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 2017

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 por x ms no processador, e que A faça uma operação de E/S, com duração de y ms, durante a sua execução. Responda, justificando a sua resposta:

   (a) (0,5) Qual será a fração do tempo total de execução de A gasta com a operação de E/S?

   Resp.: Pelo enunciado, vemos que o tempo total de execução do programa A é igual a $x + y$. Logo, a fração do tempo total de execução de A gasta com a operação de E/S é $\frac{y}{x+y}$.

   (b) (1,0) Como um programa B pode ser usado para evitar completamente a ociosidade do processador? Considere que o programa B em questão faz no máximo uma operação de E/S e torna-se pronto para executar apenas após o início da execução de A no processador mas antes da E/S de A.

   Resp.: Como o programa B somente torna-se pronto após o programa A começar a sua execução mas antes de ele fazer a sua operação de E/S, e como B faz somente uma operação de E/S, então o tempo de execução de B antes de fazer a sua operação de E/S deve ser no mínimo igual ao tempo y da operação de E/S feita por A. Além disso, para evitar que o processador fique ocioso quando B fizer a sua operação de E/S, o tempo desta operação deve ser no máximo igual ao tempo que A ainda precisa executar no processador após fazer a sua operação de E/S.

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 definido na terceira geração para permitir que vários programas pudessem acessar a memória ao mesmo tempo.

   Resp.: F (Falsa), porque apesar de o conceito de multiprogramação ter sido definido na terceira geração, ele foi criado para

permitir o compartilhamento do processador entre os diversos programas em execução no computador.

(b) (0,5) O caminho absoluto de um arquivo armazenado no sistema de arquivos sempre é definido a partir da raiz desse sistema de arquivos.

Resp.: V (Verdadeira).

(c) (0,5) Um programa é o conjunto de instruções executadas por um processo do sistema operacional.

Resp.: V (Verdadeira).

(d) (0,5) Quando um processo não pode entrar em sua seção crítica e a espera ocupada é usada, a condição para entrada na seção crítica passa a ser verificada repetidamente, gerando assim desperdício de tempo de processamento.

Resp.: V (Verdadeira).

(e) (0,5) O escalonamento de dois níveis foi criado com o objetivo de definir um escalonador para cada tipo de processo, um de alto nível para os processos *CPU-bound* e um de baixo nível para os processos *IO-bound*.

Resp.: F (Falsa), pois o escalonamento de dois níveis foi criado para gerenciar a multiprogramação quando não existe memória suficiente para armazenar todos os processos, exigindo assim o uso do disco. O escalonador de baixo nível é usado para alternar o uso do processador entre os processos que estão residentes na memória. Já o escalonador de alto nível é usado para trocar os processos armazenados na memória com os armazenados no disco, com o objetivo de garantir que todos os processos possam eventualmente executar no processador.

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.

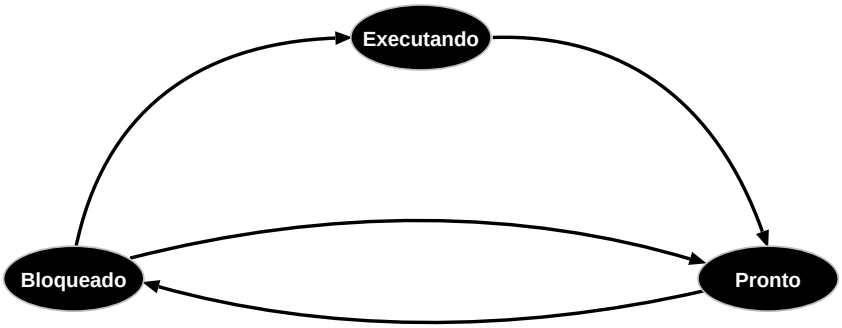
   (b) (0,5) Uma atividade em execução, possuindo um programa com o código a ser executado, uma entrada, uma saída, um contexto e um estado atual.

   Resp.: Processo.

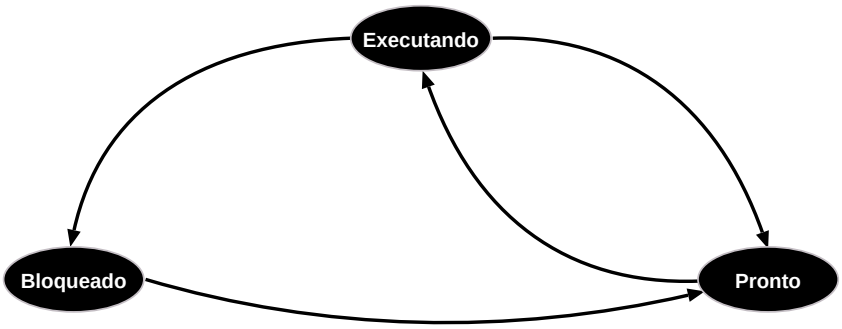
   (c) (0,5) Tipo de algoritmo de escalonamento no qual é usado um temporizador que gera periodicamente interrupções com o objetivo de, em cada interrupção, substituir o processo em execução por um outro no estado **Pronto** caso esse processo já tenha executado por tempo suficiente no processador.

   Resp.: Preemptivo.

4. (1,5) Um aluno de sistemas operacionais disse que o diagrama a seguir representa todos os estados de um processo e a correta relação entre eles. O diagrama do aluno está correto? Se você acha que sim, basta dizer isso mas, se você acha que não, indique todos os erros do diagrama.



Resp.: O diagrama do aluno não está correto porque, apesar de todos os estados estarem corretos, existe uma transição invertida, uma inexistente, e uma ausente. A transição do estado **Pronto** para o estado **Bloqueado** não existe porque somente um processo em execução no processador pode ser bloqueado. A transição do estado **Bloqueado** para o estado **Executando** também não existe porque o processo que foi desbloqueado não é necessariamente o mais prioritário. Na verdade, essa transição deveria ser do estado **Executando** para o estado **Bloqueado**, transição esta que ocorre quando o processo em execução é bloqueado devido a precisar esperar pelo término de um evento externo. Finalmente, o aluno omitiu a transição do estado **Pronto** para o estado **Executando**, que ocorre quando um processo é escolhido para ser executado pelo escalonador. A seguir está o diagrama com todas as correções descritas feitas:



5. (1,5) Suponha que dois processos, A e B, compartilhem um vetor R com n entradas que podem armazenar números, com todas as entradas inicializadas com 0. O processo A continuamente escolhe, de modo aleatório, caso exista, uma entrada com número igual a 0 e escreve nela um número diferente de 0. Já o processo B continuamente escolhe, também de modo aleatório, caso exista, uma entrada com número diferente de 0, lê o número dessa entrada, e depois altera o número dela para 0. Como dois semáforos de contagem e um semáforo binário podem ser usados para garantir a correta sincronização e 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 A e B. O semáforo binário, chamado *acesso*, é usado para garantir o acesso exclusivo ao vetor R. O primeiro semáforo de contagem, chamado *zeros*, conta o número de entradas com números iguais a 0 em R e é usado para bloquear A quando nenhuma entrada de R tem um número igual a 0. Finalmente, o segundo semáforo de contagem, chamado *naozeros*, conta o número de entradas com números diferentes de 0 em R e é usado para bloquear B quando nenhuma entrada de R tem um número diferente de 0. Como todas as entradas de R são inicializadas com o número 0, e como R não é inicialmente usado, então os semáforos *zeros*, *naozeros* 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 *removenumero*() escolhe aleatoriamente uma das entradas com um número diferente de 0, depois altera essa entrada para um número igual a 0 e depois retorna o número que estava originalmente na entrada, e a função *inserenumero*(v) escolhe aleatoriamente uma das entradas com um número igual a 0 e depois altera o número dessa entrada para v:

```
void ProcessoA(void)
{
  while(1)
  {
    // Código para gerar o número v a ser colocado em R.
    // Usa a operação P sobre zeros para garantir que exista pelo menos uma
    // entrada com um número igual a 0 em R.
    P(zeros);
    // Garante o acesso exclusivo a R.
    P(acesso);
    // Copia v para uma das entradas de R, escolhida de modo aleatório, que
    // tenha um número igual a 0.
    inserenumero(v);
    // Libera o acesso exclusivo a R.
    V(acesso);
    // Usa a operação V sobre naozeros para registrar que um número diferente
    // de 0 passou a existir em R.
    V(naozeros);
  }
}
```

```
void ProcessoB(void)
{
  while(1)
  {
    // Usa a operação P sobre naozeros para garantir que exista pelo menos um
    // número diferente de 0 em R.
    P(nonzeros);
    // Garante o acesso exclusivo a R.
    P(acesso);
    // Escolhe, de modo aleatório, uma das entradas de R com um número
    // diferente de 0 e copia esse número em v.
    v = removenumero();
    // Libera o acesso exclusivo a R.
    V(acesso);
    // Usa a operação V sobre zeros para registrar que uma das entradas de R
    // passou a ter um número igual a 0.
    V(zeros);
    // Código para usar o número v.
  }
}
```

6. (1,5) Suponha que três processos, A, B e C, precisem executar no processador por, respectivamente, 12s, 7s e 5s. Quais serão os tempos de

término dos processos se o algoritmo do trabalho mais curto primeiro for usado? Como esses tempos de término irão variar se agora o algoritmo por *round robin* for usado, supondo que a duração do quantum é de 2s e que a ordem inicial de execução dos processos é A, C e B? Justifique a sua resposta.

Resp.: -Como vimos na aula 6, no algoritmo do trabalho mais curto primeiro, os processos são executados, em ordem crescente, de acordo com os seus tempos de execução. Além disso, quando um processo começa a executar, ele executa exclusivamente no processador até terminar. Então a única possível ordem de execução é C, B e A. O processo C então executa de 0s até 5s, o processo B de 5s até o 12s e, finalmente, o processo A de 12s até 24s. Logo, quando o algoritmo do trabalho mais curto primeiro é usado, os tempos de término dos processos A, B e C são, respectivamente, 24s, 12s e 5s.
-Agora, quando o algoritmo por *round robin* é usado, vemos que a ordem de execução dos processos é como dada 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 a os tempos dos processos B e C não serem múltiplos do tamanho do quantum, de 2s, B e C somente usam 1s dos seus últimos quanta. Pela tabela, vemos que os tempos de término dos processos A, B e C, são de, respectivamente, 24s, 20s e 15s. Logo, o tempo de término do processo A não variou, o tempo de término do processo B aumentou em 8s e o tempo de término do processo C aumentou em 10s.

0	2	4	6	8	10	12	14	15	17	19	20	22
A	C	B	A	C	B	A	C	B	A	B	A	A