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 2018

Nome -
Assinatura -

---

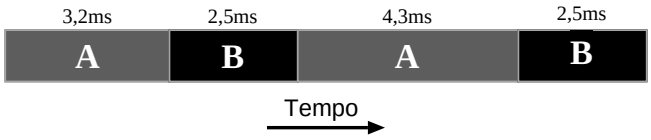
Observações:

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 tenha executado por 7,5 ms e que, durante a sua execução, tenha feito uma operação de E/S, com duração de 2,5 ms, após ter executado no processador por 3,2 ms. Suponha ainda que um programa B tenha executado por 6,8 ms e que, durante a sua execução, tenha feito uma operação de E/S, com duração de $a$ ms, após ter executado por $b$ ms no processador. Se a multiprogramação for usada somente para evitar a ociosidade quando o programa em execução faz uma operação de E/S, que condições deverão satisfazer os tempos $a$ e $b$ para que o processador não fique ocioso, supondo que A tenha iniciado sua execução antes de B? Justifique a sua resposta.

   **Resp.:** Como o programa A inicia antes do programa B então, para evitar a ociosidade do processador durante a operação de E/S feita por A, o tempo de $b$ ms, em que B executa no processador antes de fazer a sua operação de E/S, deve ser maior ou igual ao tempo da operação de E/S de A, ou seja, $b \geq 2,5$ ms. Finalmente, para que a operação de E/S feita por B não gere ociosidade do processador, o tempo de $7,5 - 3,2 = 4,3$ ms, em que A executa no processador após fazer a sua operação de E/S, deve ser maior ou igual ao tempo de $a$ ms da operação de E/S feita por B, ou seja, $a \leq 4,3$ ms. A figura a seguir mostra os casos para os quais $a \leq 4,3$ ms e $b = 2,5$ ms.




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 principal objetivo do modo supervisor, além de ser o modo no qual o núcleo do sistema operacional executa, é limitar o acesso direto ao hardware para evitar que os programas executando no modo usuário interfiram uns com os outros.

**Resp.:** V (Verdadeira).

(b) (0,5) A diferença entre um dispositivo acessado por um arquivo especial de bloco e um dispositivo acessado por um arquivo especial de caracteres é que o primeiro é composto por blocos que podem ser acessados aleatoriamente, enquanto que o segundo é modelado por um fluxo de caracteres.

**Resp.:** V (Verdadeira).

(c) (0,5) O estado de um processo contém as informações que devem ser salvas para que o processo, quando for suspenso pelo escalonador, possa retornar à sua execução do ponto em que parou, como se não tivesse sido suspenso.

**Resp.:** V (Verdadeira).

(d) (0,5) A sincronização dos processos é importante porque permite ao escalonador escolher a melhor ordem de execução para os processos, o que minimiza a ocorrência de condições de corrida.

**Resp.:** F (Falsa), porque o objetivo da sincronização é garantir que um processo somente acesse o recurso compartilhado se as condições necessárias para seu correto funcionamento são satizfeitas pelo estado atual do recurso. Por exemplo, um processo que leia um número de uma fila compartilhada somente pode acessá-la quando ela não está vazia.

(e) (0,5) O escalonador somente escolhe um novo processo para executar no processador quando o processo em execução é bloqueado ou termina a sua execução.

**Resp.:** F (Falsa), porque um escalonador preemptivo, ao contrário do não-preemptivo (que somente usa as duas condições citadas para escolher um novo processo), também pode, em um intervalo de tempo definido por um temporizador do hardware, decidir se o

processo em execução deve continuar a executar ou ser suspenso e substituído por um outro processo no estado **Pronto**.

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

   (a) (0,5) Classificação dada ao sistema operacional que permite, além de si próprio, somente um programa residente na memória por um dado intervalo de tempo.

   **Resp.:** Monoprogramado.

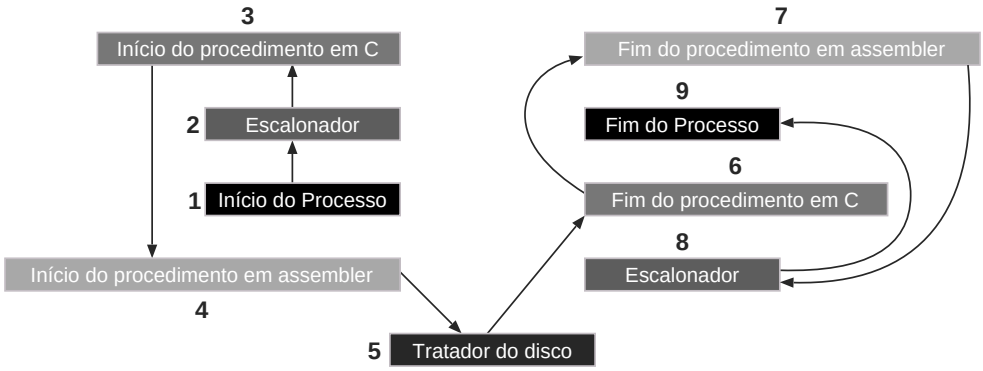
   (b) (0,5) Nome dado à técnica de exclusão mútua na qual um processo fica continuamente verificando se pode entrar na sua seção crítica até finalmente conseguir.

   **Resp.:** Espera ocupada.

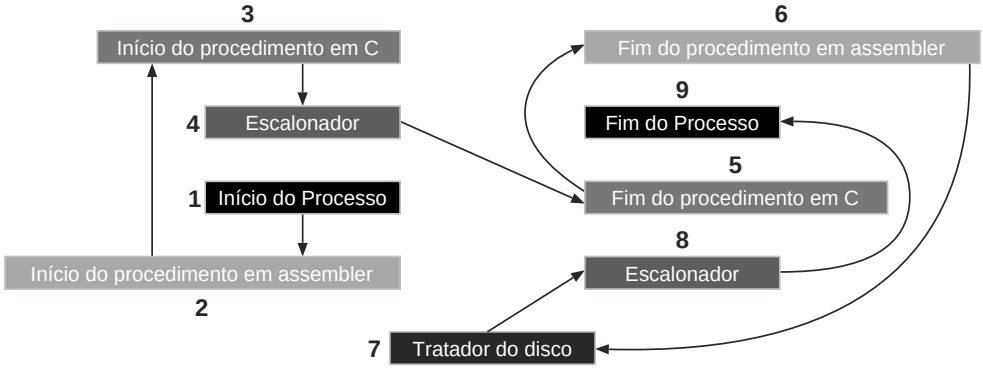
   (c) (0,5) Nome dado ao algoritmo de escalonamento em que um processo, após executar por um dado intervalo de tempo chamado de **quantum**, somente poderá executar novamente por no máximo mais um **quantum** quando todos os outros processos no estado **Pronto** já tiverem executado também por um **quantum**.

   **Resp.:** *Round robin.*

4. (1,5) Suponha que tenha ocorrido uma interrupção do disco enquanto um processo estava em execução. Um aluno de sistemas operacionais disse que a figura a seguir mostra a ordem correta das ações realizadas quando essa interrupção foi tratada pelo sistema operacional. Se você acha que a figura do aluno está correta basta responder que sim mas, se você acha que ela está errada, então indique quais são os erros.




**Resp.:** A figura do aluno está errada, porque somente as transições do estado "Fim do procedimento em C" para o estado "Fim do procedimento em assembler" e do estado "Escalonador" da parte direita da figura para o estado "Fim do Processo" estão corretas. O correto é executar, em ordem, os procedimentos em assembler, C e o escalonador da parte esquerda (a ordem dos estados ligados aos procedimentos e ao escalonador ocorre devido ao escalonador ser chamado pelo procedimento em C que, por sua vez, é chamado dentro do procedimento em assembler), e somente depois de o tratador de disco, colocado para executar no processador pela chamada do escalonador (da parte esquerda), terminar de tratar a interrupção, é que novamente o escalonador (da parte direita) escolhe o processo, que executa até terminar. A seguir está a figura com as ações executadas na ordem correta.




5. (1,5) Suponha que um processo A compartilhe uma palavra de memória $R$ com um processo B. O processo A continuamente espera $R$ conter o

valor 0 e, depois disso, copia um valor aleatório, diferente de 0, para $R$. Já o processo B continuamente espera $R$ conter um valor diferente de 0 e, depois disso, lê esse valor, imprime-o e copia o valor 0 para $R$. Como três semáforos binários podem garantir a correta execução dos processos A e B se inicialmente o conteúdo de $R$ é 2? Justifique a sua resposta.

**Resp.:** O primeiro semáforo, $acesso_A$, é usado para bloquear o processo A caso o valor em $R$ seja diferente de 0. Já o segundo semáforo, $acesso_B$, é usado para bloquear o processo B caso o valor em $R$ seja igual a 0. Finalmente, o terceiro semáforo, $acesso_R$, é usado para garantir o acesso exclusivo a $R$. Como o valor inicial em $R$ é 2, e como $R$ não está sendo inicialmente usado por A ou B, então os valores iniciais dos semáforos $acesso_A$, $acesso_B$ e $acesso_R$ são de, respectivamente, 0, 1 e 1. A seguir mostramos os pseudocódigos para os processos A e B, usando os semáforos descritos anteriormente.

```
void ProcessoA(void)
{
  while(1)
  {
    // Usa a operação P sobre acesso_A para garantir que o valor em R é igual
    // a 0.
    P(acesso_A);
    // Garante o acesso exclusivo a R.
    P(acesso_R);
    // Código para gerar um valor aleatório diferente de 0 e depois armazená-lo
    // em R.
    // Libera o acesso exclusivo a R.
    V(acesso_R);
    // Usa a operação V sobre acesso_B para registrar que um valor diferente de
    // 0 foi copiado para R.
    V(acesso_B);
  }
}
```

```
void ProcessoB(void)
{
  while(1)
  {
    // Usa a operação P sobre acesso_B para garantir que o valor em R é diferente
    // de 0.
    P(acesso_B);
    // Garante o acesso exclusivo a R.
    P(acesso_R);
    // Código para ler e imprimir o valor em R.
    // Código para armazenar o valor 0 em R.
    // Libera o acesso exclusivo a R.
    V(acesso_R);
    // Usa a operação V sobre acesso_A para registrar que o valor 0 foi copiado
    // para R.
    V(acesso_A);
  }
}
```

6. (1,5) Suponha que a ordem de execução dos processos A, B e C, quando o sistema operacional usa o algoritmo *round robin* com um quantum de 2 unidades de tempo, seja ABCABCABCABABB. Se os processos não forem bloqueados durante a sua execução, e se o tempo total de execução de A e de C for par e o de B ímpar, qual será a ordem de execução se o algoritmo *round robin* continuar a ser usado, mas agora com um quantum de 3 unidades de tempo? Suponha que a ordem inicial de execução dos processos seja C, A e B. Justifique a sua resposta.

   **Resp.:** Pela ordem de execução, a duração do **quantum**, e a paridade dos tempos, vemos que os tempos dos processos A, B e C são de, respectivamente, 10, 11 e 6 unidades de tempo. Agora, quando o algoritmo por *round robin* é usado com um **quantum** de 3 unidades de tempo, e com a ordem inicial C, A e B, vemos que a nova ordem de execução é CABCABABAB, como mostra a 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 A e B não serem múltiplos do tamanho do **quantum**, A e B somente usam, respectivamente, 1 e 2 unidades dos seus últimos **quanta**.

| 0 | 3 | 6 | 9 | 12 | 15 | 18 | 21 | 24 | 25 |
|---|---|---|---|---|---|---|---|---|---|
| C | A | B | C | A | B | A | B | A | B |