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
AD1 - Segundo Semestre de 2010

Atenção: ADs enviadas pelo correio devem ser postadas cinco dias antes da data final de entrega estabelecida no calendário de entrega de ADs.

Atenção: Tem havido muita discussão sobre a importância de que cada aluno redija suas próprias respostas às questões da AD1. Os professores da disciplina, após refletirem sobre o assunto, decidiram o seguinte: 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.

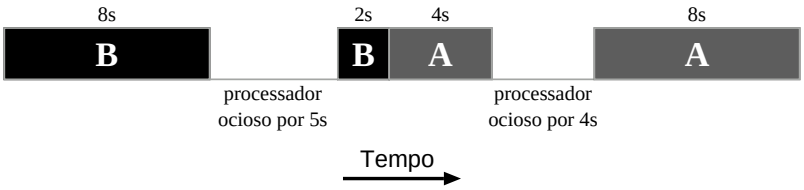
Observação: A questão 6 é opcional. Se você tentar resolvê-la, você poderá ganhar até 2,0 pontos adicionais na nota da prova, mas a sua nota ainda será limitada a 10,0 pontos.

Nome -
Assinatura -

---

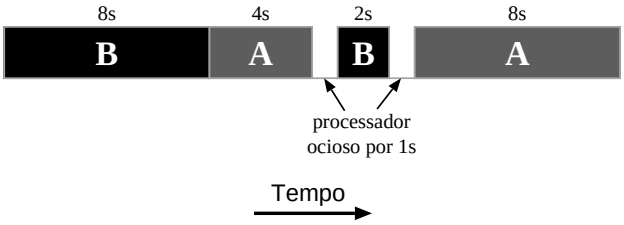
1. (2,0) Suponha que somente dois programas, A e B, estejam em execução no processador do computador. O programa B foi o primeiro a executar no processador: executou por 10s, tendo precisado fazer uma operação de E/S, com duração de 5s, após os primeiros 8s de execução. O programa A, que executou por 12s, também precisou fazer uma operação de E/S, com duração de 4s, após os primeiros 4s 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, o processador ficará ocioso? Justifique a sua resposta.

Resp.: -Se o sistema operacional não usar a multiprogramação, então não poderemos executar um outro programa quando o programa em execução fizer uma operação de E/S e, com isso, o processador ficará ocioso durante esta operação de E/S. Logo, a ordem das execuções dos programas A e B no processador será dada pela figura a seguir. Como o processador ficará ocioso durante cada operação de E/S, então o tempo de ociosidade do processador, neste caso, será a soma dos tempos das operações de E/S feitas pelos programas A e B, isto é, o tempo será de 5s+4s=9s.

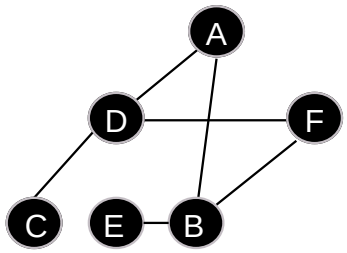


-Agora, quando o sistema operacional usar a multiprogramação, o tempo durante o qual o processador ficará ocioso poderá ser usado para executar outros programas. Neste caso, poderemos usar o tempo da operação de E/S do programa B para executar o programa A, e o tempo da operação de E/S do programa A para executar o programa B. Logo, a nova ordem de execução dos programas A e B no processador será dada pela figura a seguir. Isto ocorrerá porque o tempo da operação de E/S feita pelo programa B é maior, em um segundo, do que o tempo de execução do programa A antes de ele fazer a sua operação de E/S. Devido a isso, o processador ainda ficará ocioso por 1s antes de B voltar a executar. Além disso, o tempo de execução da

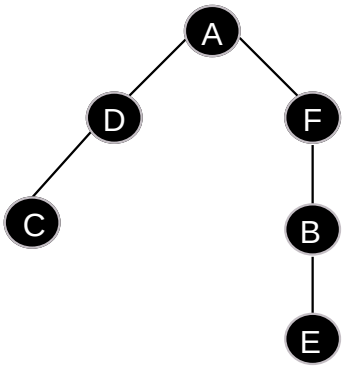
operação de E/S feita pelo programa A também é suficiente para que o programa B termine a sua execução. Pela figura, vemos que o processador ainda ficará ocioso por 2s, mas isso somente ocorrerá porque não existem outros programas para serem executados no processador.

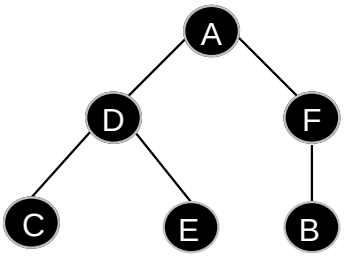


2. (2,0) Um aluno de sistemas operacionais alegou que a hierarquia dada a seguir relaciona os processos A, B, C, D, E e F em execução no sistema operacional. A alegação do aluno está correta? Justifique a sua resposta.



Resp.: Como vimos na Aula 2, os processos no primeiro nível da hierarquia não possuem um pai. Além disso, para cada processo da hierarquia em qualquer outro nível diferente do primeiro, deve existir somente um processo, no nível imediatamente anterior, que seja o seu pai. Logo, a alegação do aluno está incorreta, pois existem três erros na sua hierarquia: as conexões entre A e B (pois A não está no nível imediatamente anterior ao de B), D e F (porque D e F estão no mesmo nível) e B e E (também porque B e E estão no mesmo nível). Na figura a seguir mostramos duas possíveis hierarquias relacionando os processos A, B, C, D, E e F.

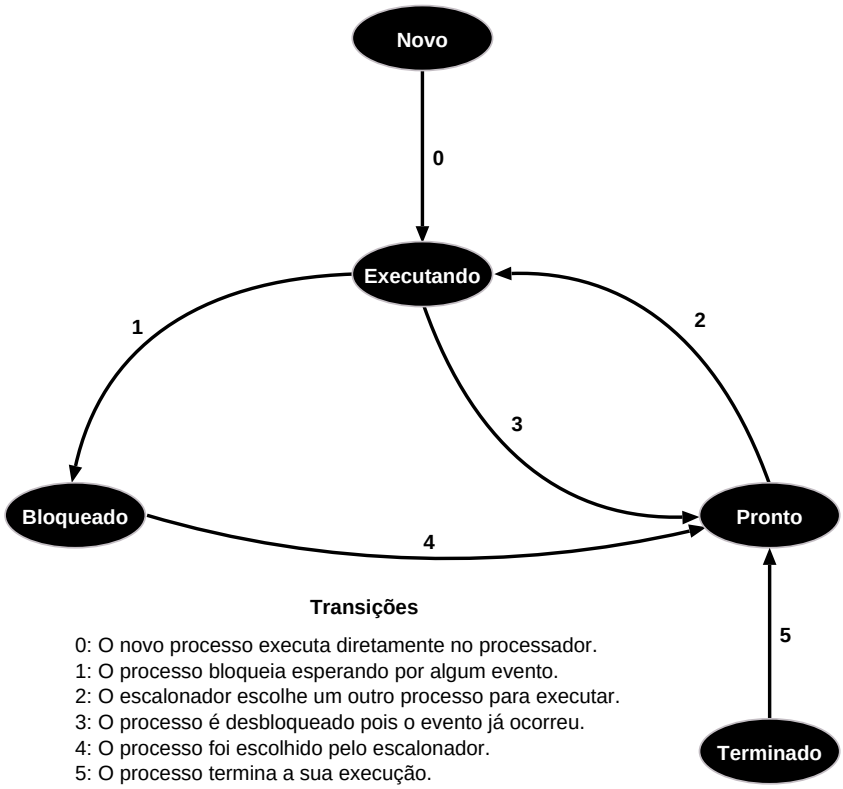




3. (2,0) Suponha que um processo execute em 10s e que durante a sua execução sejam geradas, pelo sistema operacional, 600 operações de E/S. Suponha ainda que o sistema operacional esteja executando sobre o hardware real, e que uma operação de E/S execute em 1,5ms. Se o sistema operacional agora executar sobre uma máquina virtual que reduza a velocidade das operações de E/S em 40%, e cujo processador tenha 65% da velocidade do processador real, qual será o novo tempo de execução do processo?

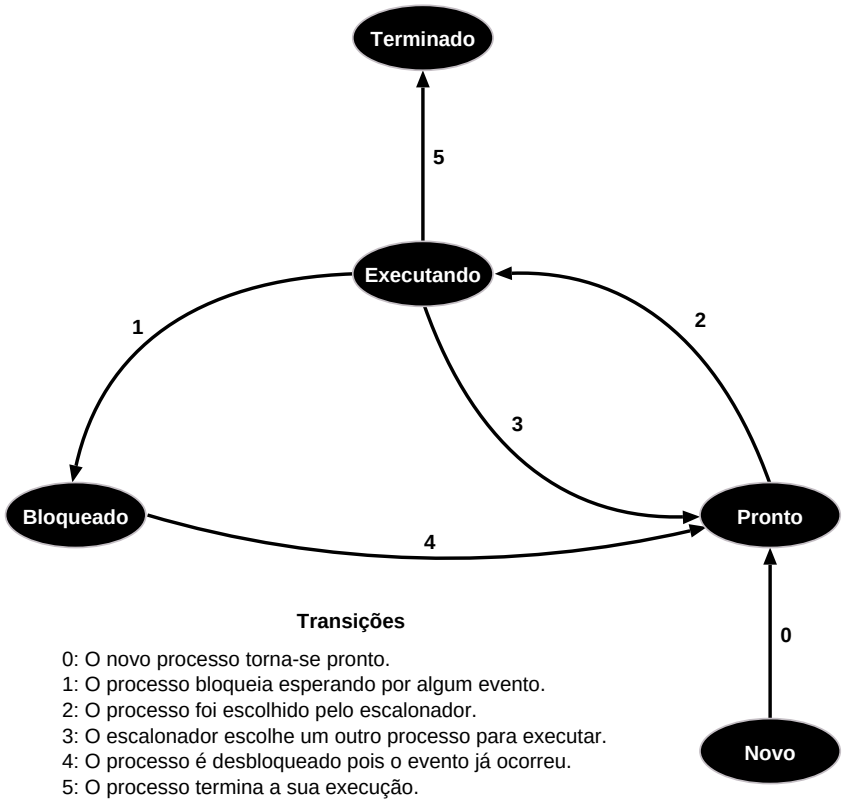
   Resp.: Pelo enunciado, o processo executa por 10s, ou seja, 10000ms. Como durante a execução do processo são geradas 600 operações de E/S, e como cada operação de E/S demora 1,5ms para ser executada, então 900ms deste tempo são gastos com as operações de E/S. Logo, o processo executa no processador do hardware por 9100ms. Considere que o processo agora execute no sistema operacional sobre a máquina virtual. Note que a velocidade de E/S mais lenta em 40% significa que, em 1,5ms, apenas 60% das instruções de uma operação de E/S podem ser executadas. Para executar 100% das instruções, é necessário um tempo de 1,5/0,6=2,5ms. Logo, as 600 operações de E/S demorarão $2,5 \times 600 = 1500$ms. Além disso, como o processador virtual possui 65% da velocidade do processador do hardware, então o programa executará no processador virtual por $9100/0,65 = 14000$ms. Logo, o tempo total de execução do processo no sistema operacional sobre a máquina virtual será de $14000 + 1500 = 15500$ms, ou seja, 15,5s.

4. (2,0) Um aluno de sistemas operacionais mostrou a seguinte versão estendida do diagrama de transição dos 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. Este diagrama está correto? Justifique a sua resposta.



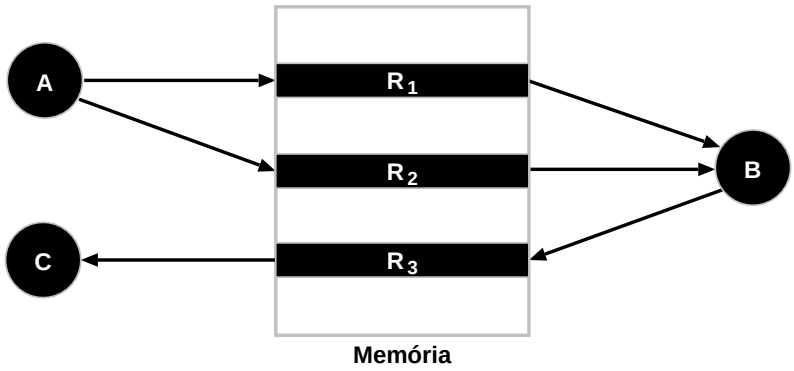
Resp.: Não, existem diversos erros no diagrama dado na figura. Quando um processo é criado, isto é, está no estado **Novo**, ele deve passar ao estado **Pronto** para esperar pela sua vez de executar no processador. Se o processo criado for crítico e precisar executar imediatamente no processador, o escalonador poderá ser chamado logo após a sua criação para colocá-lo em execução no processador (mudando o estado do processo para **Executando**). Logo, a transição 0 (assim como a sua descrição) está incorreta, e ela deve ser do estado **Novo** para o estado **Pronto**. A transição 5 do estado **Terminado** para o estado

Pronto também está incorreta, porque quando o processo termina a sua execução, ele não muda de estado. Além disso, como o processo precisa estar no estado **Executando** para terminar a sua execução, deveria existir uma transição do estado **Executando** para o estado **Terminado** (a descrição da transição 5, porém, está correta). Finalmente, com base no diagrama que vimos na Aula 4, vemos que as descrições das transições 2, 3 e 4 foram trocadas, isto é, a descrição dada na 2 é a da 3, a dada na 3 é a da 4, e a dada na 4 é a da 2. Na figura a seguir temos o diagrama correto, com todas as correções descritas anteriormente.



5. (2,0) Suponha que tenhamos três regiões de memória, R_1, R_2 e R_3, compartilhadas por três processos, A, B e C, como mostrado na figura dada a seguir. O processo A deposita dois valores, um em R_1 e o outro em R_2, os quais são consumidos pelo processo B. O processo B, por sua vez, após ler os valores de R_1 e de R_2, coloca a soma deles em R_3

para ser consumida pelo processo C. Finalmente, o processo C somente lê a soma de R_3. Como os semáforos binários podem ser usados para garantir o correto funcionamento dos processos A, B e C, supondo que eles repitam o procedimento acima indefinidamente?



Resp.: Precisamos de quatro semáforos binários, $dado\_ja\_lido_{12}$ e $dado\_disponivel_{12}$, para as regiões R_1 e R_2, e $dado\_ja\_lido_3$ e $dado\_disponivel_3$ para a região R_3. O semáforo $dado\_disponivel_{12}$ irá bloquear o processo B se o processo A ainda não tiver colocado novos valores em R_1 e R_2. Já o semáforo $dado\_ja\_lido_{12}$ irá bloquear o processo A caso o processo B ainda não tenha lido os últimos valores colocados por A em R_1 e em R_2. De modo similar, o semáforo $dado\_disponivel_3$ irá bloquear o processo C se o processo B ainda não tiver colocado um novo valor em R_3, e o semáforo $dado\_ja\_lido_3$ irá bloquear o processo B se o processo C ainda não tiver lido o último valor colocado por B em R_3. Como inicialmente os processos não estarão em execução, então não existirão valores válidos em R_1, R_2 e R_3 e, com isso, os valores dos semáforos $dado\_disponivel_{12}$ e $dado\_disponivel_3$ serão ambos 0, e os valores dos semáforos $dado\_ja\_lido_{12}$ e $dado\_ja\_lido_3$ serão ambos 1. A seguir damos um esboço (em C) dos procedimentos ProcessoA, ProcessoB e ProcessoC que devem ser executados, respectivamente, pelos processos A, B e C. A função GerarValor gera um novo valor para ser colocado em R_1 ou em R_2 e a função UsarSoma usa a soma (de $R1$ e $R2$) lida de R_3.

```
void ProcessoA(void)
{
  int Valor1, Valor2;
  while (1)
  {
    Valor1 = GerarValor();
    Valor2 = GerarValor();
    P(dado_ja_lido12);
    R1 = Valor1;
    R2 = Valor2;
    V(dado_disponivel12);
  }
}
```

```
void ProcessoB(void)
{
  int Soma;
  while (1)
  {
    P(dado_disponivel12);
    Soma = R1 + R2;
    V(dado_ja_lido12);
    P(dado_ja_lido3);
    R3 = Soma;
    V(dado_disponivel3);
  }
}
```

```
void ProcessoC(void)
{
  int Soma;
  while (1)
  {
    P(dado_disponivel3);
    Soma = R3;
    V(dado_ja_lido3);
    UsarSoma(Soma);
  }
}
```

6. (2,0) Suponha que um processo requeira, em média, um tempo T de execução antes que se bloqueie por E/S. Suponha também que uma comutação entre processos requeira um tempo S, o qual é efetivamente desperdiçado (*overhead*). Para o escalonamento por *round robin* com quantum $Q = xS$, sendo x um número real qualquer, dê uma equação para a eficiência do processador (fração do tempo usada para executar os processos) para cada um dos seguintes casos:

   (a) (1,0) $Q > T$.

   (b) (1,0) $Q < T$.

   Resp.: Note que $x > 0$, pois o quantum, por ser um tempo, não pode ser negativo e, se fosse nulo, não permitiria que um processo executasse

no processador. Note também que a eficiência do processador será o tempo T de execução do processo antes de bloquear por E/S sobre a soma deste tempo T com o tempo gasto com as trocas de contexto efetuadas até este processo bloquear.

(a) Como T é menor do que Q, então o processo executará no processador por um tempo T antes de bloquear, e depois será necessário fazer uma troca de contexto, o que irá requerer um tempo $S = Q/x$. Logo, a eficiência do processador será de

$$\frac{T}{T+S} = \frac{T}{T+\frac{Q}{x}} = \frac{xT}{xT+Q}.$$

(b) Se Q for menor do que T, então teremos que fazer T/Q trocas de contexto até que o processo seja finalmente bloqueado. Com isso, o tempo gasto com as trocas de contexto será de

$$S\frac{T}{Q} = \frac{Q}{x}\frac{T}{Q} = \frac{T}{x}.$$

Logo, a eficiência do processador será de

$$\frac{T}{T+\frac{T}{x}} = \frac{x}{x+1}.$$