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 - Primeiro Semestre de 2011

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.

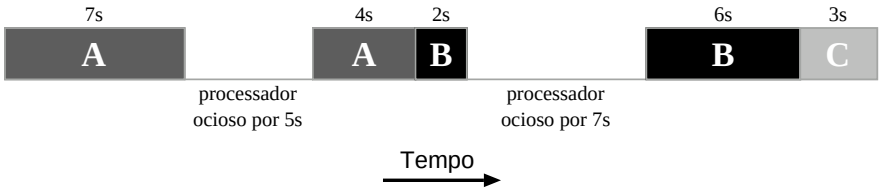
Observação: A questão 5 é opcional. Se você tentar resolvê-la, você poderá ganhar até 2,0 pontos adicionais na nota da prova, mas a sua nota ainda será limitada a 10,0 pontos.

Nome -
Assinatura -

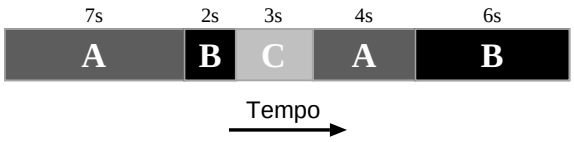
---

1. (3,0) Suponha que somente três programas, A, B e C, estejam em execução no processador do computador. O programa A executou primeiro no processador, por 11s, e precisou fazer uma operação de E/S, com duração de 5s, após os primeiros 7s de execução. O programa B, que foi o segundo a executar, por 8s, também precisou fazer uma operação de E/S, com duração de 7s, após os primeiros 2s de execução. Finalmente, o programa C foi o último a executar no processador, e executou por 3s sem fazer nenhuma operação de E/S. Qual será o tempo de ociosidade do processador se o sistema operacional não usar a multiprogramação? E se o sistema usar a multiprogramação? Justifique a sua resposta.

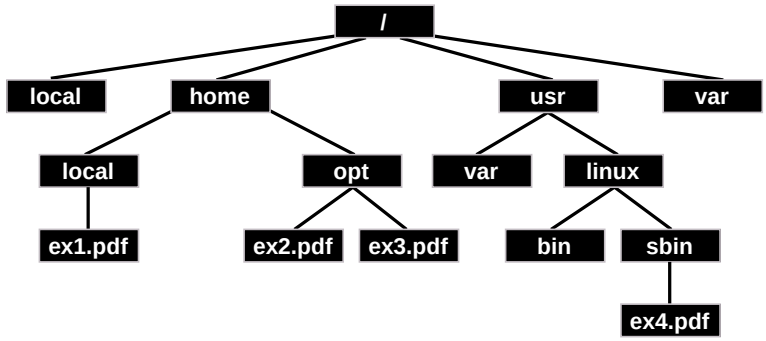
Resp.: -Se o sistema operacional não usar a multiprogramação, então não poderemos executar outro programa durante uma operação de E/S e o processador ficará ocioso. O que ocorrerá está mostrado na figura a seguir e o tempo de ociosidade do processador será de $5s + 7s = 12s$.



-Se o sistema operacional usar a multiprogramação, o tempo de ociosidade poderá ser usado para executar outros programas. Nesse caso, poderemos usar o tempo de execução da operação de E/S do programa A para executar o programa B até ele fazer E/S e depois executar completamente o programa C. Poderemos também usar o tempo da operação de E/S do programa B para executar o restante do programa A. Logo, a nova ordem de execução dos programas será como mostrado na figura a seguir, onde vemos que o processador não ficará mais ocioso.



2. (2,0) Considere a hierarquia de diretórios dada a seguir, e suponha que você está no diretório /usr/linux/bin. Com relação a este diretório, que arquivo está no diretório cujo nome é ../sbin/../../var/../../local/../home/opt/../local, supondo que ".." indica o diretório pai de um diretório? Qual é o nome mais sucinto para este arquivo, ainda relativo ao diretório /usr/linux/bin? Este nome é único?



Resp.: -Na tabela dada a seguir mostramos, passo a passo, cada um dos diretórios alcançados ao seguirmos o nome do diretório dado na questão. Como podemos ver pela última linha da tabela, o arquivo é ex1.pdf.

Nome do diretório (parcial)	Diretório atual
..	/usr/linux
../sbin	/usr/linux/sbin
../sbin/..	/usr/linux
../sbin/../..	/usr
../sbin/../../var	/usr/var
../sbin/../../var/..	/usr
../sbin/../../var/../..	/
../sbin/../../var/../../local	/local
../sbin/../../var/../../local/..	/
../sbin/../../var/../../local/../home	/home
../sbin/../../var/../../local/../home/opt	/home/opt
../sbin/../../var/../../local/../home/opt/..	/home
../sbin/../../var/../../local/../home/opt/../local	/home/local

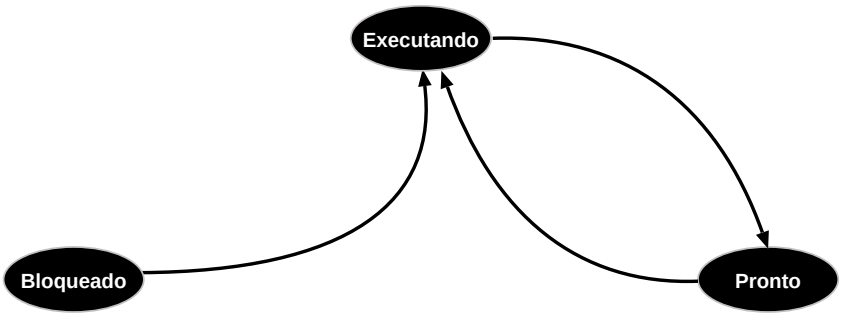
-Para obter o nome mais sucinto do arquivo, basta usarmos ".."até atin-

gir o primeiro diretório do qual o arquivo seja um descendente. Este diretório é / e o nome mais sucinto é ../../../home/local/ex1.pdf.
-Sim, o nome do arquivo é único porque representa o caminho mais curto entre dois pontos na árvore e este é único. Note que existem outros nomes não relativos a /usr/linux/bin que podem ser mais sucintos, como por exemplo /home/local/ex1.pdf, o nome absoluto (isto é, relativo a /).

3. (3,0) Suponha que um sistema operacional esteja executando sobre a máquina real. Suponha ainda que o processador virtual possua uma velocidade de processamento 40% menor do que a do processador real, e que, ao usar a máquina virtual, o tempo de execução de uma operação de E/S, que é de 15ms na máquina real, aumente em 3ms. Se um programa que fez 50 operações de E/S executar em 42s na máquina real, qual será o tempo de execução sobre a máquina virtual?

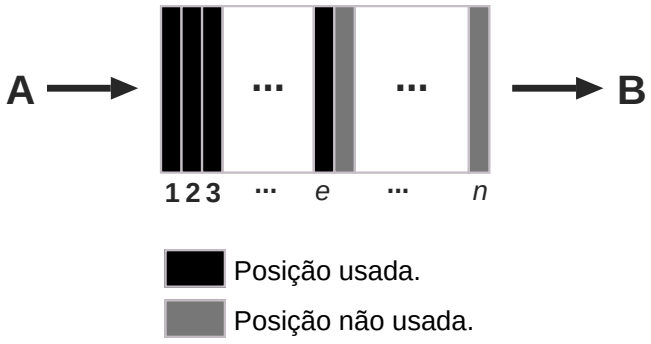
   Resp.: Pelo enunciado, o programa executa por 42s, ou seja, 42000ms. Como durante a execução são feitas 50 operações de E/S, e como cada operação de E/S demora 15ms, então 750ms desse tempo são gastos com elas. Logo, o programa executa no processador do hardware por 41250ms. Considere agora que o programa execute sobre a máquina virtual. Note que a velocidade do processador virtual ser 40% menor significa que, durante os 41250ms, somente 60% das instruções serão executadas. Com isso, no processador virtual, o programa executará em $41250/0,60 = 68750$ms. Como o tempo de cada operação de E/S é agora de 18ms, então o tempo total de E/S passará para $18 \times 50 = 900$ms. Logo, o tempo total de execução do programa na máquina virtual será de $68750 + 900 = 69650$ms, ou seja, $69,65$s.

4. (2,0) Considere o diagrama de transição entre os estados dos processos dado na figura a seguir. Todas as transições de estados estão corretas? Existe alguma transição entre os estados que não foi dada no diagrama?



Resp.: -Existe uma transição incorreta no diagrama: a do estado Bloqueado para o estado Executando. Essa transição não existe no diagrama pois, quando um processo é desbloqueado, ele deve sempre passar ao estado Pronto, já que é sobre os processos nesse estado que o escalonador atua.
-Duas transições não foram dadas no diagrama: a transição do estado Executando para o estado Bloqueado e a transição, mencionada acima, do estado Bloqueado para o Pronto. A primeira transição ocorre quando um processo em execução é bloqueado para esperar por um evento externo.

5. (2,0) Suponha que dois processos, A e B, estejam gerenciando a fila dada na figura a seguir. Esta fila, que inicialmente tem e elementos, tem espaço para armazenar até n elementos. Suponha ainda que A insira elementos no início da fila e que B remova elementos do final da fila. Como os semáforos poderão ser usados para garantir o correto funcionamento dos processos A e B?



Resp.: Podemos usar três semáforos, um binário e dois de contagem, para garantir o correto funcionamento dos processos A e B. O semáforo

binário *fila_liberada* garantirá o acesso exclusivo à fila. Já o semáforo de contagem *posições_vagas* armazenará o número de entradas não usadas da fila e será usado para bloquear A quando a fila estiver cheia. Finalmente, o semáforo de contagem *posições_ocupadas* armazenará o número de elementos na fila e será usado para bloquear B quando a fila estiver vazia. Em relação ao valor inicial dos semáforos, o do semáforo *fila_liberada* será 1 porque inicialmente nenhum processo está em execução. Os dos semáforos *posições_vagas* e *posições_ocupadas* serão, respectivamente, de $n - e$ e e, porque a fila tem n entradas e inicialmente possui e elementos. A função *Remover* dada a seguir deverá ser usada por B para obter o elemento do final da fila, e a função $Inserir$ deverá ser usada por A para inserir um elemento no início da fila. Note que A será bloqueado se a fila estiver cheia, sendo desbloqueado por B quando este remover um elemento da fila. Note também que B será bloqueado quando a fila estiver vazia, sendo desbloqueado por A quando este inserir um elemento na fila. Nos códigos dados a seguir, a função $RemoveFila$ remove e retorna o elemento do início da fila e a função $InsereFila$ insere um elemento no final da fila.

```
elemento Remover(void)
{
  elemento Primeiro;
  P(posições_ocupadas);
  P(fila_liberada);
  Primeiro = RemoveFila();
  V(fila_liberada);
  V(posições_vagas);
  return Primeiro;
}

void Inserir(elemento Novo)
{
  P(posições_vagas);
  P(fila_liberada);
  InsereFila(Novo);
  V(fila_liberada);
  V(posições_ocupadas);
}
```