17/08/21

7,5

Observação: Os valores e as respostas corretas de todas as questões encontram-se no final deste arquivo.

"Eu, Davi Augusto Neves Leite, 1910273 83, declaro que esta prova reflete o meu conhecimento sobre o conteúdo da disciplina Sistemas Operacionais II e declaro que não houve qualquer comunicação com os demais alunos da turma nem consulta a qualquer material não autorizado durante o período de realização desta prova."

Ass: Davi A. Neves Leite

P2 - S.O. II

0,75 Esqueceu de considerar que o espaço de usuário é paginado!

(1) Considerando as abordagens (a) e (b), é conveniente afirmar que cada uma é eficiente dependendo do tipo de sistema em que são aplicadas.

Em outras palavras, a abordagem (a) permite o uso de E/S assíncrona?, mas como só existe 1 buffer por processo somente é possível a existência de sistemas uniprogramados?

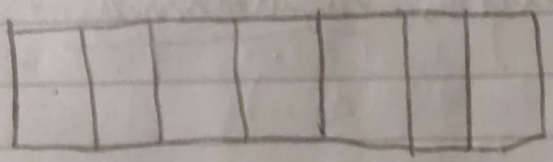
Em contrapartida, a abordagem (b) retrata a existência de buffers tanto por processo quanto por driver?, permitindo a existência da multiprogramação uma vez que o Device Driver deve ser reentrante. Nesta abordagem, a informação do buffer por driver é passada ao dispositivo quando possível.

Veja a correção no final do arquivo.

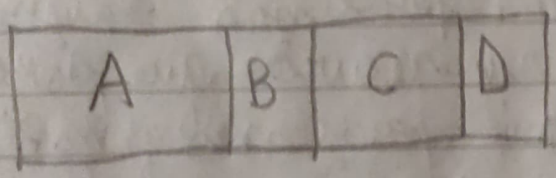
/ /

Em síntese, se considerarmos os sistemas atuais, a abordagem (b) é mais eficiente por permitir a multiprogramação.

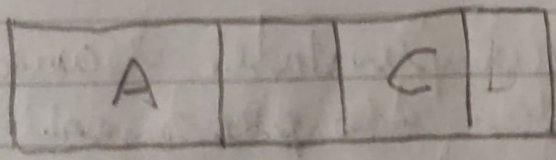
0,75 (2) A alocação contígua representa o armazenamento dos dados em áreas (blocos) adjacentes do disco. Desta forma, a fragmentação é dada da seguinte maneira (considerando um disco vazio):



→ Inicial, com 7 blocos.



→ Criação de 4 arquivos, com blocos de 3, 1, 2 e 1, respectivamente



→ Situação após retirar o 2º e 4º arquivo.

Pela última esquema acima, supondo que deseja-se criar um arquivo com 2 blocos. Isto não será possível, pois não há blocos contíguos disponíveis.

Já a alocação não contígua representa o armazenamento de dados em áreas adjacentes ou não do disco. Em outras palavras, um arquivo é representado por um conjunto de blocos ligados logicamente no disco,

tilibra

fragmentação interna?
fragmentação externa?

independente da localização física. Desta forma, o problema da fragmentação também vai ocorrer, mas somente de maneira física (ao contrário da contígua que é física e lógica), uma vez que os blocos são ligados entre si (como uma lista) de maneira lógica.

(3) a) Para a ocorrência de deadlock, são necessárias quatro condições simultâneas. São elas:

0,5

I) Exclusão mútua: processos exigem controle exclusivo sobre o recurso que solicitam;
II) Retenção e Espera: processos mantêm diversos recursos alocados enquanto solicitam novos recursos;
III) Ausência de Preemptividade: recursos não podem ser retirados dos processos enquanto estes não finalizarem o uso;
IV) Espera Circular: formação de uma cadeia circular de processos, cada qual solicitando o recurso alocado ao próximo da cadeia.

b) Analisando o cenário descrito com as quatro condições do deadlock, tem-se.

I) Exclusão mútua: por se tratar de recursos não preemptíveis (plotter e impressora), essa condição está presente para o cenário descrito, uma vez que cada processo solicitou um recurso (A com R; B com S e C com S). 0,75

II) Retenção e Espera: condição presente para o cenário, uma vez que os processos detinham recursos e solicitaram novos.

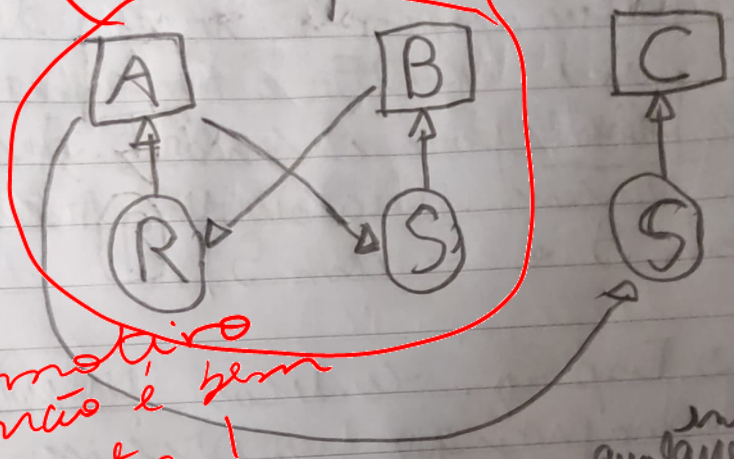
III) Ausência de Preemptividade: condição presente para o cenário, já que os recursos são, o plotter e duas impressoras.

IV) Espera Circular: ausente para o cenário descrito, uma vez que C não solicitou o recurso alocado ao próximo da cadeia. Mais explicações na letra (c). → a espera circular não precisa envolver todos os processos do sistema!

c) Se considerarmos o sub-sistema entre A e B, ocorre um deadlock, haja vista a existência das quatro condições simultâneas. Contudo, o sistema como um todo não ocorrerá deadlock, uma vez que a existência do processo C (com um Recurso de S (solicitado por A)) não solicitou um? recurso alocado por qualquer 0,25

→ isto é uma cadeia circular!

clientes pedem. Ou seja, a cadeia circular não foi formada. Isso pode ser visto no esquema abaixo:



} A solicita um recurso S, o qual tem duas instâncias (pode ser qualquer um).

e motivo não é bem este!

A existência dessa conexão e a inexistência de conexão entre o processo C e algum dos outros recursos? impede? o Deadlock no cenário descrito.

(4) a) → Um bloco = 2^{12} B

→ Número total de blocos = $\frac{(2^8 \cdot 2^{30}\ B)}{(2^{12}\ B)}$

$= 2^{26}$ blocos

1,0

⇒ Número de blocos livres pelo bitmap

BITMAP = $\frac{2^{26}\ \text{blocos}}{2^{12} \cdot 2^3}$ = 2^{11} blocos livres

= 2048 blocos livres

b) → Um bloco = 2^{12} B e 2^2 B de endereço
→ Número total de blocos = 2^{26} blocos

1,0

→ Número de blocos livres pela lista ligada

$LL = (2^{26}) \div \left[\begin{matrix}(2^{12})\,B \\ (2^2)\,B \text{ endereço}\end{matrix}\right] = 2^{16}$ blocos livres
$= 65536$ blocos

5)

Arquivo	Entrada na FAT
A	6
B	21
C	18

1,0

FAT:

0	
1	2
2	16
3	5
4	7
5	-1
6	12
7	8
8	13
9	
10	
11	17
12	11
13	-1
14	-1
15	14
16	15
17	3
18	20
19	4
20	1
21	19

1,5

6. PTR_DESC_PROC
   - sem→Q = prim
   - aux → fila_sem != NULL
   - aux → fila_sem = prim
   - sem→Q != NULL
   - p_aux = sem→Q
   - sem→S++

Valores das questões

1) 1,5 2) 1,5 3) (a) 0,5 (b) 1,0 (c) 1,0	4) (a) 1,0 (b) 1,0 5) 1,0 6) 1,5

RESPOSTAS – PROVA 2 – SO II – BCC

Resposta da questão 1

A forma mais eficiente é a ilustrada na situação (b) da figura.
Na situação (a), o buffer está localizado no espaço do processo do usuário. O software de E/S do S.O. (normalmente uma rotina de interrupção), ao receber os bytes provenientes do dispositivo, os escreverá diretamente no buffer do usuário e desbloqueará o processo. Porém, o problema, neste caso é que o buffer poderá pertencer a uma página que está sujeita a ser removida enquanto os bytes estiverem chegando, conforme visto quando estudamos a Paginação. Se a página que contém o buffer do processo for removida, quando chegarem bytes do dispositivo tem-se uma perda de dados, pois o S.O. não terá como armazenar temporariamente os dados. Este problema é contornado na situação (b). Na situação (b), o S.O. armazenará temporariamente os dados que chegam do dispositivo em seu próprio buffer no espaço do núcleo e, quando este estiver cheio, os copiará para o buffer no espaço do usuário, somente quando a página que o contém estiver presente na memória.

Resposta da questão 2

O problema da fragmentação ocorre nos dois tipos de alocação, porém não de maneira idêntica.

Na **alocação contígua**, tem-se o aparecimento da **fragmentação externa**, quando, por exemplo um arquivo é removido (por ter sido deletado ou aumentado de tamanho), o que poderá deixar uma lacuna (composta por um certo número de blocos consecutivos) entre áreas ocupadas. Esta lacuna só poderá ser ocupada por arquivos de tamanho igual ou inferior ao do arquivo removido. Na **alocação contígua** também tem-se a **fragmentação interna**, pois dificilmente um arquivo usará exatamente um número inteiro de blocos. Normalmente **a fragmentação interna** ocorrerá **no último bloco que compõe o arquivo**.

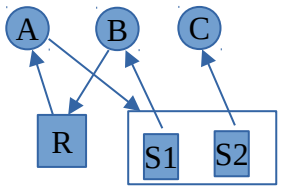
Na alocação **não contígua**, **não ocorrerá a fragmentação externa**, pois qualquer bloco livre (criado pela remoção de um arquivo) poderá ser usado para fazer parte de um novo arquivo. No entanto, a **fragmentação interna poderá ocorrer, normalmente no último bloco que compõe o arquivo.**

Resposta da questão 3

a) As 4 condições são: exclusão mútua no acesso ao recurso; condição de espera por recurso; ausência de preemptividade e espera circular.

b) Sim. As 4 condições estão presentes no cenário. A, B e C têm acesso exclusivo aos recursos R e as duas instâncias de S, respectivamente; A e B estão bloqueados na espera pelos recursos S e R, respectivamente; os recursos R e S são não preemptíveis e há uma cadeia de espera circular formada pelos processos A e B e pelos recursos R e S.

c) no cenário descrito tem-se:



Não ocorreu um deadlock. Quando C, que não está bloqueado, terminar de usar a instância do recurso S (no caso S2), esta é liberada e alocada ao processo A. Agora A é desbloqueado e pode concluir sua execução e liberar R que pode ser alocado para B, que também pode concluir sua execução. Portanto, pode-se concluir que as 4 condições são suficientes para ocorrência de um deadlock, desde que haja somente um recurso de cada tipo.

Resposta da questão 4

(a) Tem-se que:
Quantidade de blocos no disco = 256GB / 4KB = $(2^8 * 2^{30}) / 2^{12} = 2^{26}$ blocos

Como no bitmap tem-se um bit para representar a situação de alocação de cada bloco, conclui-se que:

tamanho do bitmap = 2^{26} bits, ou convertido em bytes: $2^{26} / 2^3$ bytes = 2^{23} bytes

Como um bloco tem tamanho de 4KB, o **tamanho do bitmap em blocos** será $2^{23} / 2^{12}$ = **2^{11} blocos (ou 2048 blocos).**

(b) Tem-se que:
Quantidade de blocos no disco = 256GB / 4KB = $(2^8 * 2^{30}) / 2^{12} = 2^{26}$ blocos

Em um bloco podem ser armazenados 4K/4 = $2^{12} / 2^2 = 2^{10}$ endereços de bloco

Portanto, o **tamanho da lista ligada** será: $2^{26} / 2^{10}$ **= 2^{16} blocos (ou 65536 blocos)**

5. Considere que os dados dos arquivos A, B e C estão armazenados nos blocos descritos no quadro abaixo.

Arquivo	**Sequência dos blocos de dados do arquivo**
A	6, 12, 11, 17, 3, 5
B	21, 19, 4, 7, 8, 13
C	18, 20, 1, 2, 16, 15, 14

Preencha os campos das entradas do diretório e da Tabela de Alocação de Arquivos (FAT) abaixo.
Use o valor -1 para representar o final de arquivo (EOF).

Entradas do diretório

Nome do arquivo	**Demais campos (não precisa preencher)**	**Entrada na FAT**
A	...	6
B	...	21
C	...	18

FAT

0	–
1	2
2	16
3	5
4	7
5	-1
6	12
7	8
8	13
9	–
10	–
11	17
12	11
13	-1
14	-1
15	14
16	15
17	3
18	20
19	4
20	1
21	19

Resposta da questão 6

```
void far insere_filaQ(semaforo *sem)
{
  PTR_DESC_PROC aux;
  if (sem->Q == NULL) sem->Q = prim;
    else {
          aux = sem->Q;
          while ( aux->fila_sem != NULL)
               aux = aux->fila_sem;
        aux->fila_sem = prim;
        }
  prim->fila_sem = NULL;
}
```

```
void far V(semaforo *sem)
{ PTR_DESC_PROC p_aux;
  disable();
  if ( sem->Q != NULL ) {
                    p_aux = sem->Q;
                    sem->Q = p_aux->fila_sem;
                    p_aux->fila_sem = NULL;
                    p_aux->estado = ativo;
                    }
               else sem->s++;
  enable();
}
```