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 2015

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 dois programas, A e B, estejam prontos para serem executados no sistema operacional. O programa A gasta 1/3 do seu tempo total de execução com uma operação de E/S, e faz essa operação após executar pela metade do tempo que precisa executar no processador. O programa B não faz operações de E/S. O processador ficará ocioso se a multiprogramação não for usada? E se a multiprogramação for usada exclusivamente para evitar a ociosidade do processador durante as operações de E/S? Justifique a sua resposta.

   Resp.: Como o programa A gasta 1/3 do seu tempo total de execução ao fazer a sua operação de E/S, então ele executa no processador por 2/3 do seu tempo total de execução. Agora, como a operação de E/S é feita após a metade do tempo de execução de A no processador, então ela é feita após 1/3 do tempo total de execução de A.
   -Quando a multiprogramação não é usada, o tempo de ociosidade do processador é o tempo da operação de E/S executada por A porque B não faz operações de E/S, significando que o tempo de ociosidade é 1/3 do tempo total de execução de A.
   -Se a multiprogramação for usada somente para evitar a ociosidade do processador durante as operações de E/S então, como somente o programa B está pronto para executar, e como B não faz operações de E/S, o processador somente ficará ocioso se o tempo total de execução de B for menor do que o tempo da operação de E/S de A, ou seja, menor do que 1/3 do tempo total de execução de A.

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) O conceito de multiprogramação foi desenvolvido para aumentar a eficiência dos computadores pessoais.

   Resp.: F (Falsa), pois o conceito de multiprogramação foi originalmente desenvolvido para evitar a ociosidade do processador quando o programa em execução faz operações de E/S.

(b) (0,5) A grande diferença entre o modelo cliente-servidor e os outros modelos de estruturação de um sistema operacional é que, nesse modelo, o núcleo do sistema somente gerencia a execução dos processos no processador, deixando todas as outras tarefas de gerenciamento para os processos servidores executando no modo usuário.

Resp.: F (Falsa), pois o núcleo do sistema é responsável pelo gerenciamento das mensagens trocadas entre os processos clientes e servidores executando no modo usuário, e também pelo acesso aos dispositivos físicos do hardware.

(c) (0,5) No modelo de processos, o sistema operacional é composto por um conjunto de processos sequenciais que executam nos processadores disponíveis ao sistema, sendo que todos os detalhes de como os processos são comutados nesses processadores ficam ocultos dentro do escalonador usado pelo sistema.

Resp.: V (Verdadeira).

(d) (0,5) Quando um processo tenta acessar a sua região crítica através de um semáforo e não consegue acessá-la devido a outro processo estar na sua região crítica, o processo fica em estado de espera ocupada, continuamente verificando se já pode acessar a sua região crítica.

Resp.: F (Falsa), porque um semáforo bloqueia o processo quando ele não pode acessar a sua região crítica, colocando-o no estado **Pronto** somente quando o acesso à região crítica está liberado.

(e) (0,5) Os algoritmos *round robin* e do sorteio são exemplos de algoritmos de escalonamento preemptivos, enquanto que o algoritmo do trabalho mais curto primeiro é um exemplo de um algoritmo de escalonamento não-preemptivo.

Resp.: V (Verdadeira).

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

   (a) (0,5) Sistema definido na terceira geração de computadores, no qual o tempo de processamento é dividido entre os usuários, conectados ao sistema a partir de terminais, e no qual cada usuário tem a ilusão de estar usando exclusivamente a máquina.

   Resp.: Sistema de Compartilhamento de Tempo.

   (b) (0,5) Instrução usada para garantir a exclusão mútua que permite, em uma única operação atômica, verificar o conteúdo de uma posição de memória e, depois disso, colocar nela um valor diferente de zero.

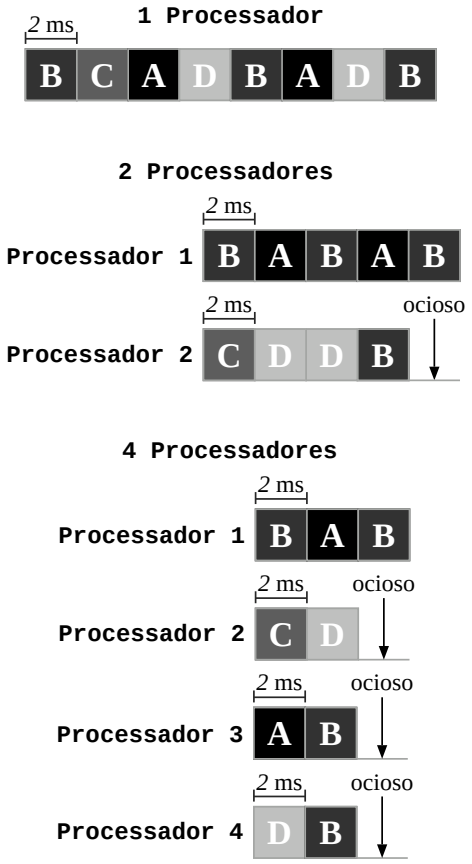
   Resp.: TSL.

   (c) (0,5) Algoritmo de escalonamento no qual cada processo executa no processador por um dado intervalo de tempo, chamado de *quantum*, sendo que um processo somente poderá executar novamente no processador por mais um *quantum*, se necessário, depois de cada um dos outros processos ter também executado por um *quantum*.

   Resp.: *Round robin.*

4. (1,5) Suponha que o escalonador substitua o processo em execução em um processador a cada *quantum* de 2ms, e que um processo somente possa executar novamente depois que todos os processos que ainda não tiverem terminado tenham recebido o mesmo número de *quanta*. Um aluno de sistemas operacionais mostrou, nas figuras a seguir, como ficam as execuções de quatro processos, A, B, C e D, com tempos de execução de, respectivamente, 4ms, 8ms, 2ms e 4ms, para os casos de 1, 2 e 4 processadores, supondo que nenhum processo fez operações de E/S. As figuras do aluno estão corretas? Se você achar que sim, basta

dizer isso mas, se você achar que não, diga quais foram os erros do aluno nas figuras.



Resp.: A primeira figura, que mostra a execução em 1 processador, e a última figura, que mostra a execução em 4 processadores, contêm erros. Na primeira figura, faltou executar o processo B no final da ordem de execução, pois B precisa executar no processador por 4 quanta devido a seu tempo de execução ser de 8ms. Na última figura, B não pode executar simultaneamente nos processadores 3 e 4. O correto é B executar simultaneamente com A e D e depois sozinho por mais dois quanta (a execução pode ser no mesmo processador ou em dois processadores diferentes).

5. (1,5) Suponha que três processos, A, B e C, estejam compartilhando uma fila, inicialmente vazia e capaz de armazenar um número ilimitado de elementos. O processo A continuamente coloca dois elementos no início da fila. Já o processo B move o elemento do final da fila para o início da fila. Finalmente, o processo C remove três elementos do final da fila. Como um semáforo binário e um semáforo de contagem podem ser usados para garantir o correto funcionamento dos processos A, B e C? Justifique a sua resposta.

Resp.: Os dois semáforos devem ser usados como descrito a seguir. O semáforo binário, chamado $acesso_F$, é usado para garantir o acesso exclusivo de A, B ou C a F. Finalmente, o semáforo de contagem, chamado $cheia_F$, conta o número de entradas usadas em F, e é usado para bloquear B ou C se F estiver vazia. Como inicialmente F está vazia e não está sendo usada, então os semáforos $cheia_F$ e $acesso_F$ são inicializados, respectivamente, com 0 e 1. A seguir mostramos os códigos para os processos A, B e C, sendo que a função $removefinal()$ remove e retorna o elemento no final de F, e a função $insereinicio(elemento)$ insere o elemento *elemento* no início de F.

```
void ProcessoA(void)
{
  while (1);
  {
    // Código para gerar 2 elementos e salvá-los em e1 e e2.
    // Garante o acesso exclusivo a F.
    P(acesso_F);
    // Insere os dois elementos gerados no início de F.
    insereinicio(e1);
    insereinicio(e2);
    // Libera o acesso exclusivo a F.
    V(acesso_F);
    // Usa a operação V sobre cheia_F para registrar que
    // 2 elementos foram inseridos em F.
    V(cheia_F);
    V(cheia_F);
  }
}
```

```
void ProcessoB(void)
{
  while (1);
  {
    // Garante que existe pelo menos um elemento em F.
    P(cheia_F);
    // Garante o acesso exclusivo a F.
    P(acesso_F);
    // Remove o elemento do final de F e o copia para e.
    e = removefinal();
    // Insere o elemento e no início de F.
    insereinicio(e);
    // Libera o acesso exclusivo a F.
    V(acesso_F);
    // Precisamos agora usar a operação V sobre cheia_P porque
    // removemos um elemento que depois foi reinserido em F.
    V(cheia_F);
  }
}
```

```
void ProcessoC(void)
{
  while (1);
  {
    // Garante que existem pelo menos 3 elementos em F.
    P(cheia_F);
    P(cheia_F);
    P(cheia_F);
    // Garante o acesso exclusivo a F.
    P(acesso_F);
    // Remove os três elementos do final de F e os copia
    // para e_1, e_2 e e_3.
    e_1 = removefinal();
    e_2 = removefinal();
    e_3 = removefinal();
    // Libera o acesso exclusivo a F.
    V(acesso_F);
    // Código para usar os 3 elementos salvos em e_1, e_2 e e_3.
  }
}
```

6. (1,5) Suponha que a execução de quatro processos, A, B, C e D, tenha gerado a ordem de execução AABBCDDCBAADCBDA. Suponha ainda que cada aparecimento de um processo nessa ordem corresponda a 3 unidades de tempo. Se o sistema operacional agora passar a usar o algoritmo *round robin* com um *quantum* de 4 unidades de tempo, e se a ordem inicial de execução for B, D, A, C, a ordem de término dos processos será a mesma? E se o algoritmo do trabalho mais curto primeiro for usado ao invés do *round robin*? Justifique a sua resposta.

   Resp.: Pela ordem de execução, vemos que os tempos dos processos A, B, C e D são de, respectivamente, 15, 12, 9 e 12 unidades de tempo. Pelo enunciado, para o algoritmo por *round robin*, vemos que a ordem de execução dos processos é como dado 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. Devido a que os

tempos dos processos A e C não são múltiplos do tamanho do quantum de 4 unidades de tempo, A somente usará 3 unidades de tempo do seu último quantum e C somente usará 1 unidade de tempo do seu último quantum. Pela tabela, vemos que a ordem de execução é BDACB-DACBDACA, o que implica que a ordem de término, B, D, C e A , não é igual à original.

0	4	8	12	16	20	24	28	32	36	40	44	45
B	D	A	C	B	D	A	C	B	D	A	C	A

-Como vimos na aula 6, no algoritmo do trabalho mais curto primeiro, os processos são executados, em ordem, de acordo com os seus tempos de execução. Além disso, quando um processo começa a executar, ele executa exclusivamente no processador até terminar, o que significa que a ordem de término é igual à ordem de execução dos processos. Agora, como os tempos de execução dos processos B e D são iguais, teremos duas possíveis ordens de término, C, B, D e A ou C, D, B e A, onde somente a segunda ordem não é igual à original.