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 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 por a ms no processador e que, durante a sua execução, ele faça duas operações de E/S, uma de 5 ms e outra de 9 ms. Suponha ainda que a multiprogramação seja usada somente para evitar a ociosidade do processador durante as operações de E/S. Qual será o tempo de ociosidade do processador se somente A estiver executando? E se agora dois programas, B e C (ambos sem operação de E/S), que precisem executar no processador por, respectivamente, 8 ms e 7 ms, puderem iniciar suas execuções antes de A fazer a sua primeira operação de E/S, supondo que A inicia sua execução antes de C e C antes de B? Justifique a sua resposta.

   Resp.: -Quando somente A estiver executando, o tempo de ociosidade do processador será a soma de todos os tempos de suas operações de E/S. Como A executa duas operações de E/S, de 5 ms e 9 ms, então o tempo total de ociosidade do processador será de $5 + 9$ ms, ou seja, 14 ms. Note que o tempo de ociosidade não dependerá do tempo a ms de execução de A no processador.
   -Agora, se a multiprogramação for usada para evitar somente a ociosidade do processador devido a operações de E/S, e se o programa A inicia a sua execução antes dos programas B e C, então B e C poderão ser usados para tentar evitar a ociosidade do processador quando as operações de E/S de A forem feitas. Como C inicia a sua execução antes de B, C poderá ser executado depois que A iniciar a sua primeira operação de E/S de 5 ms e, como C executa por 7 ms, a ociosidade do processador durante a primeira operação de E/S será totalmente evitada. Já no caso da segunda operação de E/S, como o tempo de execução de B de 8 ms é menor do que o tempo de 9 ms da segunda operação de E/S de A, e como não existe nenhum outro programa para ser executado, então o processador ainda ficará ocioso por 1 ms. Logo, o tempo de ociosidade do processador será agora de 1 ms, pois o tempo de ociosidade também não dependerá do tempo a ms de execução de A no processador.

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) A diferença entre as multiplexações dos recursos por tempo e por espaço é que, na primeira, cada recurso é usado exclusivamente por um programa por um dado intervalo de tempo e que, na última, cada programa usa uma parte do recurso.

Resp.: V (Verdadeira).

(b) (0,5) A principal diferença entre os nomes de caminho absoluto e relativo de um diretório é que o primeiro inclui, além do caminho do diretório, todos os caminhos dos arquivos nesse diretório, enquanto que o segundo inclui somente o caminho do diretório.

Resp.: F (Falsa), porque o caminho absoluto define o caminho de um arquivo ou diretório a partir do diretório raiz, enquanto que o caminho relativo define define o caminho de um arquivo ou diretório a partir do diretório de trabalho.

(c) (0,5) O diagrama de estados de um processo mostra os possíveis estados bloqueados em que um processo pode estar ao ser suspenso devido a precisar esperar por um evento externo.

Resp.: F (Falsa), porque o diagrama de estados mostra os três possíveis estados em que um processo pode estar durante a sua execução no sistema operacional: **Executando**, quando o processo está em execução em uma das unidades de processamento do hardware; **Pronto**, quando o processo está esperando para ser escolhido pelo escalonador para depois ser executado em uma das unidades de processamento do hardware; e **Bloqueado** quando o processo foi suspenso até que um dado evento externo ocorra.

(d) (0,5) Uma condição de corrida sempre ocorrerá quando dois ou mais processos de um conjunto tentarem acessar, ao mesmo tempo, uma mesma região de memória.

Resp.: V (Verdadeira).

(e) (0,5) O escalonamento de dois níveis pode ser usado pelo sistema operacional quando a memória não possui espaço suficiente para armazenar todos os processos em execução no sistema, sendo que o escalonador de alto nível é definido para minimizar o tempo de comutação dos processos entre a memória e o disco.

Resp.: V (Verdadeira).

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

   (a) (0,5) Modo de estruturar o sistema operacional no qual o núcleo do sistema não apresenta nenhum tipo de estruturação na sua implementação.

   Resp.: Monolítico.

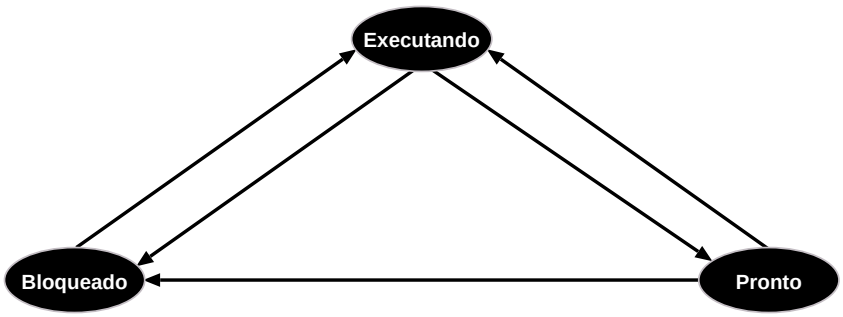
   (b) (0,5) Estrutura usada na implementação do modelo de processos para armazenar todas as informações necessárias à correta execução dos processos.

   Resp.: Tabela de processos.

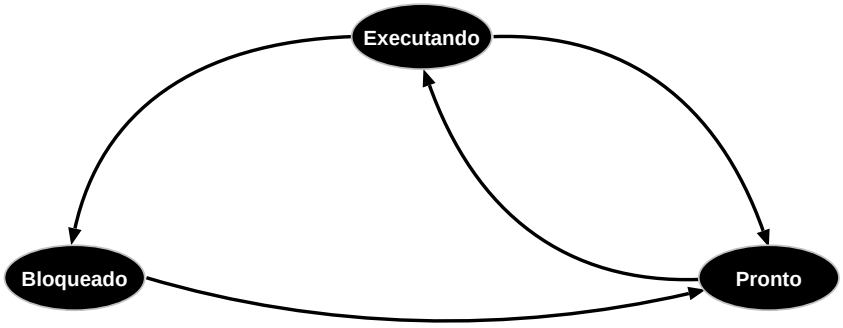
   (c) (0,5) Nome dado ao semáforo que somente pode assumir os valores 0 e 1, que é inicializado com o valor 1, e que é usado para garantir a exclusão mútua no acesso a um dado recurso.

   Resp.: Semáforo binário.

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 acha que sim, basta dizer isso mas, se você acha que não, desenhe o diagrama correto.



Resp.: O diagrama do aluno não está correto pois existem dois erros nele. O primeiro erro é a aresta do estado **Bloqueado** para o **Executando**, pois um processo, depois de ser desbloqueado, sempre deve ser colocado no estado **Pronto**, porque ele pode não ser o processo mais prioritário (se o processo desbloqueado for o mais prioritário, o escalonador poderá ser imediatamente chamado para suspender o processo em execução e colocar esse processo para executar, passando assim o processo imediatamente do estado **Pronto** para o **Executando**). O segundo erro está na aresta do estado **Pronto** para o **Bloqueado**. A aresta correta é a oposta, ou seja, do estado **Bloqueado** para o **Pronto** pois, como observamos antes, o processo desbloqueado precisa ser colocado estado **Pronto** devido a nem sempre ser o mais prioritário. A seguir está o desenho correto do diagrama de estados:



5. (1,5) Considere um conjunto de processos cooperando entre si para executar uma tafera, e suponha que esses processos compartilhem um

conjunto C, inicialmente vazio, que pode armazenar até n identificadores. Um dos processos é especial e acessa C para adicionar um novo identificador a ele, que é sempre maior que o identificador adicionado anteriormente. Cada um dos demais processos acessa C para obter um dos identificadores. Para que o resultado da execução dos processos seja correto, cada processo, com exceção do especial, deve obter um identificador diferente e, para garantir isso, cada um remove o identificador de C que obteve. Como os semáforos devem ser usados para que não ocorram condições de corrida ao acessar C? Justifique a sua resposta.

Resp.: Vamos usar três semáforos, dois de contagem e um binário. A seguir mostramos como os semáforos podem ser usados para garantir o correto funcionamento dos processos. O semáforo binário, chamado *acesso*, é usado para garantir o acesso exclusivo ao conjunto C. O primeiro semáforo de contagem, chamado *vazia*, conta o número de entradas não usadas no conjunto C, e é usado para bloquear o processo especial que coloca um novo identificador em C quando C está cheio. Finalmente, o segundo semáforo de contagem, chamado *cheia*, conta o número de identificadores no conjunto C, e é usado para bloquear um processo que tente obter um identificador quando C está vazio. Como inicialmente o conjunto C está vazio e não está sendo usado, então os semáforos *vazia*, *cheia* e *acesso* são inicializados, respectivamente, com n, 0 e 1. A seguir mostramos o código $ProcessoEspecial()$ do processo especial, e o código $ProcessoNormal()$ que deve ser usado pelos outros processos, supondo que a função $removeIdentificador()$ remova e retorne um identificador do conjunto C, e a função $insereIdentificador(i)$ insira o identificador i no conjunto C:

```
void ProcessoEspecial()
{
  // Função que implementa o processo especial.
  i = 0; // Armazena o número do próximo identificador.
  while (1)
  {
    // Garante que existe pelo menos um espaço vazio no conjunto C.
    P(vazia);
    // Garante o acesso exclusivo ao conjunto C.
    P(acesso);
    // Insere i no conjunto C.
    insereIdentificador(i);
    i = i + 1;
    // Libera o acesso exclusivo ao conjunto C.
    V(acesso);
    // Usa a operação V sobre cheia para registrar que um
    // identificador foi inserido no conjunto C.
    V(cheia);
  }
}

int ProcessoNormal()
{
  // Função que implementa os outros processos.
  while (1)
  {
    // Garante que existe pelo menos um elemento no conjunto C.
    P(cheia);
    // Garante o acesso exclusivo ao conjunto C.
    P(acesso);
    // Remove um identificador do conjunto C e o coloca em i.
    i = removeIdentificador();
    // Libera o acesso exclusivo ao conjunto C.
    V(acesso);
    // Usa a operação V sobre vazia para registrar que um
    // identificador foi removido do conjunto C.
    V(vazia);
    // Código para usar o identificador i.
  }
}
```

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