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 - Segundo Semestre de 2015

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.

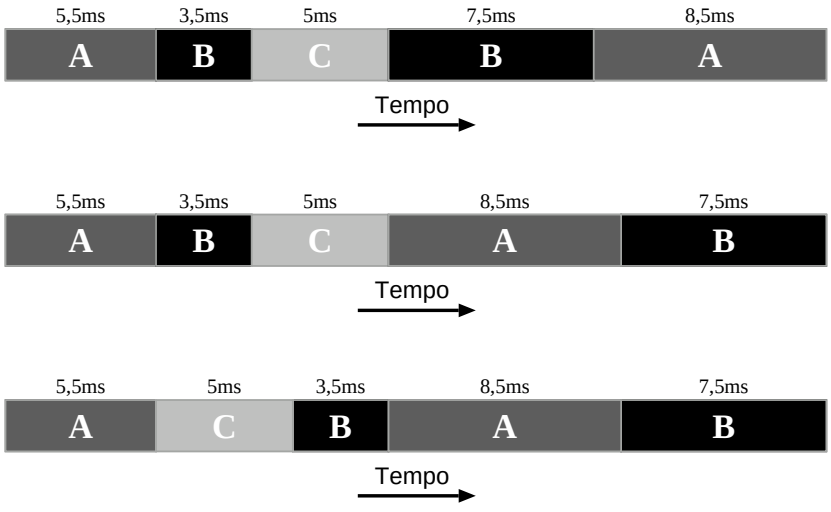
Nome -
Assinatura -

---

1. (1,0) Suponha que dois programas, A e B, estejam em execução no processador. O programa A foi o primeiro a executar: executou por 14ms, tendo precisado fazer uma operação de E/S, com duração de 6ms, após os primeiros 5,5ms de execução. O programa B, que executou por 11ms, também precisou fazer uma operação de E/S, com duração de 4,5ms, após os primeiros 3,5ms de execução. Se o sistema operacional não usar a multiprogramação, qual será o tempo de ociosidade do processador? Agora, se o sistema usar a multiprogramação somente para evitar ociosidade durante operação de E/S, e se um programa C, que não faz operações de E/S e que precisa executar por 5ms, puder executar após

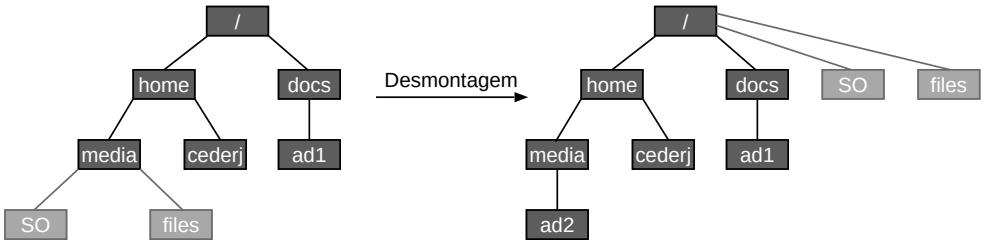
A executar por 3ms, o processador ainda ficará ocioso? Justifique a sua resposta.

Resp.: -Se a multiprogramação não for usada, o processador ficará ocioso quando as operações de E/S forem executadas, ou seja, durante as operações de E/S de A e de B. Assim, o processador ficará ocioso por $6\text{ms} + 4,5\text{ms} = 10,5\text{ms}$.
-Pelo enunciado, vemos que A executa antes de B e de C e que C fica pronto para executar antes de A fazer a sua operação de E/S. Temos então as três possibilidades dadas nas figuras a seguir. Nas duas primeiras figuras, B executa antes de C, ou seja, B executa depois de A fazer a sua operação de E/S e C somente executa depois de B fazer a sua operação de E/S. Como as operações de E/S de A e de B terminam antes de C terminar a sua execução, então A ou B podem executar depois de C terminar. Pelas figuras, vemos que em ambos os casos o processador não fica ocioso. Agora, se C executar antes de B, ou seja, se C executar depois de A fazer a sua operação de E/S e B executar somente depois de C terminar, então teremos a execução dada na última figura e, como podemos ver, o processador também não ficará ocioso.

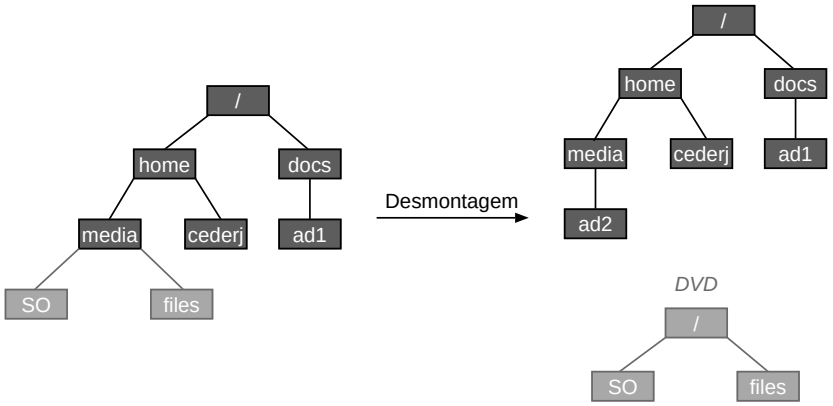


2. (1,0) Um aluno de sistemas operacionais mostrou, na figura a seguir, como o sistema de arquivos ficaria após um DVD montado no diretório

media ser desmontado. A figura do aluno está correta? Justifique a sua resposta.



Resp.: A figura do aluno não está correta. Ao desmontarmos um sistema de arquivos de um diretório, no caso o diretório media, o conteúdo do DVD não mais fica acessível e, portanto, não poderia ser acessível a partir do diretório raiz do sistema de arquivos principal. Note que a existência do arquivo ad2 no diretório media não está incorreta, porque quando um sistema de arquivos é montado em um diretório, todos os arquivos e diretórios originalmente contidos naquele diretório ficam inacessíveis até o sistema de arquivos ser desmontado. A figura correta é dada abaixo.

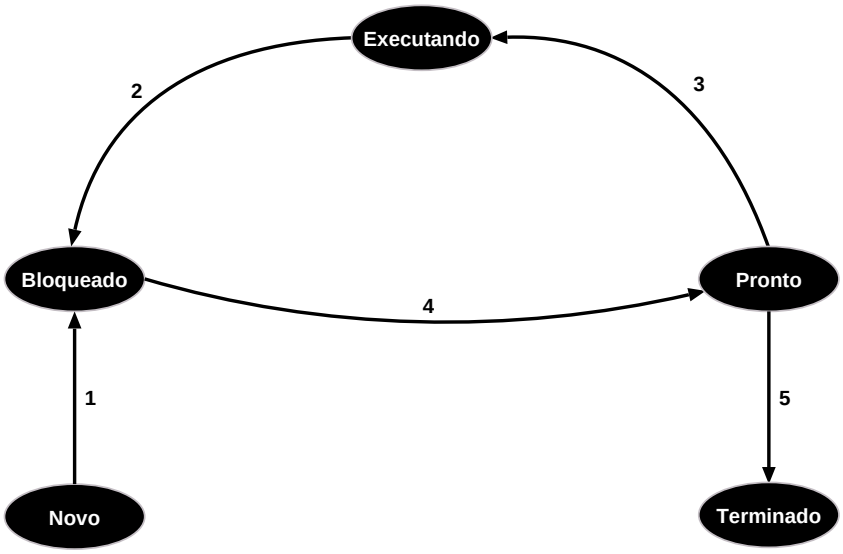


3. (1,5) Suponha que o sistema operacional esteja executando diretamente sobre o hardware de um computador cujas operações de E/S demorem 0,26ms. Suponha ainda que um processo tenha feito, durante a sua

execução, 8500 operações de E/S. Se o sistema operacional agora executar sobre uma máquina virtual que reduza a velocidade do processador em 45% e a velocidade das operações de E/S em 35%, qual será o tempo de execução do processo no hardware se ele executou por 8s na máquina virtual, fazendo as mesmas 8500 operações de E/S? Justifique a sua resposta.

Resp.: Vamos supor que o tempo de execução no hardware do computador tenha sido de x ms. Como cada operação de E/S demora 0,26ms e como o processo faz 8 500 operações, então 2 210ms do tempo de execução de x ms desse processo são gastos em operações de E/S, quando ele executa no sistema operacional sobre o hardware do computador. Logo, o processo executa no processador do hardware por $(x - 2\,210)$ ms. Note que a velocidade do processador ser reduzida em 45% significa que a velocidade do processador virtual é 55% da velocidade do processador real, o que por sua vez significa que, durante os $(x - 2\,210)$ ms, somente 55% das instruções são executadas. Com isso, quando o processo executa no sistema operacional sobre a máquina virtual, o tempo de execução dele no processador virtual é de $\frac{(x-2\,210)}{0,55}$ ms. Agora, como o processo executou as mesmas 8 500 operações de E/S, e como o novo tempo de cada operação de E/S é de $\frac{0,26}{0,65} = 0,4$ ms (porque, similarmente à redução do tempo do processador, a redução da velocidade de cada operação de E/S em 35% significa que no mesmo tempo podemos, na máquina virtual, executar somente 65% das operações de E/S originais), então $8500 \times 0,4 = 3\,400$ms dos 8s ou 8 000ms do tempo de execução do processo na máquina virtual foram gastos com E/S. Logo, o tempo de execução do processo no processador virtual é de $8\,000 - 3\,400 = 4\,600$ms e, com isso, podemos concluir que $\frac{(x-2\,210)}{0,55} = 4\,600$, ou seja, que o tempo de execução x do processo no hardware real será de 4 740ms.

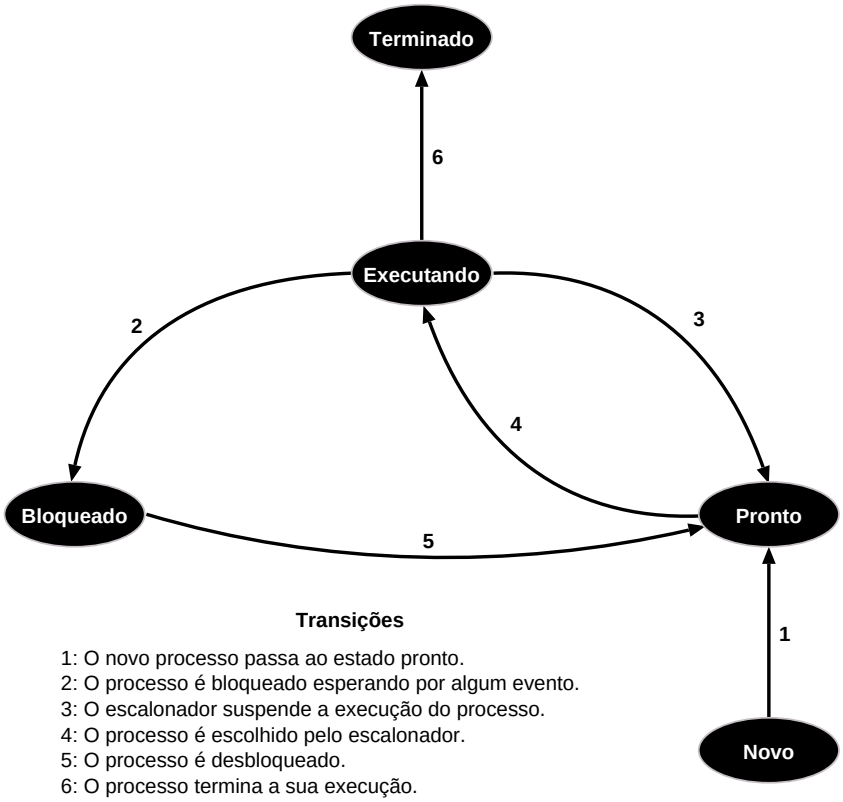
4. (1,5) Na figura dada a seguir mostramos uma versão estendida do diagrama de transição de estados de um processo, com dois novos estados: o estado **Novo**, em que o processo é colocado quando é criado, e o estado **Terminado**, em que o processo é colocado quando termina a sua execução. O diagrama está correto? Justifique a sua resposta.



Transições

1: O novo processo é bloqueado, esperando para ser executado.
2: O processo é bloqueado, esperando por algum evento ou pelo escalonador.
3: O processo é escolhido pelo escalonador.
4: O processo é desbloqueado pois o evento já ocorreu ou porque o escalonador o escolheu.
5: O processo termina a sua execução.

Resp.: O diagrama não está correto. Um processo que acabou de ser criado e que foi colocado no estado **Novo** não pode passar ao estado **Bloqueado** quando está esperando para ser executado, porque o estado **Pronto** é usado para indicar que um processo está esperando para ser executado no processador. Devido a esse erro, a descrição da transição 4 está incorreta na parte que diz que ela ocorre quando o processo foi escolhido pelo escalonador. Também está incorreta a descrição da transição 2 na parte que diz que o processo é bloqueado pelo escalonador porque, quando o escalonador suspende a execução de um processo, ele é colocado no estado **Pronto**. Devido a este último erro, também foi omitida, no diagrama, a transição do estado **Executando** para o **Pronto**, que ocorre exatamente quando o escalonador suspende o processo em execução, em geral devido a ter escolhido um novo processo para ser executado. Finalmente, o processo não pode passar do estado **Pronto** para o **Terminado** porque, para terminar, ele precisa estar executando e, devido a isso, a transição deveria ser do estado **Executando** para o **Terminado**. A seguir está o diagrama correto.



5. (2,0) Considere três processos, A, B e C, uma fila F, inicialmente vazia, e uma pilha P, também inicialmente vazia. Suponha que F possa armazenar um número ilimitado de elementos e que P possa armazenar até n elementos. Suponha ainda que A coloque dois elementos no início de F, que B remova o elemento do final de F e o copie para o topo de P, e que C remova três elementos do topo de P. Como os semáforos de contagem podem ser usados para garantir o correto funcionamento dos processos A, B e C? Justifique a sua resposta.

Resp.: Vamos precisar de dois semáforos de contagem para a fila F e de três semáforos de contagem para a pilha P, para poder garantir o correto funcionamento dos processos A, B e C. O primeiro semáforo de F, chamado $acesso_F$, é usado para garantir o acesso exclusivo de A ou B a F. Já o segundo semáforo de F, chamado $cheia_F$, conta o número de entradas usadas em F, e é usado para bloquear B quando F estiver

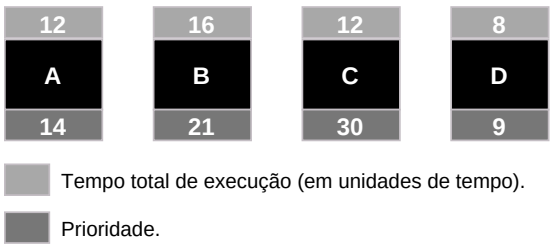
vazia. Como inicialmente F está vazia e não está sendo usada, então os semáforos $cheia_F$ e $acesso_F$ são inicializados, respectivamente, com 0 e 1. Agora, para P, o primeiro semáforo, chamado $acesso_P$, é usado para garantir o acesso exclusivo de B ou C a P. O segundo semáforo de P, chamado $vazia_P$, conta o número de entradas não usadas em P, e é usado para bloquear B quando P estiver cheia. Finalmente, o terceiro semáforo de P, chamado $cheia_P$, conta o número de entradas usadas em P, e é usado para bloquear C quando P estiver vazia. Como inicialmente P também está vazia e não está sendo usada, então os semáforos $vazia_P$, $cheia_P$ e $acesso_P$ são inicializados, respectivamente, com n, 0 e 1. A seguir mostramos os códigos para os processos A, B e C, sendo que a função $removefinal()$ remove e retorna o elemento no final de F, a função $insereinicio(elemento)$ insere o elemento *elemento* no início de F, a função $removetopo()$ remove e retorna o elemento no topo de P, e a função $inseretopo(elemento)$ insere o elemento *elemento* no topo de P.

```
void ProcessoA(void)
{
  while (1);
  {
    // Código para gerar 2 elementos e salvá-los em e1 e e2.
    // Garante o acesso exclusivo a F.
    P(acesso_F);
    // Insere os dois elementos gerados em F.
    insereinicio(e1);
    insereinicio(e2);
    // Libera o acesso exclusivo a F.
    V(acesso_F);
    // Usa a operação V sobre cheia_F para registrar que
    // 2 elementos foram inseridos em F.
    V(cheia_F);
    V(cheia_F);
  }
}
```

```
void ProcessoB(void)
{
  while (1);
  {
    // Garante que existe pelo menos um elemento em F.
    P(cheia_F);
    // Garante o acesso exclusivo a F.
    P(acesso_F);
    // Remove o elemento do final de F e o copia para e.
    e = removefinal();
    // Libera o acesso exclusivo a F.
    V(acesso_F);
    // Garante que existe pelo menos um espaço vazio em P.
    P(vazia_P);
    // Garante o acesso exclusivo a P.
    P(acesso_P);
    // Insere e no topo de P.
    inseretopo(e);
    // Libera o acesso exclusivo a P.
    V(acesso_P);
    // Usa a operação V sobre cheia_P para registrar que
    // um elemento foi inserido em P.
    V(cheia_P);
}
```

```
void ProcessoC(void)
{
  while (1);
  {
    // Garante que existem pelo menos três elementos em P.
    P(cheia_P);
    P(cheia_P);
    P(cheia_P);
    // Garante o acesso exclusivo a P.
    P(acesso_P);
    // Remove os três elementos do topo de P e os copia
    // para e_1, e_2 e e_3.
    e_1 = removetopo();
    e_2 = removetopo();
    e_3 = removetopo();
    // Libera o acesso exclusivo a P.
    V(acesso_P);
    // Usa a operação V sobre vazia_P para registrar que
    // três elementos foram removidos do topo de P.
    V(vazia_P);
    V(vazia_P);
    V(vazia_P);
    // Código para usar os 3 elementos salvos em e_1, e_2 e e_3.
  }
}
```

6. (3,0) Suponha que os processos da figura dada a seguir tenham acabado de entrar no estado **Pronto**, e que sejam os únicos processos em execução. Diga, justificando a sua resposta, quais seriam os tempos de término desses processos se o sistema operacional usasse, ao escalonar os processos:



(a) (1,5) o algoritmo de *round robin*, supondo que o quantum seja de 3 unidades de tempo e que a ordem inicial de execução tenha sido C, B, A, D.

Resp.: Pelo enunciado, vemos que a ordem de execução dos processos é como dado nas tabelas a seguir. Nestas tabelas 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 D não serem múltiplos do tamanho do quantum de 3 unidades de tempo, B somente usará 1 unidade de tempo do seu último quantum e D somente usará 2 unidades de tempo do seu último quantum. Pelas tabelas, vemos que os tempos de término dos processos A, B, C e D, são de, respectivamente, 44, 48, 38 e 35 unidades de tempo.

0	3	6	9	12	15	18	21	24
C	B	A	D	C	B	A	D	C

27	30	33	35	38	41	44	47
B	A	D	C	B	A	B	B

(b) (1,5) o algoritmo de prioridades, supondo que a prioridade seja reduzida em 3 unidades a cada 4 unidades de tempo, e que um processo execute até existir um outro processo cuja prioridade seja maior.

Resp.: A tabela a seguir é similar às tabelas do item anterior e possui somente mais uma linha, a terceira, que mostra a prioridade de cada processo antes de ele executar na unidade de tempo dada na mesma coluna desse processo. Pela tabela, vemos que os tempos de término dos processos A, B, C e D, são de, respectivamente, 44, 32, 12 e 48 unidades de tempo.

0	4	8	12	16	20	24	28	32	36	40	44
C	C	C	B	B	B	A	B	A	D	A	D
30	27	24	21	18	15	14	12	11	9	8	6