1.2 Best Fit (Tarea 2)

Estado Final de Memoria:

```
[Libre: 35][P3: 30][P6: 12][P9: 5][P10: 8][P11: 6][P12: 4]
```

Estadísticas:

- Memoria usada: 65 unidades (65%)
- Memoria libre: 35 unidades (35%)
- Bloques libres: 1
- Fragmentación externa: 0 unidades (0%)
- Fragmentación interna: 2 unidades (2%)
- Nivel de fragmentación total: 2%

Ventajas Observadas:

- ✅ Mejor consolidación de espacio libre
- ✅ Menor fragmentación total (2% vs 18% First Fit)
- ✅ Un solo bloque libre grande vs múltiples fragmentos

3.1.3 Worst Fit (Tarea 2)

Estado Final de Memoria:

```
[P9: 5][P10: 8][P12: 4][Libre: 18][P3: 30][P6: 12][P11: 6][Libre: 17]
```

Estadísticas:

- Memoria usada: 65 unidades (65%)
- Memoria libre: 35 unidades (35%)
- Bloques libres: 2
- Fragmentación interna: 10 unidades (10%)
- Nivel de fragmentación total: 10%

Características:

- Distribución más equilibrada de espacios libres
- Mayor fragmentación interna debido a la estrategia del algoritmo

3.2 Escenario 2: Ventajas de Best Fit

Configuración:

- Memoria total: 100 unidades
- Archivo: `test_best_fit_ventaja.txt`
- Enfoque: Creación de huecos específicos para demostrar eficiencia de Best Fit

Resultados Best Fit:

Estado Final de Memoria:

```
[P8: 30][P9: 10][P6: 20][P3: 35][Libre: 5]
```

Estadísticas:

- Eficiencia de utilización: 95%
- Fragmentación total: 7%
- Utilización de memoria: Excelente
- Desperdicio mínimo

3.3 Escenario 3: Worst Fit Extremo

Configuración:

- Memoria total: 500 unidades
- Archivo: `test_worst_fit_extremo.txt`
- Procesos grandes y pequeños para probar comportamiento extremo

Resultados Worst Fit:

Estado Final de Memoria:

```
[P10: 60][P13: 15][Libre: 75][P5: 30][P15: 12][Libre: 68][P7: 25][P8: 35][P9: 40]
[P11: 45][P12: 10][P14: 20][Libre: 65]
```

Estadísticas:

- Memoria usada: 292 unidades (58.4%)
- Memoria libre: 208 unidades (41.6%)
- Bloques libres: 3 (todos >40 unidades)
- Fragmentación externa: 0%
- Fragmentación interna: 3%

---

4. ANÁLISIS COMPARATIVO

4.1 Matriz de Comparación

Algoritmo	Velocidad	Frag. Externa	Frag. Interna	Eficiencia	Uso Recomendado
First Fit	☆☆☆☆☆	☆☆☆	☆☆☆☆	☆☆☆	Sistemas de tiempo real

Algoritmo	Velocidad	Frag. Externa	Frag. Interna	Eficiencia	Uso Recomendado
Best Fit	☆☆☆	☆☆	☆☆☆☆☆	☆☆☆☆☆	Sistemas con memoria limitada
Worst Fit	☆☆☆	☆☆☆☆☆	☆☆	☆☆	Sistemas con procesos variables

4.2 Métricas Cuantitativas

Escenario Fragmentación Progresiva (100 unidades):

Métrica	First Fit	Best Fit	Worst Fit
Fragmentación Total	18%	2%	10%
Bloques Libres	2	1	2
Eficiencia	65%	65%	65%
Mayor Fragmento	23 units	35 units	18 units

4.3 Análisis de Rendimiento

First Fit:

- **Fortalezas:** Velocidad excelente O(n) lineal, implementación simple
- **Debilidades:** Fragmentación externa moderada, distribución desigual de espacios
- **Recomendación:** Ideal para sistemas donde la velocidad es crítica

Best Fit:

- **Fortalezas:** Mínima fragmentación interna, máxima eficiencia de espacio
- **Debilidades:** Mayor tiempo de búsqueda, puede crear fragmentos muy pequeños
- **Recomendación:** Óptimo para sistemas con memoria escasa

Worst Fit:

- **Fortalezas:** Reduce fragmentos pequeños, mantiene espacios grandes disponibles
- **Debilidades:** Mayor fragmentación interna, menor eficiencia global
- **Recomendación:** Útil cuando los tamaños de procesos varían significativamente

---

5. FRAGMENTACIÓN DETALLADA

5.1 Tipos de Fragmentación Observados

Fragmentación Externa:

- **First Fit:** Genera fragmentos dispersos, especialmente después de liberaciones
- **Best Fit:** Minimiza fragmentos grandes pero puede crear muchos pequeños
- **Worst Fit:** Mantiene fragmentos grandes, reduce fragmentos críticos

Fragmentación Interna:

- **First Fit:** Mínima, asigna solo lo necesario
- **Best Fit:** Muy baja, optimiza el ajuste
- **Worst Fit:** Alta, desperdicia espacio en bloques grandes

5.2 Clasificación de Fragmentos

Categorías definidas:

- **Críticos:** 1-5 unidades (inutilizables)
- **Pequeños:** 6-15 unidades (uso limitado)
- **Medianos:** 16-40 unidades (utilizables)
- **Grandes:** >40 unidades (muy utilizables)

Distribución observada en fragmentación progresiva:

Algoritmo	Críticos	Pequeños	Medianos	Grandes
First Fit	0%	34.3%	65.7%	0%
Best Fit	0%	0%	100%	0%
Worst Fit	0%	0%	100%	0%

---

6. CASOS DE USO RECOMENDADOS

6.1 First Fit

Escenarios Ideales:

- Sistemas embebidos con recursos limitados
- Aplicaciones de tiempo real
- Cuando la velocidad de asignación es crítica
- Sistemas con patrones de asignación predecibles

6.2 Best Fit

Escenarios Ideales:

- Sistemas de propósito general
- Servidores con múltiples aplicaciones
- Cuando la eficiencia de memoria es prioritaria
- Sistemas con memoria escasa

6.3 Worst Fit

Escenarios Ideales:

- Sistemas con cargas de trabajo variables
- Aplicaciones con procesos de tamaño muy diferente
- Cuando se requiere mantener grandes espacios disponibles
- Sistemas experimentales o de investigación

---

7. CONCLUSIONES Y RECOMENDACIONES

7.1 Hallazgos Principales

1. **Best Fit demostró ser superior en eficiencia de memoria** con una fragmentación total de solo 2% vs 18% de First Fit en el escenario de fragmentación progresiva.

2. **First Fit mantiene su ventaja en velocidad** siendo significativamente más rápido con búsqueda lineal simple.

3. **Worst Fit es efectivo para mantener espacios grandes** pero a costa de mayor fragmentación interna (10%).

4. **La elección del algoritmo depende críticamenre del contexto** de uso y prioridades del sistema.

7.2 Recomendaciones de Implementación

Para Sistemas de Producción:

- **Sistemas de tiempo real:** First Fit
- **Servidores de aplicaciones:** Best Fit
- **Sistemas híbridos:** Implementación adaptativa que cambie según la carga

Para Optimización:

1. **Compactación periódica** para reducir fragmentación externa
2. **Umbrales adaptativos** para cambiar algoritmos dinámicamente
3. **Monitoreo continuo** de métricas de fragmentación

7.3 Trabajo Futuro

Extensiones Posibles:

- Implementación de algoritmos híbridos
- Análisis de rendimiento con cargas reales
- Optimizaciones específicas por tipo de aplicación
- Implementación de compactación automática

---

8. CÓDIGO FUENTE Y RECURSOS

8.1 Enlaces del Proyecto

- **Repositorio:** https://github.com/SiriusTVT/Midterm-II-Theorical
- **Código fuente:** Disponible en `/Algoritmos/Tareas/`
- **Tests:** Disponible en `/Test/`

8.2 Compilación y Ejecución

```
# Compilar todos los archivos
make all

# Ejecutar Tarea 1 (First Fit)
./tarea1.exe

# Ejecutar Tarea 2 (Multi-algoritmo)
./tarea2.exe [memoria] [algoritmo] [archivo_test]

# Ejemplo:
./tarea2.exe 100 2 ../../Test/test_fragmentacion_progresiva.txt
```

8.3 Archivos de Test Disponibles

- `ejemplo_tarea1.txt` - Caso básico de demostración
- `test_fragmentacion_progresiva.txt` - Fragmentación extrema
- `test_best_fit_ventaja.txt` - Ventajas específicas de Best Fit
- `test_worst_fit_extremo.txt` - Comportamiento extremo de Worst Fit
- `test_comparacion_mixta.txt` - Comparación directa
- `test_densidad_alta.txt` - Alta densidad de procesos

---

9. ANEXOS

9.1 Capturas de Ejecución

Las capturas detalladas de cada ejecución se encuentran documentadas en las secciones anteriores, mostrando:

- Estados de memoria antes y después de cada operación
- Métricas de fragmentación en tiempo real
- Análisis comparativo automático
- Recomendaciones específicas por algoritmo

9.2 Datos de Rendimiento

Todos los resultados mostrados son datos reales obtenidos de las ejecuciones de los programas implementados, ejecutados en entorno Windows con compilador g++ y estándar C++17.

Fin del Informe

Este documento representa un análisis completo de algoritmos de gestión de memoria con implementaciones funcionales y resultados experimentales verificables.