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
AP1 - Segundo Semestre de 2014

Nome -
Assinatura -

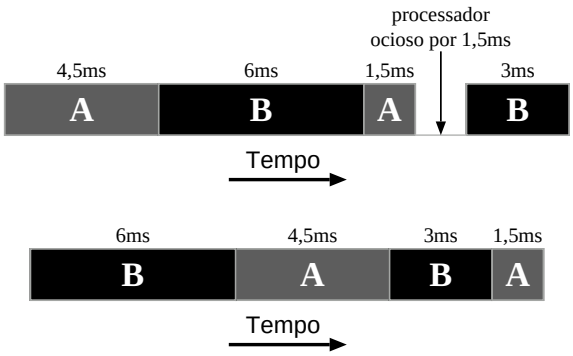
Observações:

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, tenham sido executados em um sistema operacional da segunda geração e que, durante as suas execuções, tenham feito uma operação de E/S de 3ms de duração. Suponha ainda que o tempo de execução de A no processador seja o dobro do tempo da sua operação de E/S, e que o tempo de execução de B no processador seja o triplo do tempo da sua operação de E/S. Se cada processo fizer a sua operação de E/S após executar por metade do seu tempo de execução, qual será o tempo de ociosidade máximo do processador se os programas agora executarem em um sistema operacional no qual a multiprogramação é usada somente para evitar a ociosidade ao fazer operações de E/S? Justifique a sua resposta.

   Resp.: Pelo enunciado, vemos que os tempos de execução dos processos A e B no processador são de, respectivamente, 6ms e 9ms, o que significa que os tempos de execução dos processos foram de, respectivamente, 9ms e 12ms (para cada processo, o tempo de execução é a soma do tempo de execução no processador mais o tempo da operação de E/S). Logo, o processo A executou a operação de E/S após executar por 4,5ms no processador (faltando então 1,5ms) e o processo B executou a operação de E/S após executar por 6ms no processador (faltando então 3ms). Como não foi dada a ordem de execução dos processos, então temos as duas possibilidades dadas nas figuras a seguir e, pelas figuras, vemos que o maior tempo de ociosidade é de 1,5ms.



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) Os sistemas em lote surgiram na segunda geração de computadores.

Resp.: V (Verdadeira).

(b) (0,5) O ponto de montagem é um arquivo especial do sistema de arquivos que permite acessar, de modo aleatório, os blocos de um disco rígido do computador.

Resp.: F (Falsa), pois o ponto de montagem é o nome dado ao diretório do sistema de arquivos usado ao montar um outro sistema de arquivos.

(c) (0,5) Uma das principais características de um sistema cliente/servidor é que seu núcleo, chamado de **micronúcleo**, trata das trocas de mensagens entre os processos e do acesso direto aos dispositivos físicos do hardware.

Resp.: V (Verdadeira).

(d) (0,5) A espera ocupada ocorre quando um processo fica esperando, em um *loop*, pelo término de uma operação de E/S, ao invés de ser bloqueado até a operação terminar.

Resp.: F (Falsa), pois a espera ocupada ocorre quando um processo fica esperando, em um *loop*, até poder acessar a sua seção crítica.

(e) (0,5) O escalonamento em dois níveis é usado quando o computador possui mais de uma unidade de processamento, para garantir que o paralelismo real sempre ocorra ao executar os processos.

Resp.: F (Falsa), pois o escalonamento de dois níveis é usado quando o disco também serve para armazenar os processos, por não existir memória física suficiente para armazená-los. Nesse caso, existem dois escalonadores, o de **baixo nível**, responsável

por escolher o próximo processo a ser executado dentre todos os processos que estão na memória, e o de **alto nível**, responsável por escolher qual dos processos armazenados no disco será copiado para a memória.

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

   (a) (0,5) Classificação dada ao sistema operacional no qual, excluindo o próprio sistema, somente um programa pode estar armazenado na memória em um dado intervalo de tempo.

   Resp.: Monoprogramado.

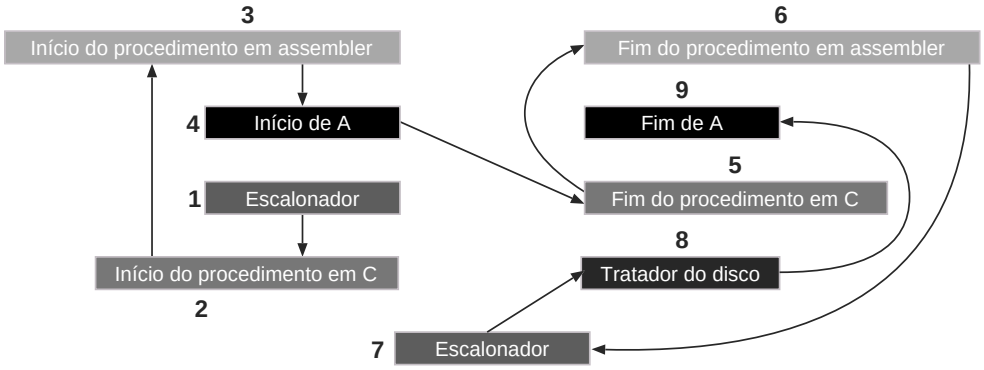
   (b) (0,5) Método usado para garantir a exclusão mútua no qual o processo fica esperando até finalmente poder acessar a sua seção crítica.

   Resp.: Espera ocupada.

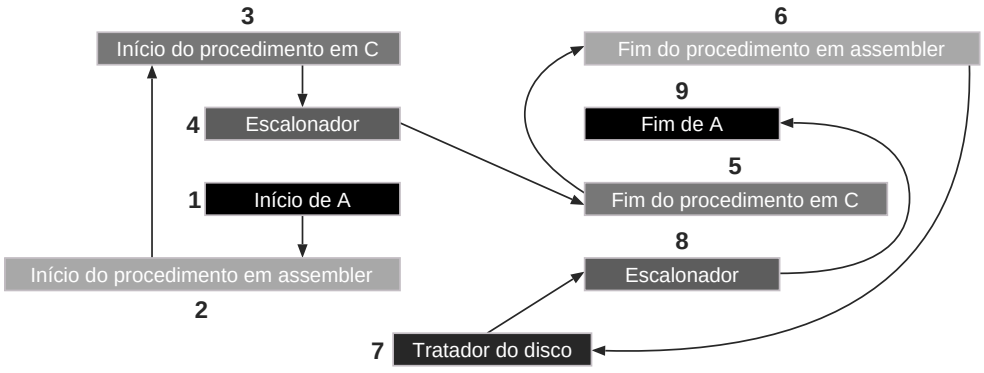
   (c) (0,5) Nome do algoritmo de escalonamento que sempre escolhe, após o processo em execução terminar ou ser bloqueado, o processo com o menor tempo de execução dentre todos os processos prontos.

   Resp.: Trabalho mais curto primeiro.

4. (1,5) Um aluno de sistemas operacionais disse que o diagrama a seguir representa, durante o tratamento de uma interrupção ocorrida durante a execução de um processo A, a correta ordem das ações executadas. 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.



Resp.: O diagrama do aluno está errado porque ele cometeu três erros devidos a permutações nos passos. Nos passos de 1 até 4 existem dois erros, pois o aluno permutou os passos 1 com o 4 e o 2 com o 3. Note que o processo A precisa estar executando quando a interrupção do disco ocorre, porque os procedimentos em assembler e em C e o escalonador são executados exatamente para suspender A quando a interrupção ocorrer para, depois disso, executar o tratador do disco. Finalmente, ele também permutou os passos 7 e 8. Neste caso, o escalonador deve ser chamado depois do tratador do disco porque essa segunda execução do escalonador vai reiniciar A e não o tratador do disco, que foi iniciado pela primeira execução do escalonador. O diagrama correto é o dado a seguir:



5. (1,5) Suponha que três processos, A, B e C, estejam em execução no sistema operacional, que A e B compartilhem uma fila, inicialmente

vazia, com espaço de armazenamento ilimitado, e que B e C compartilhem uma região de memória R. Suponha ainda que A continuamente coloque elementos no início da fila, que B continuamente remova o elemento final da fila e o copie para R, e que C leia o elemento atualmente em R. Como os semáforos podem ser usados para garantir o correto funcionamento dos processos A, B e C? Justifique a sua resposta.

Resp.: Precisamos usar três semáforos para garantir o correto funcionamento dos processos A, B e C: um de contagem, *cheia*, e dois binários, $acesso\_fila$ e $acesso\_R$. O semáforo binário $acesso\_fila$ garante o acesso exclusivo de um processo à fila, e é inicializado com 1 porque inicialmente nenhum processo está acessando a fila. Já o semáforo binário $acesso\_R$ garante o acesso exclusivo de um processo à região R, e também deve ser inicializado com 1 porque inicialmente nenhum processo está acessando R. Finalmente, o semáforo de contagem *cheia* conta o número de entradas usadas na fila, e é usado para bloquear o processo B se a fila estiver vazia. Como inicialmente a fila está vazia, então o semáforo *cheia* é inicializado com 0. A seguir mostramos os códigos para os processos A, B e C:

```
void ProcessoA(void)
{
  while (1);
  {
    // Garante o acesso exclusivo à fila.
    P(acesso_fila);
    // Código para gerar o elemento e inseri-lo no início da fila.
    // Libera o acesso exclusivo à fila.
    V(acesso_fila);
    // Usa a operação V sobre cheia para registrar que
    // um elemento foi inserido na fila.
    V(cheia);
  }
}
```

```
void ProcessoB(void)
{
  while (1);
  {
    // Garante que exista um elemento na fila.
    P(cheia);
    // Garante o acesso exclusivo à fila.
    P(acesso_fila);
    // Código para remover o elemento do início da fila e colocá-lo
    // em e.
    // Libera o acesso exclusivo à fila.
    V(acesso_fila);
    // Garante o acesso exclusivo a R.
    P(acesso_R);
    // Código para copiar e para R.
    // Libera o acesso exclusivo a R.
    V(acesso_R);
}

void ProcessoC(void)
{
  while (1);
  {
    // Garante o acesso exclusivo a R.
    P(acesso_R);
    // Lê e usa o valor em R.
    // Libera o acesso exclusivo a R.
    V(acesso_R);
}
```

6. (1,5) Suponha que dois processos, A e B, estejam prontos para executar no computador, e que o sistema operacional use o algoritmo por *round robin* com um quantum de t unidades de tempo. Se A terminar a sua execução após $2at$ unidades de tempo, e B for o primeiro a executar, em quantas vezes o tempo de B deverá ser maior do que o tempo de A para o tempo de término de B ser abt, $b > 2$, unidades de tempo? Justifique a sua resposta.

Resp.: Como vimos na aula 6, o algoritmo por *round robin* executa os processos no estado **Pronto** de modo alternado, sendo que cada processo executa por um tempo igual ao quantum. Além disso, um processo somente volta a executar, se necessário, por mais um quantum, quando todos os outros processos no estado **Pronto** também já tenham executado por um quantum. Agora, como os tempos de término são múltiplos da duração t do quantum, podemos dizer que A termina após $2a$ quanta e B após ab quanta. Na figura a seguir mostramos a ordem de execução dos processos A e B no processador. Note que A sempre termina antes de B porque o tempo de término de B é ab quanta e porque $b > 2$. Pela figura, vemos que o tempo de execução de A no processador é de a quanta e que o tempo de execução de B no processador é de $a + (ab - 2a) = ab - a = a(b - 1)$ quanta. Como ambos os tempos de execução são múltiplos da duração t do quanta, então o número de vezes que o tempo de execução de B deve ser maior do que o de A é $\frac{a(b-1)}{a} = b - 1$ vezes.