PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO – PUC-RIO

DANIELA BRAZÃO MAKSOUD – 1321873
THAÍSSA FALBO – 1421110

SEGUNDO TRABALHO DE SISTEMAS DE COMPUTAÇÃO – INF1019

Gerenciador de Memória

"*os programas e seus dados aumentam de forma a ocupar toda a memória disponível para contê-los*".
Lei de Parkinson

Rio de Janeiro, 2015

RESUMO

Este trabalho apresenta os códigos fonte de quatro gerenciadores de memória distintos referentes aos algoritmos de alocação **First Fit (ajuste rápido)**, **Next Fit (próximo ajuste)**, **Best Fit (melhor ajuste)** e **Worst Fit (pior ajuste)**; um relatório indicando o comportamento dos algoritmos para uma entrada e o algoritmo que melhor se comportou, além de uma explicação se esse comportamento era ou não esperado.

Palavras-chave: Gerenciador, Memória, First, Next, Best, Worst, Fit.

SUMÁRIO

1 PROGRAMA PRINCIPAL

Segundo Tanenbaum e Woodhull (2008), o **gerenciador de memória** é “a parte do sistema operacional que gerencia a hierarquia de memória”. Sendo assim, “sua tarefa é monitorar as partes da memória que estão em uso e as que não estão, alocar memória para os processos quando eles precisarem dela e liberá-la quando terminam, e gerenciar a transferência (swapping) entre a memória principal e o disco, quando a memória principal for pequena demais para conter todos os processos”.

1.1 FIRST FIT

Segundo Tanenbaum e Woodhull (2008), o algoritmo de alocação **First Fit (ajuste rápido)** é aquele em que o “gerenciador de memória varre toda a lista de segmentos até localizar uma lacuna que seja suficientemente grande”, a “lacuna, então, é dividida em dois pedaços, um para o processo e um para a memória não-utilizada, exceto no improvável caso de um ajuste exato”. Sendo assim, “o algoritmo do primeiro ajuste é rápido porque pesquisa o mínimo possível”.

1.2 NEXT FIT

Segundo Tanenbaum e Woodhull (2008), o algoritmo de alocação **Next Fit (próximo ajuste)** é aquele que “funciona da mesma maneira que o primeiro ajuste, exceto que monitora a posição em que ele está sempre que encontra uma lacuna adequada” e “da próxima vez que é chamado para localizar uma lacuna, ele começa pesquisando na lista a partir do lugar que ele deixou da última vez, em vez de sempre a partir do começo, como o primeiro ajuste faz”.

1.3 WORST FIT

Segundo Tanenbaum e Woodhull (2008), o algoritmo de alocação **Worst Fit (pior ajuste)** é aquele que sempre pega “a maior lacuna disponível, de modo que a lacuna resultante seja suficientemente grande para ser útil”.

1.4 BEST FIT

Segundo Tanenbaum e Woodhull (2008), o algoritmo de alocação **Best Fit (melhor ajuste)** é aquele que “pesquisa na lista inteira e pega a menor lacuna que seja adequada” e “antes de dividir uma lacuna grande que talvez seja necessária mais tarde... tenta localizar uma lacuna que é próxima do tamanho real necessário”.

1.5 CÓDIGO FONTE

- Trabalho_2_INF1019/lista.h
- Trabalho_2_INF1019/lista.c
- Trabalho_2_INF1019/utilities.h
- Trabalho_2_INF1019/definitions.h
- Trabalho_2_INF1019/main.c

2 RELATÓRIO

2.1 FIRST FIT

Tendo como entrada o seguinte arquivo de texto:

entrada.txt

```
4
Processo #1 – 3Mb
4
exec 10
io 10
exec 10
io 20
Processo #2 – 4Mb
5
exec 10
io 10
exec 10
io 20
exec 20
Processo #3 – 1Mb
1
```

```
exec 20
Processo #4 – 7Mb
1
exec 10
1
```

A execução do algoritmo de alocação de memória de ajuste rápido (**First Fit**) produz o seguinte resultado:

1. A fila de prontos recebe todos os processos do arquivo de entrada.
2. Os processos **1**, **2** e **3** são alocados nos blocos de memória **1**, **2** e **3** respectivamente.
3. O processo **4** permanece na fila de prontos, pois não pôde ser alocado na memória principal.
4. O primeiro comando (**exec – 10s**) do primeiro processo (**Processo #1**) é executado por **10** segundos.
5. O primeiro comando (**exec – 10s**) do segundo processo (**Processo #2**) é executado por **10** segundos.
6. O primeiro comando (**exec – 20s**) do terceiro processo (**Processo #3**) é executado por **10** segundos.
7. O primeiro processo (**Processo #1**) entra na fila de bloqueados.
8. O segundo processo (**Processo #2**) entra na fila de bloqueados.
9. O primeiro comando (**exec – 20s**) do terceiro processo (**Processo #3**) é executado por mais **10** segundos.
10. O segundo comando (**io – 10s**) do primeiro processo (**Processo #1**) é executado em paralelo por **10** segundos.
11. O segundo comando (**io – 10s**) do segundo processo (**Processo #2**) é executado em paralelo por **10** segundos.
12. O terceiro comando (**exec – 10s**) do primeiro processo (**Processo #1**) é executado por **10** segundos.

13. O terceiro comando (**exec – 10s**) do segundo processo (**Processo #2**) é executado por **10** segundos.

14. O primeiro processo (**Processo #1**) entra na fila de bloqueados.

15. O segundo processo (**Processo #2**) entra na fila de bloqueados.

16. O quarto comando (**io – 20s**) do primeiro processo (**Processo #1**) é executado em paralelo por **10** segundos.

17. O quarto comando (**io – 20s**) do segundo processo (**Processo #2**) é executado em paralelo por **10** segundos.

18. O quarto comando (**io – 20s**) do primeiro processo (**Processo #1**) é executado em paralelo por mais **10** segundos.

19. O quarto comando (**io – 20s**) do segundo processo (**Processo #2**) é executado em paralelo por mais **10** segundos.

20. O quinto comando (**exec – 20s**) do segundo processo (**Processo #2**) é executado por **10** segundos.

21. O quinto comando (**exec – 20s**) do segundo processo (**Processo #2**) é executado por mais **10** segundos.

22. O processo 4 é alocado no bloco de memória 1.

23. O primeiro comando (**exec – 20s**) do quarto processo (**Processo #4**) é executado por **10** segundos.

24. O primeiro comando (**exec – 20s**) do quarto processo (**Processo #4**) é executado por mais **10** segundos.

2.2 NEXT FIT

Tendo como entrada o seguinte arquivo de texto:

entrada.txt

```
4
Processo #1 – 3Mb
4
exec 10
io 10
exec 10
io 20
Processo #2 – 4Mb
5
exec 10
io 10
exec 10
io 20
exec 20
Processo #3 – 1Mb
1
```

```
exec 20
Processo #4 – 7Mb
1
exec 10
2
```

A execução do algoritmo de alocação de memória de próximo ajuste (**Next Fit**) produz o seguinte resultado:

1. A fila de prontos recebe todos os processos do arquivo de entrada.
2. Os processos **1**, **2** e **3** são alocados nos blocos de memória **1**, **2** e **3** respectivamente.
3. O processo **4** permanece na fila de prontos, pois não pôde ser alocado na memória principal.
4. O primeiro comando (**exec – 10s**) do primeiro processo (**Processo #1**) é executado por **10** segundos.
5. O primeiro comando (**exec – 10s**) do segundo processo (**Processo #2**) é executado por **10** segundos.
6. O primeiro comando (**exec – 20s**) do terceiro processo (**Processo #3**) é executado por **10** segundos.
7. O primeiro processo (**Processo #1**) entra na fila de bloqueados.
8. O segundo processo (**Processo #2**) entra na fila de bloqueados.
9. O primeiro comando (**exec – 20s**) do terceiro processo (**Processo #3**) é executado por mais **10** segundos.
10. O segundo comando (**io – 10s**) do primeiro processo (**Processo #1**) é executado em paralelo por **10** segundos.
11. O segundo comando (**io – 10s**) do segundo processo (**Processo #2**) é executado em paralelo por **10** segundos.
12. O terceiro comando (**exec – 10s**) do primeiro processo (Processo #1) é executado por **10** segundos.

13. O terceiro comando (**exec – 10s**) do segundo processo (**Processo #2**) é executado por **10** segundos.

14. O primeiro processo (**Processo #1**) entra na fila de bloqueados.

15. O segundo processo (**Processo #2**) entra na fila de bloqueados.

16. O quarto comando (**io – 20s**) do primeiro processo (**Processo #1**) é executado em paralelo por **10** segundos.

17. O quarto comando (**io – 20s**) do segundo processo (**Processo #2**) é executado em paralelo por **10** segundos.

18. O quarto comando (**io – 20s**) do primeiro processo (**Processo #1**) é executado em paralelo por mais **10** segundos.

19. O quarto comando (**io – 20s**) do segundo processo (**Processo #2**) é executado em paralelo por mais **10** segundos.

20. O quinto comando (**exec – 20s**) do segundo processo (**Processo #2**) é executado por **10** segundos.

21. O quinto comando (**exec – 20s**) do segundo processo (**Processo #2**) é executado por mais **10** segundos.

22. O processo **4** é alocado no bloco de memória **1**.

23. O primeiro comando (**exec – 20s**) do quarto processo (**Processo #4**) é executado por **10** segundos.

24. O primeiro comando (**exec – 20s**) do quarto processo (**Processo #4**) é executado por mais **10** segundos.

2.3 WORST FIT

Tendo como entrada o seguinte arquivo de texto:

entrada.txt

```
4
Processo #1 – 3Mb
4
exec 10
io 10
exec 10
io 20
Processo #2 – 4Mb
5
exec 10
io 10
exec 10
io 20
exec 20
Processo #3 – 1Mb
1
```

```
exec 20
Processo #4 – 7Mb
1
exec 10
3
```

A execução do algoritmo de alocação de memória de pior ajuste (**Worst Fit**) produz o seguinte resultado:

1. A fila de prontos recebe todos os processos do arquivo de entrada.
2. Os processos **1**, **2** e **3** são alocados nos blocos de memória **1**, **2** e **3** respectivamente.
3. O processo **4** permanece na fila de prontos, pois não pôde ser alocado na memória principal.
4. O primeiro comando (**exec – 10s**) do primeiro processo (**Processo #1**) é executado por **10** segundos.
5. O primeiro comando (**exec – 10s**) do segundo processo (**Processo #2**) é executado por **10** segundos.
6. O primeiro comando (**exec – 20s**) do terceiro processo (**Processo #3**) é executado por **10** segundos.
7. O primeiro processo (**Processo #1**) entra na fila de bloqueados.
8. O segundo processo (**Processo #2**) entra na fila de bloqueados.
9. O primeiro comando (**exec – 20s**) do terceiro processo (**Processo #3**) é executado por mais **10** segundos.
10. O segundo comando (**io – 10s**) do primeiro processo (**Processo #1**) é executado em paralelo por **10** segundos.
11. O segundo comando (**io – 10s**) do segundo processo (**Processo #2**) é executado em paralelo por **10** segundos.
12. O terceiro comando (**exec – 10s**) do primeiro processo (**Processo #1**) é executado por **10** segundos.

13. O terceiro comando (**exec – 10s**) do segundo processo (**Processo #2**) é executado por **10** segundos.

14. O primeiro processo (**Processo #1**) entra na fila de bloqueados.

15. O segundo processo (**Processo #2**) entra na fila de bloqueados.

16. O quarto comando (**io – 20s**) do primeiro processo (**Processo #1**) é executado em paralelo por **10** segundos.

17. O quarto comando (**io – 20s**) do segundo processo (**Processo #2**) é executado em paralelo por **10** segundos.

18. O quarto comando (**io – 20s**) do primeiro processo (**Processo #1**) é executado em paralelo por mais **10** segundos.

19. O quarto comando (**io – 20s**) do segundo processo (**Processo #2**) é executado em paralelo por mais **10** segundos.

20. O quinto comando (**exec – 20s**) do segundo processo (**Processo #2**) é executado por **10** segundos.

21. O quinto comando (**exec – 20s**) do segundo processo (**Processo #2**) é executado por mais **10** segundos.

22. O processo **4** é alocado no bloco de memória **1**.

23. O primeiro comando (**exec – 20s**) do quarto processo (**Processo #4**) é executado por **10** segundos.

24. O primeiro comando (**exec – 20s**) do quarto processo (**Processo #4**) é executado por mais **10** segundos.

2.4 BEST FIT

Tendo como entrada o seguinte arquivo de texto:

entrada.txt

```
4
Processo #1 – 3Mb
4
exec 10
io 10
exec 10
io 20
Processo #2 – 4Mb
5
exec 10
io 10
exec 10
io 20
exec 20
Processo #3 – 1Mb
1
```

```
exec 20

Processo #4 – 7Mb

1

exec 10

4
```

A execução do algoritmo de alocação de memória de melhor ajuste (**Best Fit**) produz o seguinte resultado:

1. A fila de prontos recebe todos os processos do arquivo de entrada.
2. Os processos **1**, **2** e **3** são alocados nos blocos de memória **2**, **1** e **5** respectivamente.
3. O processo **4** permanece na fila de prontos, pois não pôde ser alocado na memória principal.
4. O primeiro comando (**exec – 10s**) do segundo processo (**Processo #2**) é executado por **10** segundos.
5. O primeiro comando (**exec – 10s**) do primeiro processo (**Processo #1**) é executado por **10** segundos.
6. O primeiro comando (**exec – 20s**) do terceiro processo (**Processo #3**) é executado por **10** segundos.
7. O segundo processo (**Processo #2**) entra na fila de bloqueados.
8. O primeiro processo (**Processo #1**) entra na fila de bloqueados.
9. O primeiro comando (**exec – 20s**) do terceiro processo (**Processo #3**) é executado por mais **10** segundos.
10. O segundo comando (**io – 10s**) do primeiro processo (**Processo #1**) é executado em paralelo por **10** segundos.
11. O segundo comando (**io – 10s**) do segundo processo (**Processo #2**) é executado em paralelo por **10** segundos.
12. O terceiro comando (**exec – 10s**) do segundo processo (**Processo #2**) é executado por **10** segundos.

13. O terceiro comando (**exec – 10s**) do primeiro processo (**Processo #1**) é executado por **10** segundos.

14. O segundo processo (**Processo #2**) entra na fila de bloqueados.

15. O primeiro processo (**Processo #1**) entra na fila de bloqueados.

16. O quarto comando (**io – 20s**) do primeiro processo (**Processo #1**) é executado em paralelo por **10** segundos.

17. O quarto comando (**io – 20s**) do segundo processo (**Processo #2**) é executado em paralelo por **10** segundos.

18. O quarto comando (**io – 20s**) do primeiro processo (**Processo #1**) é executado em paralelo por mais **10** segundos.

19. O quarto comando (**io – 20s**) do segundo processo (**Processo #2**) é executado em paralelo por mais **10** segundos.

20. O quinto comando (**exec – 20s**) do segundo processo (**Processo #2**) é executado por **10** segundos.

21. O quinto comando (**exec – 20s**) do segundo processo (**Processo #2**) é executado por mais **10** segundos.

22. O processo 4 é alocado no bloco de memória **1**.

23. O primeiro comando (**exec – 20s**) do quarto processo (**Processo #4**) é executado por **10** segundos.

24. O primeiro comando (**exec – 20s**) do quarto processo (**Processo #4**) é executado por mais **10** segundos.

2.5 CONCLUSÃO

O algoritmo que melhor se comportou foi o **Best Fit**, pois aproveitou melhor a memória na alocação dos processos em comparação com os demais algoritmos implementados. Esse comportamento já era esperado, pois nesse algoritmo a melhor partição, ou seja, aquela que o processo deixa o menor espaço sem utilização, é a escolhida para a alocação do processo.

Os algoritmos **First Fit**, **Next Fit** e **Worst Fit** obtiveram desempenho similar devido às partições fixas de memória e ao tamanho do quarto processo de 7 Mb que só pode ser alocado no primeiro bloco de memória de 8 Mb.

3 OBSERVAÇÕES

O modelo do arquivo de entrada foi alterado, incluindo-se em sua última linha o número referente à opção de execução do algoritmo de alocação de memória dentre os quatro implementados neste trabalho:

1. **First Fit**
2. **Next Fit**
3. **Worst Fit**
4. **Best Fit**

O trabalho completo e a sua respectiva documentação encontra-se disponível também no GitHub.

REFERÊNCIAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS – ABNT. **NBR 10719**: apresentação de relatórios técnico-científicos. Rio de Janeiro, 1989. 9 p.

TANENBAUM, A; WOODHULL, A. **Sistemas Operacionais: Projeto e Implementação**. Tradução João Tortello. 3. ed. Porto Alegre: Bookman, 2008.

Wikipedia. **Programar em C**. Disponível em <https://pt.wikibooks.org/wiki/Programar_em_C>. Acesso em: 11 de novembro 2015.

BIBLIOGRAFIA RECOMENDADA

GOOGLE CODE. **Mini-Shell**. Desenvolvida por BHARATHI. Disponível em: <https://code.google.com/p/mini-shell/>. Acesso em: 27 de setembro 2015.

TANENBAUM, A; WOODHULL, A. **Sistemas Operacionais: Projeto e Implementação**. Tradução João Tortello. 3. ed. Porto Alegre: Bookman, 2008.

Wikipedia. **Programar em C**. Disponível em <https://pt.wikibooks.org/wiki/Programar_em_C>. Acesso em: 11 de novembro 2015.