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 AD1 - Primeiro Semestre de 2019

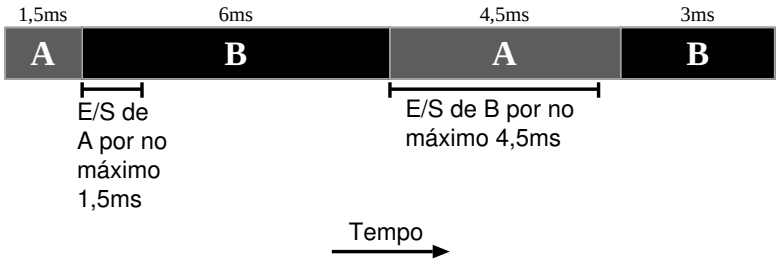
Atenção: Cada aluno é responsável por redigir suas próprias respostas. Provas iguais umas às outras terão suas notas diminuídas. As diminuições nas notas ocorrerão em proporção à similaridade entre as respostas. Exemplo: Três alunos que respondam identicamente a uma mesma questão terão, cada um, 1/3 dos pontos daquela questão.

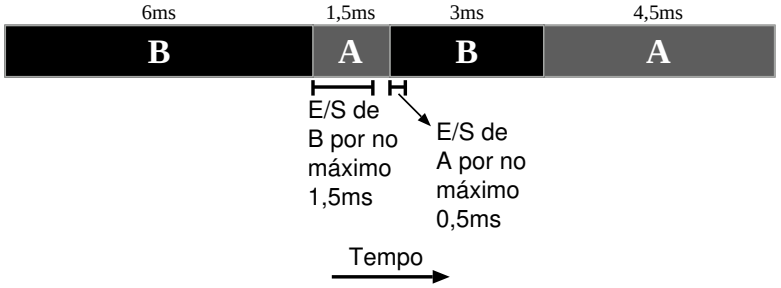
Nome -
Assinatura -

---

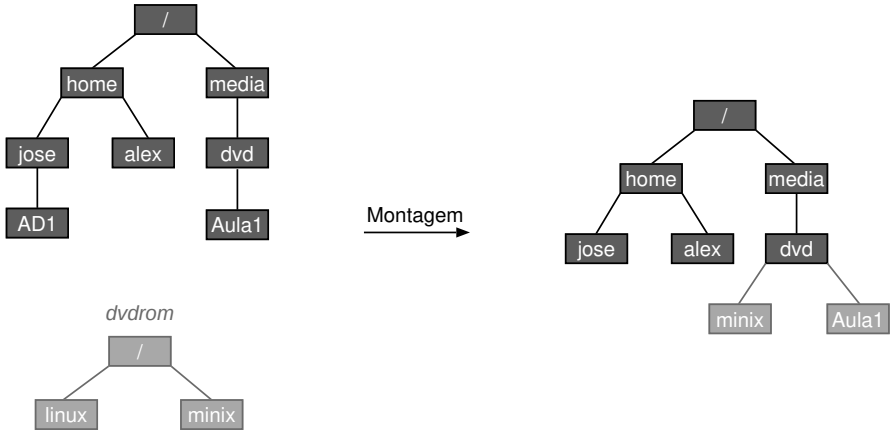
1. (2,0) Suponha que um programa A, que precisa executar no processador por 6,0 ms, faça uma operação de E/S de a ms de duração após executar por 1/4 do seu tempo de execução no processador. Suponha ainda que um programa B, que precisa executar no processador por 9,0 ms, faça uma operação de E/S de $3a$ ms de duração após executar por 2/3 do seu tempo de execução no processador. Qual é o maior valor que a pode assumir para que a ociosidade seja completamente evitada, independetemente de A iniciar a sua execução antes ou depois de B? Justifique a sua resposta.

Resp.: Pelo enunciado, vemos que o programa A executará por 1,5 ms antes de fazer a sua operação de E/S, com duração de a ms, e por 4,5 ms depois de fazer a operação de E/S. Já o programa B executará por 6,0 ms antes de fazer a sua operação de E/S, com duração de $3a$ ms, e por 3,0 ms depois de fazer a operação de E/S. Se A começar a executar antes de B então, para evitar completamente a ociosidade do processador, o tempo de 6,0 ms em que B executa antes de fazer a sua operação de E/S deverá ser pelo menos igual ao tempo da operação de E/S feita por A, que é de a ms. Além disso, o tempo de 4,5 ms em que A executa após fazer a sua operação de E/S deverá ser pelo menos igual ao tempo da operação de E/S feita por B, que é de $3a$ ms. Podemos concluir que $a \leq 1,5$ ms neste primeiro caso. Agora, se B começar a executar antes de A então, para evitar completamente a ociosidade do processador, o tempo de 1,5 ms em que A executa antes de fazer a sua operação de E/S deverá ser pelo menos igual ao tempo da operação de E/S feita por B, que é de $3a$ ms. Adicionalmente, o tempo de 3,0 ms em que B executa após fazer a sua operação de E/S deverá ser pelo menos igual ao tempo da operação de E/S feita por A, que é de a ms. Podemos concluir que $a \leq 0,5$ ms neste segundo caso. Finalmente, temos $a \leq \min\{0,5; 1,5\} = 0,5$. Como desejamos o maior valor possível para a, então $a = 0,5$. Na figura a seguir mostramos os dois casos descritos acima.



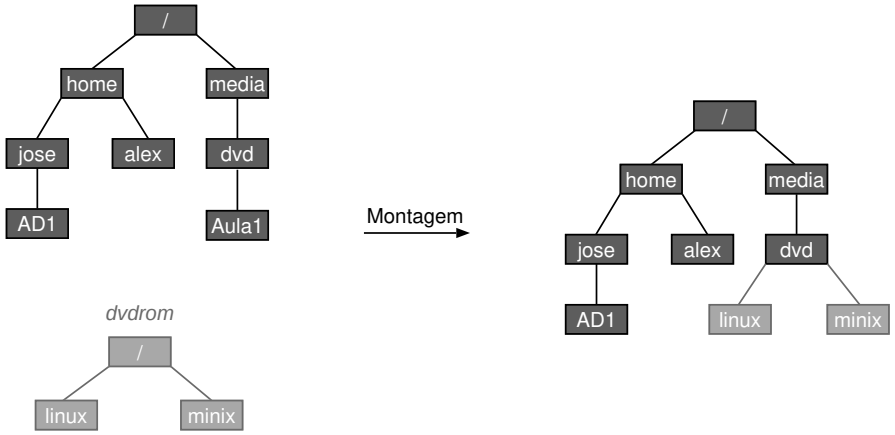


2. (1,0) Um aluno de sistemas operacionais alega que a figura a seguir está correta, pois representa um exemplo de montagem do sistema de arquivos de um *dvdrom* no ponto de montagem **/media/dvd**. A figura do aluno está correta? Se estiver errada, descreva os erros e dê a figura correta, justificando a sua resposta.



Resp.: A figura do aluno está errada porque foram cometidos três erros ao montar o *dvdrom*. O primeiro erro foi a ocultação do arquivo **AD1**, pertencente ao diretório **jose**, não relacionado ao ponto de montagem, porque arquivos ou diretórios de outros diretórios não são ocultados pela montagem. O segundo erro foi a não remoção do arquivo ou diretório **Aula1**, armazenado no ponto de montagem **/media/dvd** porque, quando um sistema de arquivos é montado em um diretório, todos os arquivos e diretórios armazenados nele ficam inacessíveis até o sistema de arquivos ser desmontado. Finalmente, não foi mostrado

todo o sistema de arquivos do *dvdrom* após a montagem, porque faltou o arquivo ou diretório **linux**, o que deveria ocorrer porque todo o sistema de arquivos é montado e acessível a partir do ponto de montagem. Logo, a figura correta é a dada a seguir.



3. (2,0) Suponha que o sistema operacional esteja executando diretamente sobre o hardware de um computador cujas operações de E/S demorem 1,2 ms. Suponha ainda que um processo tenha executado por 10 000 ms e que, durante a sua execução, tenha feito 3 000 operações de E/S. Se o sistema operacional agora executar sobre uma máquina virtual que reduza a velocidade do processador em $x\%$ e a velocidade das operações de E/S em 40%, e se além disso forem executadas somente 2 000 operações de E/S, para que valor de x o processo executará na máquina virtual pelo dobro do tempo da máquina real? Justifique a sua resposta.

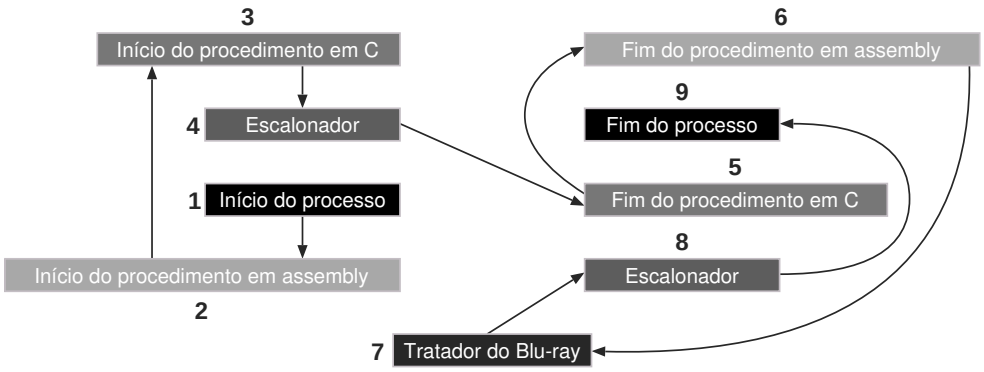
Resp.: Como o tempo total de execução é de 10 000 ms, e como o processo faz 3 000 operações de E/S com duração de $1,2$ ms cada, então 3 600 ms dos 10 000 ms são gastos com operações de E/S quando a execução ocorre sobre o hardware do computador. Logo, o processo executa no processador desse hardware por 6 400 ms. Note que a velocidade do processador ser reduzida em $x\%$ significa que a velocidade do processador virtual é $(100 - x)\%$ da velocidade do processador real, o que por sua vez significa que, durante os 6 400 ms, somente $(100 - x)\%$

das instruções podem ser executadas. O tempo necessário para executar 100% das instruções sobre a máquina virtual é de $\frac{6\,400}{\frac{100-x}{100}} = \frac{640\,000}{100-x}$ ms. Agora, como o processo faz 2 000 operações de E/S na máquina virtual, e como o novo tempo de cada operação de E/S é de $\frac{1,2}{0,6} = 2$ ms (já que, similarmente à redução do tempo do processador, a redução da velocidade de cada operação de E/S em 40% significa que no mesmo tempo podemos, na máquina virtual, executar somente 60% das operações de E/S originais), então 4 000 ms dos 20 000 ms (o dobro do tempo gasto na máquina real) de execução do processo na máquina virtual são gastos com E/S. Logo, o tempo de execução do processo no processador virtual é de 16 000 ms. Usando $\frac{640\,000}{100-x} = 16\,000$, temos $x = 60$, ou seja, o fator de redução da velocidade do processador virtual em relação ao real é de 60%.

4. (1,0) Suponha que tenha ocorrido uma interrupção da unidade de Blu-ray enquanto um processo estava em execução. Mostre a ordem na qual as ações dadas na figura a seguir devem ser executadas a fim de representar o tratamento da interrupção pelo sistema operacional.

Resp.: Como vimos no slide 21 da aula 4, o processo precisa estar executando quando a interrupção do disco ocorre, porque os procedimentos em assembly e em C, assim como o escalonador, são executados exatamente para suspender o processo quando a interrupção ocorre, para depois disso executar o tratador do Blu-ray. Além disso, o escalonador

deve ser chamado depois desse tratador, para poder reiniciar o processo depois de a interrupção ser tratada. O diagrama correto é dado a seguir.



5. (2,0) Suponha que três processos, A, B e C, compartilhem um conjunto que pode armazenar n números e uma fila com tamanho ilimitado que também pode armazenar números, ambos inicialmente vazios. O processo A continuamente gera três números, tenta colocar dois números no conjunto e, somente após ter sucesso em colocar esses dois números, tenta colocar na fila o número restante. Já o processo B continuamente tenta remover dois números do conjunto, calcula a sua soma, e depois tenta colocar a soma na fila. Finalmente, o processo C continuamente tenta remover dois números da fila e, somente após ter sucesso em remover esses números, tenta remover um elemento do conjunto para, após ter conseguido, imprimir o produto dos três números. Como três semáforos de contagem e dois semáforos binários podem ser usados para garantir a correta execução dos processos, sem condições de corrida ou impasses? Suponha que existam as seguintes funções: $RemoveNumero()$ para remover e retornar um número do conjunto; $AdicionaNumero(x)$ para adicionar o número x ao conjunto; $InsereFila(x)$ para inserir o número x na fila; e $RemoveFila()$ para remover e retornar um número da fila. Justifique a sua resposta.

Resp.: A seguir mostramos como os cinco semáforos, dois binários e três de contagem, podem ser usados para implementar os códigos

dos processos. O primeiro semáforo binário, chamado *AcessoConjunto*, é usado para garantir o acesso exclusivo ao conjunto. Já o segundo semáforo binário, chamado *AcessoFila*, é usado para garantir o acesso exclusivo à fila. Em relação aos semáforos de contagem, o primeiro deles, chamado *LivresConjunto*, conta a quantidade de números que ainda podem ser inseridos no conjunto e é usado para bloquear A quando o conjunto não possui espaço para dois números adicionais. Já o segundo semáforo, chamado de *UsadosConjunto*, conta quantidade de números no conjunto e é usado para bloquear B quando o conjunto não possui pelo menos dois números ou C quando o conjunto está vazio. Finalmente, o terceiro semáforo, chamado *UsadosFila*, conta a quantidade de números presentes na fila e é usado para bloquear C quando a fila não possui pelo menos dois números. Como inicialmente o conjunto e a fila estão vazios e não estão sendo usados, então os semáforos *AcessoConjunto*, *AcessoFila*, *LivresConjunto*, *UsadosConjunto* e *UsadosFila* são inicializados, respectivamente, com 1, 1, n, 0 e 0. A seguir mostramos os códigos para os processos A, B e C.

```
void ProcessoA(void)
{
  while(1)
  {
    // Código para gerar três números e salvá-los em x, y e z.
    // Usa a operação P sobre LivresConjunto para garantir que podemos
    // inserir dois números (x e y) no conjunto.
    P(LivresConjunto);
    P(LivresConjunto);
    // Garante o acesso exclusivo ao conjunto.
    P(AcessoConjunto);
    // Insere x e y no conjunto.
    AdicionaNumero(x);
    AdicionaNumero(y);
    // Libera o acesso exclusivo ao conjunto.
    V(AcessoConjunto);
    // Usa a operação V sobre UsadosConjunto para indicar que x e y foram
    // inseridos no conjunto.
    V(UsadosConjunto);
    V(UsadosConjunto);
    // Garante o acesso exclusivo à fila.
    P(AcessoFila);
    // Insere z na fila.
    InsereFila(z);
    // Libera o acesso exclusivo à fila.
    V(AcessoFila);
    // Usa a operação V sobre UsadosFila para indicar que z foi inserido na
    // fila.
    V(UsadosFila);
  }
}
```

```
void ProcessoB(void)
{
  while(1)
  {
    // Usa a operação P sobre UsadosConjunto para garantir que temos pelo
    // menos dois números no conjunto.
    P(UsadosConjunto);
    P(UsadosConjunto);
    // Garante o acesso exclusivo ao conjunto.
    P(AcessoConjunto);
    // Remove dois números do conjunto e os armazena em x e y.
    x = RemoveNumero();
    y = RemoveNumero();
    // Libera o acesso exclusivo ao conjunto.
    V(AcessoConjunto);
    // Usa a operação V sobre LivresConjunto para indicar que dois números
    // foram removidos do conjunto.
    V(LivresConjunto);
    V(LivresConjunto);
    // Garante o acesso exclusivo à fila.
    P(AcessoFila);
    // Insere na fila a soma de x com y.
    InsereFila(x + y);
    // Libera o acesso exclusivo à fila.
    V(AcessoFila);
    // Usa a operação V sobre UsadosFila para indicar que x + y foi inserido na
    // fila.
    V(UsadosFila);
  }
}
```

```
void ProcessoC(void)
{
  while(1)
  {
    // Usa a operação P sobre UsadosFila para garantir que temos pelo
    // menos dois números na fila.
    P(UsadosFila);
    P(UsadosFila);
    // Garante o acesso exclusivo à fila.
    P(AcessoFila);
    // Remove dois números da fila e os armazena em x e y.
    x = RemoveFila();
    y = RemoveFila();
    // Libera o acesso exclusivo à fila.
    V(AcessoFila);
    // Usa a operação P sobre UsadosConjunto para garantir que temos pelo
    // menos um número no conjunto.
    P(UsadosConjunto);
    // Garante o acesso exclusivo ao conjunto.
    P(AcessoConjunto);
    // Remove um número do conjunto e o armazena em z.
    z = RemoveNumero();
    // Libera o acesso exclusivo ao conjunto.
    V(AcessoConjunto);
    // Usa a operação V sobre LivresConjunto para indicar que um número
    // foi removido do conjunto.
    V(LivresConjunto);
    // Código para imprimir o produto de x, y e z.
  }
}
```

6. (2,0) Suponha que o algoritmo por *round robin* seja usado pelo sistema operacional, com um **quantum** de a unidades de tempo, e que três processos, A, B e C, precisem executar por, respectivamente xa, ya e za unidades de tempo, sendo x, y e z inteiros e $z > x > y > 0$. Qual será o tempo médio decorrido entre o início e o término de A, B e C, em função de x, y, z e a, supondo que a ordem inicial de execução dos processos é C, A, B? Justifique a sua resposta.

   Resp.: Pelo enunciado, vemos que os processos A, B e C precisam executar, respectivamente, por exatos x, y e z **quanta**, já que x, y e z são inteiros. Agora, como $z > x > y$, e como a ordem de execução inicial é

C, A, B, então teremos y execuções alternadas de C, A e B, seguidas de $x-y$ execuções alternadas de C e A, e finalmente de $z-x$ execuções consecutivas de C. A execução dos processos é mostrada na figura a seguir. Pelo descrito, e pela figura, o tempo decorrido entre o início e o término de A será de $3y+2(x-y)-1 = 2x+y-1$ **quanta** ou $(2x+y-1)a$ unidades de tempo. Já para B, vemos que o tempo decorrido entre o início e o término será de $3y-2$ **quanta** ou $(3y-2)a$ unidades de tempo. Finalmente, para C, vemos que o tempo decorrido entre o início e o término será de $3y+2(x-y)+z-x = x+y+z$ **quanta** ou $(x+y+z)a$ unidades de tempo, o que está de acordo com C ser o primeiro a executar e o último a terminar. Logo, o tempo médio decorrido entre o início e o término será de $\frac{(2x+y-1+3y-2+x+y+z)a}{3} = \frac{(3x+5y+z-3)a}{3} = \left(x + \frac{5}{3}y + \frac{z}{3} - 1\right)a$ unidades de tempo.