Universidade Federal de Itajubá

SRSC02 – Sistemas Operacionais

Luara Perilli - 202200841

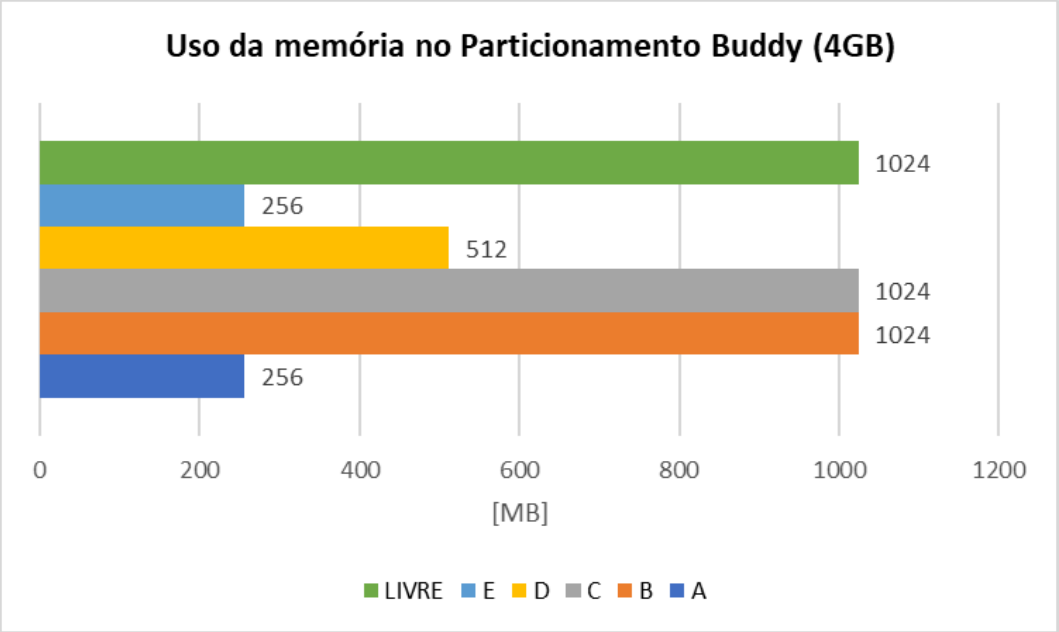
Guilherme Silva Teixeira - 2021033061

Exercício Prático 04 - EP04: Gerência da Memória Principal e Memória Virtual

a) Esboce o gráfico (Gráfico Gantt) que representa o uso da memória caso a seguinte sequência de requisições seja apresentada no sistema: A (130 MB), B (750 MB), C (600 MB), D (300 MB) e E (230 MB).

Resposta: O Particionamento Buddy é um esquema de alocação de memória que divide a memória em blocos de tamanho que são potências de dois. Ele tenta minimizar a fragmentação dividindo e combinando blocos de memória de maneira eficiente.

No particionamento Buddy, a memória total (4 GB = 4096 MB) é dividida em potências de dois. Quando um bloco é necessário, o algoritmo busca o menor bloco de potência de dois que seja suficiente para atender à requisição. Para o gráfico solicitado A ocupa 256MB, B ocupa 1024MB, C ocupa 1024MB, D ocupa 512MB E E ocupa 256 MB. Assim, o total alocado é 256 MB + 1024 MB + 1024 MB + 512 MB + 256 MB = **3072 MB**. O restante da memória (4096 MB - 3072 MB = **1024 MB**) estará livre.



b) É possível no particionamento Buddy haver fragmentação externa? Justifique.

Resposta: Não, o particionamento Buddy não causa fragmentação externa. No Buddy System, a fragmentação externa é evitada porque a memória é sempre dividida em blocos de tamanho potência de dois. No entanto, pode ocorrer fragmentação interna, pois um processo pode não utilizar todo o bloco que lhe foi atribuído, deixando sobras dentro do bloco.

c) Indique no gráfico, do item a) que representa o uso de memória onde seria carregado o processo X, de tamanho igual a 240 MB.

Resposta: O processo X (240 MB) precisa de um bloco de memória de tamanho 256 MB (a menor potência de dois que pode acomodar 240 MB). Assim, ele ocuparia um bloco de 256 MB. No gráfico do item a, o primeiro bloco livre que possa acomodar o processo seria aquele alocado para E, pois E ocupa um bloco de 256 MB e já está disponível para acomodar o processo X.

2) Um sistema utiliza alocação particionada dinâmica como mecanismo de gerência de memória. O sistema operacional aloca uma área de memória total de 50 KB e possui, inicialmente, os programas da tabela a seguir:

Tamanho	Status
5 KB	Processo A
3 KB	Processo B
10 KB	Livre
6 KB	Processo C
26 KB	Livre

Realize as operações abaixo, sequencialmente através de tabelas, mostrando o estado da memória após cada uma delas. Resolva a questão utilizando as estratégias best-fit, worst-fit e first-fit.

a) alocar área para o processo D que possui 6KB:

Resposta: Best-fit: encontra o menor espaço livre que acomode o processo. No caso, os blocos livres são os de 10KB e 26KB. Assim, o bloco de 10KB é o menor que pode acomodar o processo.

Worst-fit: encontra o maior espaço livre. No caso, os blocos livres são os de 10KB e 26KB. Assim, o bloco de 26KB é o menor que pode acomodar o processo.

First-fit: encontra o primeiro espaço livre que acomode o processo. Dessa forma, o bloco de 10 KB é o primeiro bloco que acomoda o processo.

b) liberar a área do programa A:

Resposta: Neste caso, o bloco onde o processo A está alocado é liberado. Assim, o bloco de 5 KB fica disponível.

c) alocar área para o processo E que possui 4 KB:

Resposta: Best-fit: o menor bloco livre que pode acomodar o processo E é o de 4KB.

Worst-fit: o maior bloco livre é de 20KB, então o processo E será alocado nele.

First-fit: encontra o primeiro espaço livre que acomode o processo. Assim, o bloco de 5 KB é o primeiro que pode acomodar o processo.

3) Considere que os processos da tabela a seguir estão aguardando para serem executados e que cada um permanecerá na memória pelo tempo especificado. O SO ocupa uma área de 20KB no início da memória e gerencia a memória utilizando um algoritmo de particionamento dinâmico modificado. A memória total disponível do sistema é de 64KB e é alocada em blocos

múltiplos de 4 KB. Os processos são alocados de acordo com a sua identificação (em ordem crescente), e irão aguardar até obter a memória de que necessitam. Calcule a perda de memória por fragmentação interna e externa sempre que um processo é retirado ou colocado na memória. O SO compacta a memória apenas quando existem duas ou mais partições livres adjacentes.

Processos	Memória	Tempo
1	30 KB	5
2	6 KB	10
3	36 KB	5

Resposta: Cálculo da fragmentação interna (arredondamento dos processos para blocos de 4 KB):

Processo 1:

- Memória solicitada: 30 KB
- Arredondado para múltiplo de 4 KB: 32 KB (8 blocos de 4 KB)
- Fragmentação interna: 32 KB - 30 KB = 2 KB

Processo 2:

- Memória solicitada: 6 KB
- Arredondado para múltiplo de 4 KB: 8 KB (2 blocos de 4 KB)
- Fragmentação interna: 8 KB - 6 KB = 2 KB

Processo 3:

- Memória solicitada: 36 KB
- Arredondado para múltiplo de 4 KB: 36 KB (9 blocos de 4 KB)
- Fragmentação interna: 36 KB - 36 KB = 0 KB (nenhuma fragmentação interna)

Fragmentação externa:

De acordo com a descrição, não há fragmentação externa mencionada. Isso ocorre porque o sistema compacta a memória sempre que existem duas ou mais partições livres adjacentes. Portanto, todas as partições livres são consolidadas, evitando lacunas que causam fragmentação externa.

Perda total de memória por fragmentação:

- Total de fragmentação interna:
- Processo 1: 2 KB

- Processo 2: 2 KB
- Processo 3: 0 KB

- Total de fragmentação externa: Não ocorre devido à compactação automática.
- Perda total de memória por fragmentação: 4 KB (apenas por fragmentação interna).

4) Pesquise sobre a anomalia de Belady, descreva resumidamente sobre esse fenômeno.

Resposta: A Anomalia de Belady é um fenômeno observado em alguns algoritmos de substituição de páginas na memória, como o FIFO (First-In, First-Out). Normalmente, espera-se que o aumento do número de quadros de página (espaço de memória disponível) resulte em uma diminuição no número de faltas de página (page faults). No entanto, a anomalia de Belady mostra que, para o algoritmo FIFO, em alguns casos, aumentar o número de quadros pode, paradoxalmente, aumentar o número de faltas de página, contrariando a intuição.

Isso acontece porque o FIFO remove as páginas mais antigas sem considerar sua relevância futura, o que pode fazer com que páginas importantes sejam removidas e causem mais faltas quando forem necessárias novamente. Este comportamento foi descoberto e documentado por Laszlo Belady em 1969.

5) Considere a tabela de páginas mostrada na tabela abaixo para um sistema com endereços virtuais e físicos de 12 bits e páginas de 256 bytes. A lista de quadros de páginas livres é D, E e F (isto é, D é a cabeça da lista, E é o segundo quadro e F é o último).

Página	Quadro da página
0	C
1	5
2	-
3	A
4	-
5	2
6	7
7	-
8	0
9	3

Converta os endereços virtuais, a seguir, nos endereços físicos equivalentes em hexadecimais. Todos os números são dados em hexadecimal. (Um travessão para um quadro de página indica que a página não está em memória.). Mostre os cálculos que o MMU fará para encontrar o endereço físico.

a) 0x9EF:

Resposta:

- Endereço binário: 1001 1110 1111
- Número da página: 1001 (Página 9)
- Deslocamento: 11101111 (Deslocamento: 0xEF)
- A página 9 não está na tabela de páginas fornecida, então este endereço virtual causaria uma falha de página.

b) 0x111:

Resposta:

- Endereço binário: 0001 0001 0001
- Número da página: 0001 (Página 1)
- Deslocamento: 00010001 (Deslocamento: 0x11)
- A página 1 também não está carregada na memória. Portanto, ocorre uma falha de página.

c) 0x700:

Resposta:

- Endereço binário: 0111 0000 0000
- Número da página: 0111 (Página 7)
- Deslocamento: 00000000 (Deslocamento: 0x00)
- A página 7 também não está mapeada, logo ocorre uma falha de página.

d) 0x0FF:

Resposta:

- Endereço binário: 0000 1111 1111
- Número da página: 0000 (Página 0)
- Deslocamento: 11111111 (Deslocamento: 0xFF)
- A página 0 está mapeada no quadro C. Para encontrar o endereço físico, concatenamos o quadro com o deslocamento:
- Quadro: C (hexadecimal)
- Endereço físico: CFF (hexadecimal)

e) 0x275:

Resposta:

- Endereço binário: 0010 0111 0101
- Número da página: 0010 (Página 2)
- Deslocamento: 01110101 (Deslocamento: 0x75)
- A página 2 está mapeada no quadro 0. O quadro 0 é concatenato com o deslocamento:
- Quadro: 0
- Endereço físico: 075 (hexadecimal)

f) 0x532:

Resposta:

- Endereço binário: 0011 0011 1010
- Número da página: 0011 (Página 3)
- Deslocamento: 00111010 (Deslocamento: 0x3A)
- A página 3 não está mapeada na memória, portanto ocorre uma falha de página.

6) Considere a sequência de referências de página a seguir: 1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6. Quantas faltas de página ocorreriam para os algoritmos de substituição abaixo, supondo a existência de um, dois, três, quatro, cinco, seis e sete quadros? Lembre-se de que todos os quadros estão inicialmente vazios e, portanto, as primeiras páginas apresentarão uma falta de página cada. Discuta os algoritmos.

a) Substituição LRU:

Resposta:

1 → Falta de página (Quadros: 1, -, -)

2 → Falta de página (Quadros: 1, 2, -)

3 → Falta de página (Quadros: 1, 2, 3)

4 → Falta de página, substitui a página 1 (Quadros: 4, 2, 3)

1 → Falta de página, substitui a página 2 (Quadros: 4, 1, 3)

2 → Falta de página, substitui a página 3 (Quadros: 4, 1, 2)

5 → Falta de página, substitui a página 4 (Quadros: 5, 1, 2)

1 → Não há falta de página (Quadros: 5, 1, 2)

2 → Não há falta de página (Quadros: 5, 1, 2)

3 → Falta de página, substitui a página 5 (Quadros: 3, 1, 2)

4 → Falta de página, substitui a página 1 (Quadros: 3, 4, 2)

5 → Falta de página, substitui a página 2 (Quadros: 3, 4, 5)

Total de faltas de página (LRU): 9

b) Substituição FIFO:

Resposta:

1 → Falta de página (Quadros: 1, -, -)

2 → Falta de página (Quadros: 1, 2, -)

3 → Falta de página (Quadros: 1, 2, 3)

4 → Falta de página, substitui a página 1 (Quadros: 4, 2, 3)

1 → Falta de página, substitui a página 2 (Quadros: 4, 1, 3)

2 → Falta de página, substitui a página 3 (Quadros: 4, 1, 2)

5 → Falta de página, substitui a página 4 (Quadros: 5, 1, 2)

1 → Não há falta de página (Quadros: 5, 1, 2)

2 → Não há falta de página (Quadros: 5, 1, 2)

3 → Falta de página, substitui a página 1 (Quadros: 5, 3, 2)

4 → Falta de página, substitui a página 2 (Quadros: 5, 3, 4)

5 → Não há falta de página (Quadros: 5, 3, 4)

Total de faltas de página (FIFO): 8

c) Substituição ótima:

Resposta: A substituição ótima substitui a página que será usada mais tarde ou a que não será usada no futuro.

1 → Falta de página (Quadros: 1, -, -)

2 → Falta de página (Quadros: 1, 2, -)

3 → Falta de página (Quadros: 1, 2, 3)

4 → Falta de página. A próxima página a ser usada mais tarde é a 1, então substitui a página 1 (Quadros: 4, 2, 3)

1 → Falta de página. A página 2 será usada mais tarde, então substitui a página 2 (Quadros: 4, 1, 3)

2 → Falta de página. A página 3 será usada mais tarde, então substitui a página 3 (Quadros: 4, 1, 2)

5 → Falta de página. A página 4 não será mais usada, então substitui a página 4 (Quadros: 5, 1, 2)

1 → Não há falta de página (Quadros: 5, 1, 2)

2 → Não há falta de página (Quadros: 5, 1, 2)

3 → Falta de página. A página 5 será usada mais tarde, então substitui a página 5 (Quadros: 3, 1, 2)

4 → Falta de página. A página 1 será usada mais tarde, então substitui a página 1 (Quadros: 3, 4, 2)

5 → Falta de página. A página 2 será usada mais tarde, então substitui a página 2 (Quadros: 3, 4, 5)

Total de faltas de página (Ótimo): 8

7) Uma visão simplificada de estados dos threads é Pronto, Em Execução e Bloqueado, em que um thread está pronto e esperando para ser incluído no schedule, está sendo executado no processador, ou está bloqueado (por exemplo, está esperando por I/O). Isso é ilustrado na figura abaixo.

Supondo que um thread esteja no estado Em Execução, responda às perguntas a seguir e explique sua resposta:

a) O thread mudará de estado se incorrer em um erro de página? Se mudar, para que estado passará?

Resposta: Sim, o thread mudará de estado se incorrer em um erro de página (page fault). Um erro de página ocorre quando o thread tenta acessar uma página que não está presente na memória física (RAM), mas está armazenada no disco. Assim, o thread será movido para o estado "Bloqueado". Quando ocorre um erro de página, o sistema operacional precisa carregar a página faltante do disco para a memória. Isso é uma operação de I/O (entrada/saída), e como a execução do thread não pode continuar até que a página seja carregada, ele fica bloqueado aguardando a conclusão dessa operação.

b) O thread mudará de estado se uma referência de endereço for resolvida na tabela de páginas? Se mudar, para que estado passará?

Resposta: Não, o thread não mudará de estado se uma referência de endereço for resolvida na tabela de páginas corretamente. Se a referência de endereço for resolvida na tabela de páginas, significa que a página já está presente na memória física. Nesse caso, o thread pode continuar sua execução sem interrupção. Como não há necessidade de buscar dados do disco (nenhum erro de página ocorreu), o thread permanece no estado "Em Execução" e continua processando suas instruções normalmente.