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 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 executando no computador. O programa A executa no processador por a unidades de tempo e não faz operações de E/S. Já o programa B faz uma operação de E/S com duração de xa, $x > 0$. Qual será o tempo de ociosidade do processador, em função de a e x, se a multiprogramação for usada somente para evitar a ociosidade do processador durante a execução de operações de E/S? Justifique a sua resposta.

   Resp.: Inicialmente, note que se o programa A executar antes do programa B, então não será possível evitar a ociosidade do processador, pois não existe nenhum outro programa para ser executado. Nesse caso, o tempo de ociosidade será igual ao tempo da operação de E/S, ou seja, xa unidades de tempo. Agora, se o programa A executar depois do programa B, então A poderá ser executado logo após B ser suspenso devido a ter iniciado a operação de E/S. Nesse caso, B somente voltará a executar após terem terminado a execução de A e a operação de E/S. Como o tempo de execução de A no processador é de a unidades de tempo, então teremos um tempo de ociosidade de $ax - a = (x - 1)a$ unidades de tempo se $x > 1$, ou nenhuma ociosidade no caso contrário.

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) Somente o modo núcleo ou supervisor permite o acesso direto aos dispositivos físicos do computador.

   Resp.: V (Verdadeira).

   (b) (0,5) As chamadas ao sistema operacional servem apenas para definir a interface entre os processos e os dispositivos físicos do hardware.

   Resp.: F (Falsa), pois as chamadas ao sistema operacional definem a interface entre o sistema operacional e os processos em execução no modo usuário, sem que essa interface se limite ao

acesso a dispositivos físicos.

(c) (0,5) O contexto de um processo é onde são armazenados os dados gerados durante a sua execução.

Resp.: F (Falsa), porque o contexto do processo armazena todas as informações necessárias para reiniciar o processo do ponto em que ele parou quando foi suspenso pelo escalonador ou bloqueado por precisar esperar pela ocorrência de um evento externo.

(d) (0,5) Os processos *CPU-bound* passam a maior parte do tempo da sua execução executando no processador.

Resp.: V (Verdadeira).

(e) (0,5) O semáforo binário, quando inicializado com o valor 1, pode ser usado para garantir a exclusão mútua.

Resp.: V (Verdadeira).

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

(a) (0,5) Definem a interface entre o sistema operacional e os processos em execução no sistema operacional.

Resp.: Chamadas ao sistema operacional.

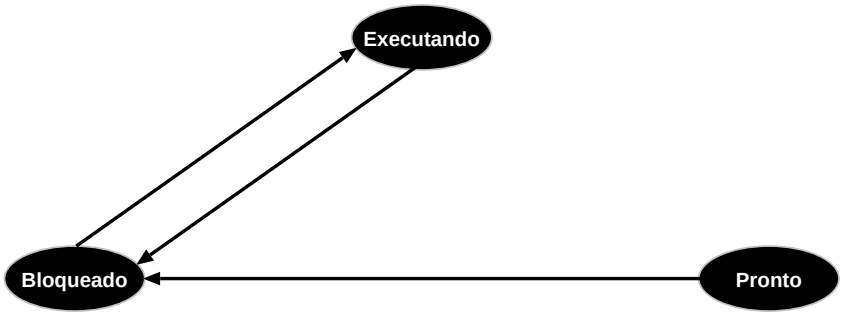
(b) (0,5) Conceito que impede que dois ou mais processos acessem um mesmo recurso simultaneamente, evitando assim a ocorrência de condições de corrida.

Resp.: Exclusão mútua.

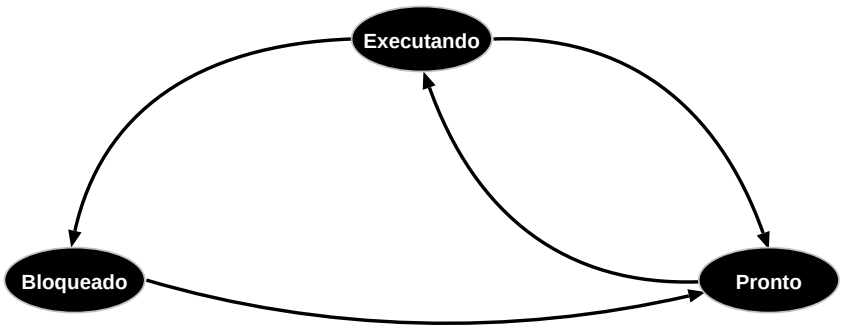
(c) (0,5) Algoritmo de escalonamento que associa um conjunto de bilhetes ao processador, um subconjunto desse conjunto a cada processo em execução no sistema, e no qual o próximo processo a ser executado é o que possuir o bilhete sorteado pelo algoritmo.

Resp.: Escalonamento por sorteio.

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 um conjunto, inicialmente vazio, que pode armazenar um número ilimitado de elementos. O processo A continuamente coloca três elementos no conjunto, e o processo B continuamente remove dois elementos do conjunto, calcula o produto desses elementos, e coloca o resultado novamente no conjunto. Como dois semáforos, um binário e um de contagem, podem ser usados para garantir a correta execução dos processos? Justifique a sua resposta.

   Resp.: O semáforo binário *acesso* é usado para garantir que um dos processos possa acessar exclusivamente o conjunto. Já o semáforo de contagem *cheio* conta o número de elementos no conjunto, e é usado para bloquear o processo B se o conjunto estiver vazio. Como inicialmente o conjunto está vazio, então o semáforo *cheio* é inicializado com 0, e como nenhum processo está inicialmente acessando o conjunto, então o semáforo *acesso* é inicializado com 1. A seguir mostramos os códigos dos processos A e B:

```
void ProcessoA(void)
{
  while (1);
  {
    // Garante o acesso exclusivo ao conjunto.
    P(acesso);
    // Código para gerar os três elementos e inseri-los no conjunto.
    // Libera o acesso exclusivo ao conjunto.
    V(acesso);
    // Usa três operações V sobre cheio para registrar que três
    // elementos foram inseridos no conjunto.
    V(cheio);
    V(cheio);
    V(cheio);
  }
}

void ProcessoB(void)
{
  while (1);
  {
    // Usa duas operações P sobre cheio para garantir que
    // existam pelo menos dois elementos no conjunto.
    P(cheio);
    P(cheio);
    // Garante o acesso exclusivo ao conjunto.
    P(acesso);
    // Código para obter dois elementos do conjunto, multiplicá-los, e
    // colocar o resultado no conjunto.
    // Libera o acesso exclusivo ao conjunto.
    V(acesso);
  }
}
```

6. (1,5) Suponha que a execução de três processos, A, B e C, tenha gerado a ordem de execução AABBCABCAAABBCCABC, usando algum algoritmo de escalonamento, e que cada aparecimento de um processo nessa ordem corresponda a 2 unidades de tempo. Suponha ainda que

o sistema operacional agora passe a usar a versão do algoritmo *round robin* vista na aula 6, na qual as prioridades dos processos definem a ordem de execução inicial dos processos pelo algoritmo, e na qual cada processo tem o seu próprio quantum. Qual será a nova ordem de execução se a prioridade de cada processo for o seu tempo total de execução, se o processo mais prioritário for o que possuir a menor prioridade, e se o quantum atribuído aos processos A, B e C for de, respectivamente, 7, 3 e 5 unidades de tempo? Justifique a sua resposta.

Resp.: Os tempos totais de execução no processador para os processos A, B e C são, respectivamente, 14, 12 e 10 unidades de tempo. Assim, C é o processo mais prioritário, seguido por B e A. Considerando os quanta dados no enunciado, a ordem de execução será conforme dado na tabela abaixo, onde a primeira linha informa o tempo em que será iniciada cada passagem do processo correspondente pelo processador. A nova ordem de execução será então CBACBABB.

0	5	8	15	20	23	30	33
C	B	A	C	B	A	B	B