Curso de Tecnologia em Sistemas de Computação
Disciplina de Sistemas Operacionais
Professores: Valmir C. Barbosa e Felipe M. G. França
Assistente: Alexandre H. L. Porto

Quarto Período
Gabarito da AP1 - Segundo Semestre de 2018

Nome -
Assinatura -

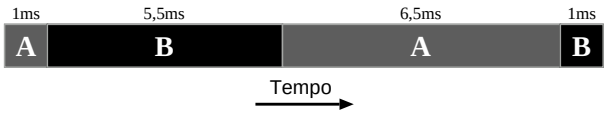
Observações:

1. Prova sem consulta e sem uso de máquina de calcular.
2. Use caneta para preencher o seu nome e assinar nas folhas de questões e nas folhas de respostas.
3. Você pode usar lápis para responder as questões.
4. Ao final da prova devolva as folhas de questões e as de respostas.
5. Todas as respostas devem ser transcritas nas folhas de respostas. As respostas nas folhas de questões não serão corrigidas.

---

1. (1,5) Suponha que um programa A precise executar no processador por 7,5ms e que, durante a sua execução, faça uma operação de E/S de 4,5ms de duração. Suponha ainda que um programa B precise executar no processador por 6,5ms e que, durante a sua execução, faça uma operação de E/S de 5,5ms de duração. Se a multiprogramação for usada somente para evitar a ociosidade do processador quando operações de E/S são feitas, e supondo que A seja o primeiro a iniciar sua execução, por quanto tempo A e B deverão executar, no mínimo e no máximo, antes de realizarem suas respectivas operações de E/S, a fim de não haver qualquer ociosidade? Justifique a sua resposta.

   Resp.: Suponha que o programa A tenha feito a sua operação de E/S após a ms, e que o programa B tenha feito a sua operação de E/S após b ms. Como A inicia sua execução antes de B, então para evitar a ociosidade do processador quando a operação de E/S de A for feita, B deverá executar por pelo menos 4,5ms, o tempo da operação de E/S feita por A, antes de fazer a sua operação de E/S, implicando em $b \geq 4,5$ms. Além disso, para evitar a ociosidade do processador ao B fazer a sua operação de E/S, o tempo restante de execução de A no processador, após fazer a sua operação de E/S, deverá ser pelo menos igual ao tempo da operação de E/S feita por B, implicando em $7,5 - a \geq 5,5$ms, ou seja, $a \leq 2,0$ms. Logo, A deverá executar no processador, antes de fazer a sua operação de E/S, por no mínimo 0ms e por no máximo 2,0ms, e B deverá executar no processador, antes de fazer a sua operação de E/S, por no mínimo 4,5ms e no por máximo 6,5ms. Como ilustração, na figura a seguir, mostramos o caso para $a = 1,0$ms e $b = 5,5$ms.



2. (2,5) Diga se as seguintes afirmativas são falsas ou verdadeiras. Para responder, escreva apenas F ou V para cada item em seu caderno de respostas.

(a) (0,5) Na primeira geração de computadores, os usuários utilizavam o computador de maneira exclusiva e eram os responsáveis por seu gerenciamento, diferentemente da segunda geração, na qual os usuários nem gerenciavam o computador nem o usavam com exclusividade.

Resp.: V (Verdadeira).

(b) (0,5) Um arquivo especial de bloco é usado para acessar um dispositivo mapeado em um conjunto de blocos consecutivos da memória.

Resp.: F (Falsa), porque um arquivo especial de bloco é usado para acessar os dispositivos representados por um conjunto de blocos que podem ser acessados aleatoriamente.

(c) (0,5) Um sistema operacional em camadas é aquele estruturado como uma hierarquia de camadas, onde cada uma implementa uma parte do sistema operacional e somente pode acessar os serviços fornecidos por uma camada inferior a ela.

Resp.: V (Verdadeira).

(d) (0,5) A exclusão mútua evita a ocorrência de condições de corrida, impedindo que dois ou mais processos acessem simultaneamente as suas seções críticas, nos casos em que tais acessos possam afetar a sua correta execução.

Resp.: V (Verdadeira).

(e) (0,5) Quando uma classe de prioridades é usada como alternativa ao escalonador por prioridades, os processos dentro de uma classe são escalonados usando o algoritmo do trabalho mais curto primeiro.

Resp.: F (Falsa), pois os processos dentro de uma das classes são escalonados de acordo com o escalonador por *round robin*.

3. (1,5) Diga a quais conceitos vistos em aula se referem as seguintes definições:

   (a) (0,5) Conceito, desenvolvido na terceira geração de computadores, que permite aos programas se alternarem no uso do processador, com cada um deles usando o processador por um dado intervalo de tempo, levando a que se possa tentar evitar a ociosidade do processador quando operações de E/S são feitas.

   Resp.: Multiprogramação.

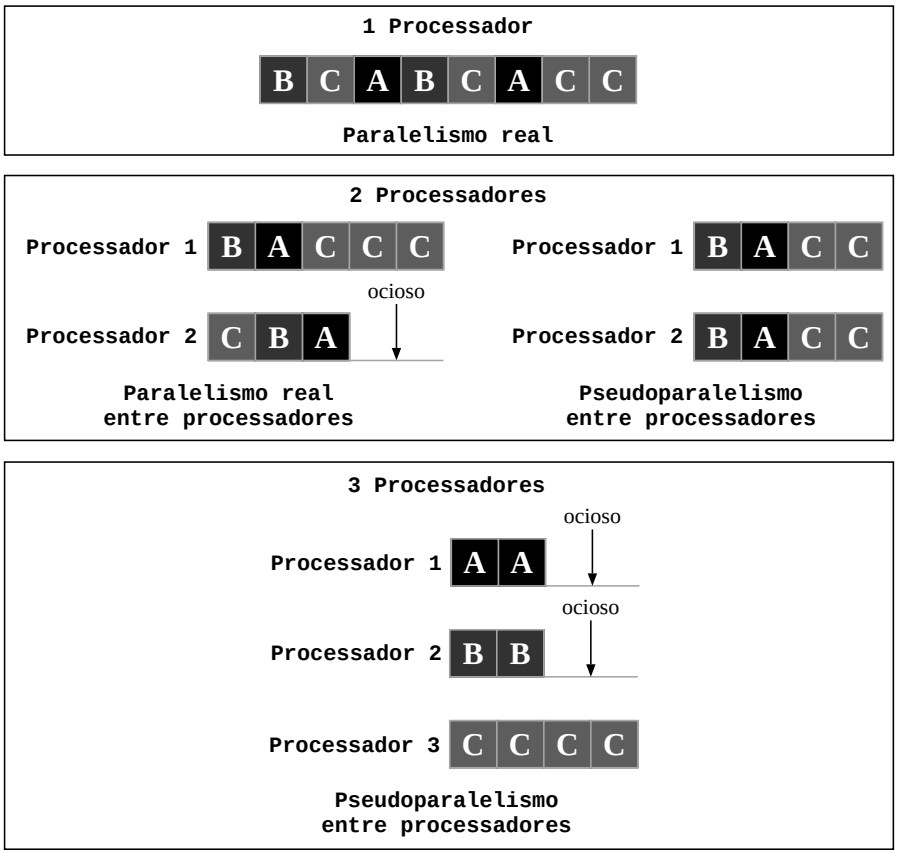
   (b) (0,5) Nome dado ao grafo cujos vértices são os possíveis estados em que um processo pode estar, e cujas arestas são as possíveis transições entre esses estados.

   Resp.: Diagrama de transições.

   (c) (0,5) Técnica para garantir que um processo somente acesse recursos compartilhados com outros processos quando todas as condições necessárias para a sua execução tenham sido satisfeitas.

   Resp.: Sincronização.

4. (1,5) Um aluno afirmou que cada ordem de execução e tipo de paralelismo dados nas figuras a seguir estão corretos, supondo que os processos executem em um sistema operacional que não permita a execução de partes paralelas de um mesmo processo. Todas as afirmações do aluno estão corretas? Se você acha que sim, basta dizer isso mas, se você acha que não, diga quais afirmações estão erradas.



Resp.: Quatro das afirmações do aluno não estão corretas, porque existe um erro na primeira figura, dois erros na segunda figura, e um erro na terceira figura. Como vimos na aula 4, o pseudoparalelismo ocorre quando um processador executa mais de um processo, e o paralelismo real ocorre quando processos distintos são executados por processadores distintos. Logo, na figura que mostra a execução com um processador, temos pseudoparalelismo. Na figura que mostra a execução com dois processadores, temos uma situação mista nas duas partes (esquerda e direita), pois ambas mostram paralelismo real entre processadores e pseudoparalelismo em cada processador. Assim, não há qualquer erro na parte esquerda. Já na parte direita, há dois erros: o paralelismo mostrado é proibido pelo enunciado e além disso é real. Na figura que mostra a execução com três processadores, temos paralelismo real entre processadores.

5. (1,5) Suponha que dois processos, A e B, compartilhem uma matriz $m \times n$, inicialmente com todos os elementos com o valor 0. O processo A continuamente coloca um valor diferente de 0 em um elemento com valor igual a 0 da matriz, escolhido de modo aleatório, se existir tal elemento. Já o processo B continuamente escolhe dois elementos com valores diferentes de 0 da matriz, de modo aleatório, caso existam, e depois copia o valor 0 para os elementos escolhidos. Como três semáforos, um binário e dois de contagem, podem ser usados para garantir que os processos executem corretamente, sem que ocorram condições de corrida? Justifique a sua resposta.

   Resp.: A seguir mostramos como os três semáforos podem ser usados para implementar os códigos dos processos A e B. O semáforo binário, chamado *acesso*, é usado para garantir o acesso exclusivo à matriz. O primeiro semáforo de contagem, chamado *zeros*, conta o número de elementos com valores iguais a 0 na matriz e é usado para bloquear A quando nenhum elemento da matriz tem valor igual a 0. Finalmente, o segundo semáforo de contagem, chamado *naozeros*, conta o número de elementos com valores diferentes de 0 na matriz e é usado para bloquear B quando não existem pelo menos dois elementos com valores diferentes de 0 na matriz. Como todos os elementos da matriz são inicialmente iguais a 0, como temos mn elementos porque a matriz tem m linhas e n colunas, e como a matriz está inicialmente disponível para uso, então os semáforos *naozeros*, *zeros* e *acesso* são inicializados, respectivamente, com 0, mn e 1. A seguir mostramos os pseudocódigos para os processos A e B, usando os semáforos descritos.

```
void ProcessoA(void)
{
  while(1)
  {
    // Código para gerar um valor x diferente de 0 a ser colocado na matriz.
    // Usa a operação P sobre zeros para garantir que exista pelo menos um
    // elemento com valor igual a 0 na matriz.
    P(zeros);
    // Garante o acesso exclusivo à matriz.
    P(acesso);
    // Código para copiar x em um dos elementos da matriz, escolhido de modo
    // aleatório, que tenha um valor igual a 0.
    // Libera o acesso exclusivo à matriz.
    V(acesso);
    // Usa a operação V sobre naozeros para registrar que mais um valor
    // diferente de 0 passou a existir em algum elemento da matriz.
    V(naozeros);
  }
}

void ProcessoB(void)
{
  while(1)
  {
    // Usa a operação P sobre naozeros para garantir que existam pelo menos
    // dois valores diferentes de 0 na matriz.
    P(naozeros);
    P(naozeros);
    // Garante o acesso exclusivo à matriz.
    P(acesso);
    // Código para escolher, de modo aleatório, dois elementos da matriz com
    // valores diferentes de 0, copiar esses valores em x e y, e finalmente copiar
    // o valor 0 para os dois elementos escolhidos.
    // Libera o acesso exclusivo à matriz.
    V(acesso);
    // Usa a operação V sobre zeros para registrar que mais dois elementos da
    // matriz passaram a ter valores iguais a 0.
    V(zeros);
    V(zeros);
    // Código para usar os valores x e y removidos da matriz.
  }
}
```

6. (1,5) Suponha que três processos, A, B e C, tenham sido executados na ordem ABCCBACBACAA quando um algoritmo de escalo-

namento proprietário foi usado pelo sistema operacional. Segundo esse algoritmo, cada ocorrência de um processo na ordem de execução corresponde a exatamente 3ms de execução, com exceção da última ocorrência de cada processo, que pode corresponder a um tempo menor. Suponha ainda que, nas suas últimas ocorrências, A, B e C tenham usado, respectivamente, 1ms, 3ms e 2ms do seu tempo de 3ms. Os tempos de término dos processos serão os mesmos se o sistema operacional passar a usar o escalonador *round robin* com um **quantum** de 4ms, sendo B, A e C a ordem inicial de execução dos processos? Justifique a sua resposta.

Resp.: Pelo enunciado, vemos que os tempos de execução dos processos A, B e C são de, respectivamente, 13ms, 9ms e 11ms, e que os tempos de término de A, B e C são de, respectivamente, 33ms, 24ms e 29ms. Agora, quando o algoritmo por *round robin* for usado com um **quantum** de 4ms e com a ordem inicial B, A e C, vemos que a nova ordem de execução será a dada na tabela a seguir. Nesta tabela, mostramos como os processos são escolhidos pelo algoritmo, sendo que cada coluna refere-se à execução de um processo, dando o tempo de início de cada **quantum** e o processo correspondente. Pela tabela, vemos que os novos tempos de término de A, B e C serão de, respectivamente, 33ms, 25ms e 32ms. Logo, o tempo de término de A não mudará, o tempo de término de B aumentará em 1ms, e o tempo de término de C aumentará em 3ms.

0	4	8	12	16	20	24	25	29	32	33
B	A	C	B	A	C	B	A	C	A	-