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 2016

Nome -
Assinatura -

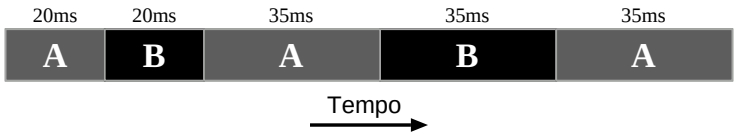
Observações:

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 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, que faz somente uma operação de E/S de 10ms de duração, precisa executar no processador antes e depois dessa operação para garantir que o processador não fique ocioso? Suponha que A comece 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) Na primeira geração de computadores, não existiam linguagens de programação ou sistemas operacionais, o que obrigava os

usuários a programarem diretamente os dispositivos do hardware através de painéis de conectores.

Resp.: V (Verdadeira).

(b) (0,5) Um *pipe* é um pseudoarquivo que permite a troca de informações entre dois processos sem que estes saibam que o estão utilizando. A troca ocorre através da conexão da saída de um dos processos à entrada do outro.

Resp.: V (Verdadeira).

(c) (0,5) Os sistemas monolítico e cliente-servidor são exemplos de dois possíveis modos de gerenciar a multiprogramação em um sistema operacional.

Resp.: F (Falsa), pois os sistemas monolítico e cliente-servidor são exemplos de dois possíveis modos de estruturar um sistema operacional.

(d) (0,5) O contexto de um processo armazena suas entradas e saídas a cada passo de sua execução.

Resp.: F (Falsa), pois o contexto de um processo armazena todas as informações necessárias para reiniciar o processo, do ponto em que ele parou, quando ele é suspenso pelo escalonador, garantindo que o resultado do processo não seja comprometido. A entrada, devido a não variar, não é incluída no contexto, assim como a saída, porque ela é gerada conforme o processo executa no processador.

(e) (0,5) O algoritmo *round robin* cosidera todos os processos como igualmente importantes, pois cada processo, depois de executar por um quantum, somente pode executar novamente, por mais um quantum, se necessário, após todos os outros processos terem

executado também por um quantum.

Resp.: V (Verdadeira).

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

   (a) (0,5) Nome dado aos processos que não precisam interagir com um usuário para poderem executar suas tarefas.

   Resp.: Em segundo plano ou *background*.

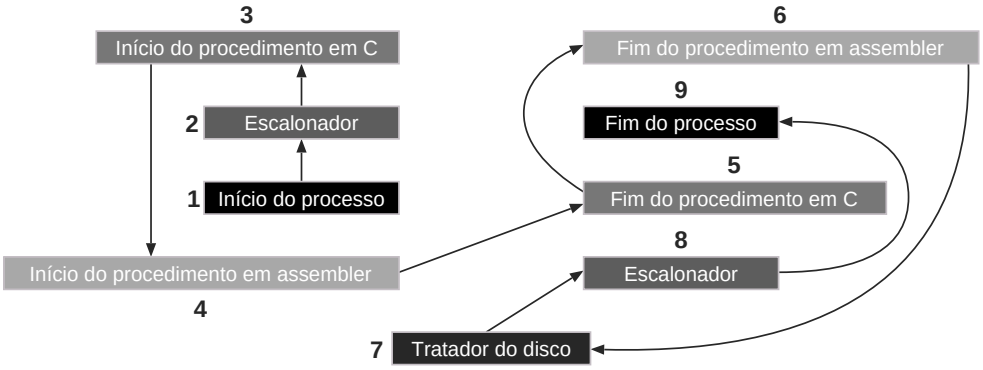
   (b) (0,5) Nome dado ao código através do qual um processo acessa um recurso compartilhado com outros processos.

   Resp.: Região crítica ou seção crítica.

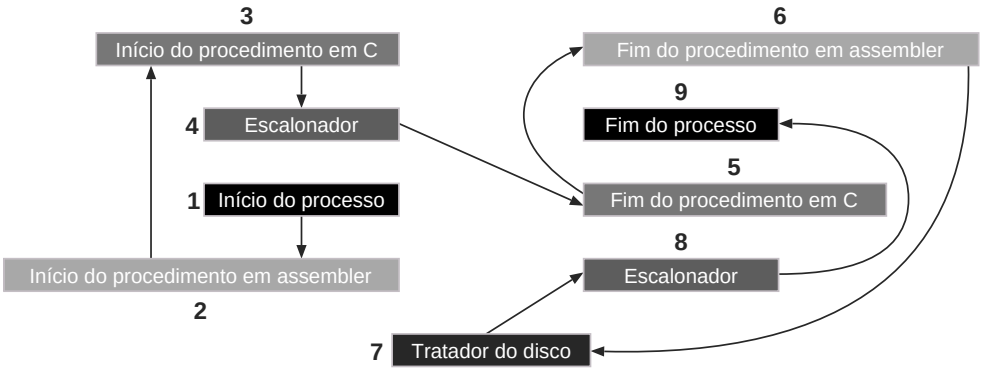
   (c) (0,5) Nome dado aos algoritmos de escalonamento que não usam o temporizador do hardware, caso ele exista, levando a que um processo somente seja suspenso quando termina ou quando é bloqueado.

   Resp.: Algoritmos não-preemptivos.

4. (1,5) Um aluno de sistemas operacionais disse que o diagrama a seguir representa, para o tratamento de uma interrupção de acesso ao disco ocorrida durante a execução de um processo, a correta ordem das ações executadas. O diagrama do aluno está correto? Se você acha que sim, basta dizer isso mas, se você acha que não, diga quais foram os erros do aluno.



Resp.: A figura do aluno está errada, pois ele trocou a ordem de algumas ações. Depois de o processo ser suspenso, o correto é ação "Início do procedimento em assembler" ser executada antes da ação "Início do procedimento em C", e a ação "Escalonador" somente ser executada depois da ação "Início do procedimento em C". Além disso, a transição para a ação "Fim do procedimento em C" vem da ação "Escalonador", e não da ação "Início do procedimento em assembler". A figura a seguir mostra a ordem correta da execução das ações:



5. (1,5) Suponha que dois processos, A e B, compartilhem uma fila, inicialmente vazia mas que pode armazenar até n caminhos de arquivos a serem impressos. O processo A continuamente coloca um novo caminho no final da fila e o processo B continuamente remove dois caminhos do início da fila para serem impressos, em paralelo, nas duas impressoras

disponíveis. Como os semáforos binários podem ser usados para garantir o correto funcionamento de A e de B? Ao responder, utilize um contador a para indicar o número atual de caminhos na fila. Justifique a sua resposta.

Resp.: Para garantir o correto funcionamento dos processos A e B, vamos precisar de três semáforos binários, *acesso*, *vazia* e *cheia*. O semáforo *acesso* é usado para garantir o acesso exclusivo à fila e ao contador a, e é inicializado com o valor 1, pois A e B ainda não estão executando. O semáforo *cheia* é usado para bloquear A se a fila estiver cheia, e o semáforo *vazia* é usado para bloquear B se a fila tiver menos do que dois caminhos. Como os semáforos *cheia* e *vazia* são usados para bloquear, respectivamente, A e B, e como somente executaremos uma operação **P** sobre eles quando for necessário (como veremos nos algoritmos a seguir), então o valor inicial de ambos os semáforos é 0. Nos algoritmos a seguir, a função $InserirCaminho(c)$ insere o caminho c no final da fila, e a função $RemoverCaminho()$ remove o caminho do início da fila. O contador a deve ser inicializado com 0 porque a fila está inicialmente vazia. A seguir mostramos os pseudocódigos para os processos A e B:

```
void ProcessoA()
{
  while (1);
  {
    // Garante o acesso exclusivo à fila e ao contador a.
    P(acesso);
    // Verifica se a fila tem espaço para armazenar um novo
    // caminho.
    if (a == n)
    {
      // Bloqueia o processo até uma entrada estar disponível na fila.
      V(acesso);
      P(cheia);
      P(acesso);
    }
    // Código para obter o novo caminho c.
    // Insere c no final da fila.
    InserirCaminho(c);
    // Incrementa o contador a da fila em uma unidade para
    // registrar o novo caminho inserido.
    a = a + 1;
    // Se dois caminhos existem agora na fila, usamos a operação V
    // sobre vazia para desbloquear B.
    if (a == 2)
    {
      V(vazia);
    }
    // Libera o acesso exclusivo à fila.
    V(acesso);
  }
}
```

```
void ProcessoB()
{
  while (1);
  {
    // Garante o acesso exclusivo à fila e ao contador a.
    P(acesso);
    // Verifica se a fila tem pelo menos dois caminhos.
    if (a < 2)
    {
      // Bloqueia o processo até dois caminhos estarem armazenados
      // na fila.
      V(acesso);
      P(vazia);
      P(acesso);
    }
    // Remove um caminho do início da fila e o copia em c1.
    c1 = RemoverCaminho();
    // Remove um caminho do início da fila e o copia em c2.
    c2 = RemoverCaminho();
    // Decrementa o contador a da fila em duas unidades para
    // registrar os dois caminhos que foram removidos.
    a = a - 2;
    // Como removemos dois caminhos, usamos a operação V sobre
    // cheia para desbloquear A, se antes a fila estava cheia.
    if (a == (n - 2))
    {
      V(cheia);
    }
    // Libera o acesso exclusivo à fila.
    V(acesso);
    // Código para imprimir os arquivos cujos caminhos estão em c1
    // e c2.
  }
}
```

6. (1,5) Suponha que o algoritmo *round robin* tenha sido usado pelo sistema operacional, com um quantum de 2 unidades de tempo, e que três processos, A, B e C, tenham sido executados por, respectivamente, 6, 4 e 5 quanta, sendo que A usou integralmente o seu último quantum, assim como B, enquanto que C usou somente metade do seu. Suponha agora que o sistema operacional passe a usar o algoritmo por prioridades, no qual o processo em execução executa até existir um outro processo com prioridade menor, e no qual cada prioridade é incrementada por 2 unidades a cada 3 unidades de tempo. Quais serão os tempos de término de A, B e C, se as suas prioridades iniciais forem de, respectivamente, 7, 1 e 4? Justifique a sua resposta.

   Resp.: Pelo enunciado, vemos que os tempos totais de execução no processador para os processos A, B e C são, respectivamente, 12, 8 e 9 unidades de tempo. Usando estes tempos e as prioridades dadas no enunciado para A, B e C, teremos a execução dada na tabela a seguir. Nessa tabela mostramos, na primeira linha, o tempo antes de o processo da coluna ter sido executado. Na segunda linha mostramos, da esquerda para a direita, a ordem de execução dos processos no processador. Finalmente, na última linha mostramos, para cada coluna, a prioridade do processo antes de ele executar no processador no tempo dado na mesma coluna. Pela tabela, vemos que os tempos de término de A, B e C serão, respectivamente, de 29, 11 e 20 unidades de tempo.

0	3	6	9	11	14	17	20	23	26
B	B	C	B	C	A	C	A	A	A
1	3	4	5	6	7	8	9	11	13