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 - Primeiro Semestre de 2018
(Segunda chamada)

Nome -
Assinatura -

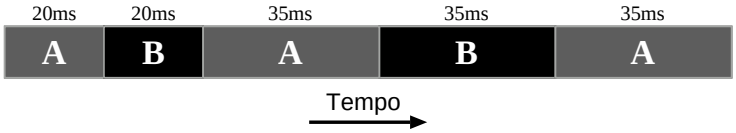
Observações:

1. Prova sem consulta e sem uso de celular ou 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 90ms e que, durante a sua execução, precise fazer duas operações de E/S, a primeira com 20ms de duração e a segunda com 35ms. Suponha ainda que a primeira operação seja feita após o processo ter executado por 20ms no processador, e que a segunda seja feita após o processo ter executado por mais 35ms no processador. Se a multiprogramação somente é usada para evitar a ociosidade do processador quando operações de E/S são feitas, qual é o tempo mínimo que um programa B precisa executar no processador para garantir que o processador não fique ocioso? Suponha que A começa a executar antes de B e que B esteja pronto para executar antes de A fazer a sua primeira operação de E/S. Justifique a sua resposta.

   Resp.: Para que a ociosidade do processador seja evitada, B precisa executar logo após A iniciar sua primeira operação de E/S, e A precisa voltar a executar logo após B terminar a sua única operação de E/S. Além disso, quando A ou B executarem, eles precisarão fazê-lo por um tempo pelo menos igual ao da operação de E/S (do outro processo) que acabou de ser iniciada. Assim, basta que B execute por no mínimo 20ms antes de sua operação de E/S (fazendo-a coincidir com os primeiros 35ms de execução de A após sua primeira operação de E/S) e por no mínimo 35ms após sua operação de E/S (a fim de sobrepor-se ao tempo gasto pela segunda operação de E/S de A). A figura abaixo ilustra o caso dos tempos mínimos:



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) Nome do arquivo usado pelos dispositivos baseados em blocos acessados aleatoriamente, o que permite acessar esses dispositivos como se fossem arquivos regulares.

   Resp.: Arquivo especial de blocos.

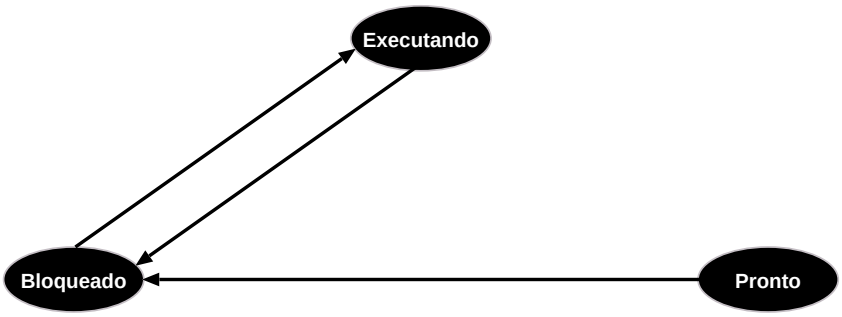
   (b) (0,5) Uma atividade em execução, possuindo um programa com o código a ser executado, uma entrada, uma saída, um contexto e um estado atual.

   Resp.: Processo.

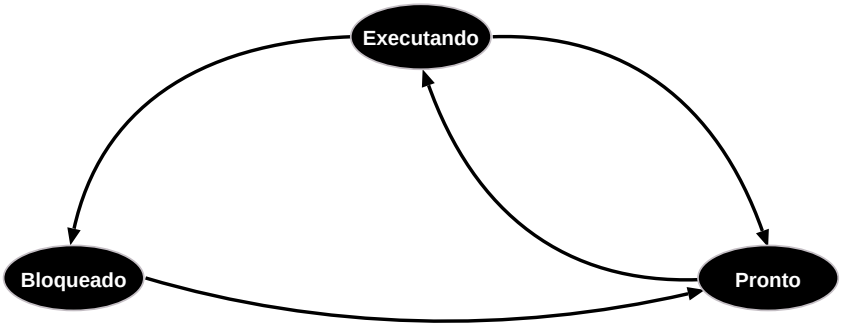
   (c) (0,5) Tipo de algoritmo de escalonamento no qual é usado um temporizador que gera periodicamente interrupções com o objetivo de, em cada interrupção, substituir o processo em execução por um outro no estado **Pronto** caso esse processo já tenha executado por tempo suficiente no processador.

   Resp.: Preemptivo.

4. (1,5) Um aluno de sistemas operacionais disse que o diagrama a seguir representa a correta relação entre os possíveis estados de um processo. O diagrama do aluno está correto? Se você achar que sim, basta dizer isso mas, se você achar que não, diga quais foram os erros do aluno no diagrama.



Resp.: O diagrama do aluno contém quatro erros. O primeiro erro é que, quando desbloqueado, um processo passa do estado **Bloqueado** para o **Pronto**, não para o **Executando**. O segundo erro é que a transição do estado **Pronto** para o **Bloqueado** não tem sentido, porque um processo deve estar executando para poder ser bloqueado. Finalmente, os dois últimos erros são que duas transições estão faltando: uma do estado **Pronto** para o **Executando**, indicando que um processo no estado **Pronto** foi escolhido pelo escalonador para ser executado, e outra do estado **Executando** para o **Pronto**, indicando que o processo em execução foi suspenso devido ao escalonador ter escolhido um novo processo para ser executado. A figura a seguir mostra o diagrama correto.



5. (1,5) Suponha que dois processos, A e B, compartilhem uma árvore, inicialmente sem nós, que pode armazenar até n nós. O processo A continuamente cria um novo nó e o associa aleatoriamente a um dos nós da árvore, se a árvore não estiver cheia, enquanto que o processo

B sempre remove um dos nós folhas da árvore, se a árvore não estiver vazia. Como um semáforo binário e dois semáforos de contagem podem ser usados para garantir o correto funcionamento dos processos A e B? Justifique a sua resposta.

Resp.: A seguir mostramos como três semáforos, um binário e dois de contagem, podem ser usados para implementar os códigos dos processos. O semáforo binário, chamado *acesso*, é usado para garantir o acesso exclusivo à arvore. O primeiro semáforo de contagem, chamado *vazias*, conta o número de nós que ainda podem ser criados na árvore e é usado para bloquear o processo A quando a árvore está cheia. Finalmente, o segundo semáforo de contagem, chamado *cheias*, conta o número nós da árvore e é usado para bloquear o processo B quando a árvore está vazia. Como inicialmente a árvore está vazia e não está sendo usada, então os semáforos *vazias*, *cheias* e *acesso* são inicializados, respectivamente, com n, 0 e 1. A seguir mostramos os códigos para os processos A e B, sendo que a função $removenofolha()$ remove e retorna um dos nós folha da árvore, e a função $insereno(e)$ insere o nó e na árvore:

```
void ProcessoA(void)
{
  while(1)
  {
    // Código para criar um novo nó e depois salvá-lo em e.
    // Usa a operação P sobre vazias para garantir que podemos criar um novo
    // nó na árvore.
    P(vazias);
    // Garante o acesso exclusivo à árvore.
    P(acesso);
    // Insere e na árvore.
    insereno(e);
    // Libera o acesso exclusivo à árvore.
    V(acesso);
    // Usa a operação V sobre cheias para registrar que um nó foi inserido na
    // árvore.
    V(cheias);
  }
}
```

```
void ProcessoB(void)
{
  while(1)
  {
    // Usa a operação P sobre cheias para garantir que existe pelo menos um nó
    // folha na árvore.
    P(cheias);
    // Garante o acesso exclusivo à árvore.
    P(acesso);
    // Remove um nó folha da árvore e o armazena em e.
    e = removenofolha();
    // Libera o acesso exclusivo à árvore.
    V(acesso);
    // Usa a operação V sobre vazias para registrar que um nó folha foi removido
    // da árvore.
    V(vazias);
    // Código para usar o nó folha e.
  }
}
```

6. (1,5) Suponha que três processos, A, B e C, sejam executados no processador na ordem CABCBACBACAA. Suponha também que essa ordem resulte do fato de o processo em execução ser interrompido a cada t unidades de tempo para uma decisão do escalonador. Qual será a sequência de execução se o sistema operacional passar a usar o algoritmo *round robin* com um quantum de $2t$ unidades de tempo, e se a ordem inicial de execução dos processos for CAB? Justifique a sua resposta.

   Resp.: Pelo enunciado, vemos que os tempos totais de execução (no processador) dos processos A, B e C são de, respectivamente, $5t$, $3t$ e $4t$ unidades de tempo. Agora, ao usar 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 aos tempos dos processos A e B não serem múltiplos do tamanho do quantum de $2t$ unidades de tempo, A e B usam somente t unidades de tempo do seu último quantum. Pela tabela, vemos que a nova ordem de execução é CABCABA.

0	$2t$	$4t$	$6t$	$8t$	$10t$	$11t$
C	A	B	C	A	B	A