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 2012

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 dois programas, A e B, estejam para ser executados no processador. O programa A executa por 12s, sendo que 25% desse tempo é gasto esperando pelo término de uma operação de E/S. Já o programa B, que não faz operações de E/S, executa por 3s no processador. Se o sistema operacional não implementar o conceito de multiprogramação, qual será o tempo de ociosidade do processador se A executar antes de B? E se o sistema operacional passar a usar a multiprogramação, a ociosidade ainda ocorrerá? Justifique a sua resposta.

   Resp.: Pelo enunciado, vemos que o tempo gasto em operações de E/S pelo programa A, igual a 25% do seu tempo total de execução de 12s, é de $12 \times 0,25$s ou 3s. Se o sistema operacional não usa a multiprogramação, então o processador fica ocioso somente pelos 3s necessários à execução da operação de E/S feita por A, pois B não pode executar durante esses 3s, e B não faz operações de E/S. Porém, se a multiprogramação é usada, o processador não fica ocioso, pois um dos objetivos da multiprogramação é exatamente o de evitar a ociosidade do processador durante a execução de operações de E/S e, coincidentemente, B precisa executar por 3s sem fazer operações de E/S.

2. (2,0) Suponha que um computador possa executar 3 bilhões de instruções por segundo, e que uma chamada ao sistema requeira 1250 instruções, incluindo a de TRAP e todas as necessárias à troca de contexto. Quantas chamadas ao sistema o computador pode executar por segundo para ainda possuir dois terços da capacidade do processador para executar códigos de aplicação? Justifique a sua resposta.

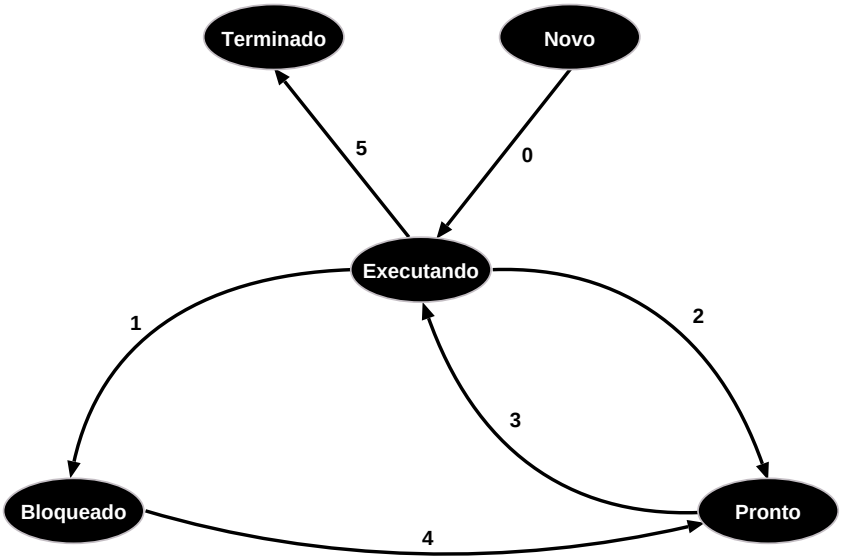
   Resp.: Como dois terços da capacidade do processador devem estar disponíveis para executar código de aplicação, então podemos usar até $\frac{1}{3} \times 3 = 1$ bilhão de instruções por segundo para executar as chamadas ao sistema operacional. Como cada chamada ao sistema requer 1250 instruções, então podemos executar até $\frac{1000000000}{1250} = 800000$ chamadas ao sistema por segundo.

3. (2,0) Suponha que um processo execute em 9s e que durante a sua execução sejam geradas, pelo sistema operacional, 2000 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 na qual cada operação de E/S executa em 2ms, e na qual a velocidade do processador virtual é 75% da velocidade do processador real, qual será o novo tempo de execução do processo, supondo que ele execute a mesma quantidade de operações de E/S? Justifique a sua resposta.

Resp.: Pelo enunciado, o programa executa por 9s, ou seja, 9000ms. Como durante a execução são feitas 2000 operações de E/S, e como cada operação de E/S demora 1,5ms, então 3000ms dos 9000ms são gastos com elas. Logo, o programa executa no processador do hardware por 6000ms. Considere agora que o programa execute sobre a máquina virtual. Note que a velocidade do processador virtual ser 75% da velocidade do processador real significa que, durante os 6000ms, somente 75% das instruções serão executadas. Com isso, no processador virtual, o programa executará em 6000/0,75=8000ms. Como o tempo de cada operação de E/S é agora de 2ms, então o tempo total de E/S passará para $2 \times 2000 = 4000$ms. Logo, o tempo total de execução do programa na máquina virtual será de $8000 + 4000 = 12000$ms, ou seja, 12s.

4. (2,0) Na figura dada a seguir mostramos uma 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. Esse diagrama está correto? Justifique a sua resposta.



Transições

0: O novo processo inicia a sua execução.
1: O escalonador escolhe um outro processo para executar.
2: O processo bloqueia esperando por algum evento.
3: O processo é desbloqueado pois o evento já ocorreu.
4: O processo volta a executar no processador.
5: O processo termina a sua execução.

Resp.: Não, porque existe um erro na figura. A transição 0 deveria ser do estado **Novo** para o estado **Pronto**, pois o processo recém-criado pode ser menos prioritário do que o processo em execução ou do que algum outro processo no estado **Pronto**. Note que se o novo processo for o mais prioritário e, adicionalmente, precisar ser executado imediatamente, bastará o sistema operacional chamar o escalonador para colocá-lo em execução.

5. (2,0) Suponha que três processos, A, B e C, compartilhem uma pilha, usada para armazenar números. Suponha ainda que essa pilha, inicialmente vazia, possa armazenar até n números, e que não exista uma variável para contabilizar a quantidade de números armazenados na pilha. O processo A continuamente insere números na pilha. Já o processo B continuamente remove dois números da pilha, calcula o produto desses números, e depois insere o resultado do produto na pilha. Finalmente, o processo C remove um número da pilha, calcula o

seu módulo, e o imprime na saída padrão. Como os semáforos devem ser usados para garantir a correta execução dos processos A, B e C? Justifique a sua resposta.

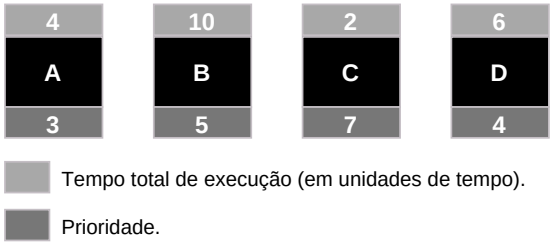
Resp.: Precisamos usar três semáforos para garantir o correto funcionamento dos processos A, B e C: dois de contagem, *vazia* e *cheia*, e um binário, *acesso*. O semáforo binário *acesso* garante o acesso exclusivo de um processo à pilha, e é inicializado com 1 porque inicialmente nenhum processo está acessando a pilha. Já o semáforo de contagem *vazia* conta o número de entradas usadas na pilha, e é usado para bloquear os processos B e C quando a pilha estiver vazia. Finalmente, o semáforo de contagem *cheia* conta o número de entradas vazias na pilha, e é usado para bloquear o processo A quando a pilha estiver cheia. Como inicialmente a pilha está vazia, então os semáforos *cheia* e *vazia* são inicializados, respectivamente, com n e 0. A seguir mostramos os códigos para os processos A, B e C, sendo que a função $removetopo()$ remove e retorna o número no topo da pilha, e a função $inseretopo(num)$ insere o número dado em num no topo da pilha.

```
void ProcessoA(void)
{
  while (1);
  {
    P(cheia);
    P(acesso);
    // Código para gerar um número e salvá-lo em num.
    inseretopo(num);
    V(acesso);
    V(vazia);
  }
}
```

```
void ProcessoB(void)
{
  while (1);
  {
    P(vazia); // Garante a existência do primeiro número.
    P(vazia); // Garante a existência do segundo número.
    P(acesso);
    num1 = removetopo();
    num2 = removetopo();
    inseretopo(num1 × num2);
    V(acesso);
    V(cheia);
  }
}

void ProcessoC(void)
{
  while (1);
  {
    P(vazia);
    P(acesso);
    num = abs(removetopo());
    // Código para imprimir o número em num;
    V(acesso);
    V(cheia);
  }
}
```

6. (2,0) Suponha que os processos da figura dada a seguir tenham acabado de entrar no estado **Pronto**, e que sejam os únicos processos em execução. Quais seriam os tempos de término desses processos se o sistema operacional usasse, ao escalonar os processos:



(a) (1.0) o algoritmo de *round robin*, supondo que o quantum é de 2 unidades de tempo, que A executa pela primeira vez antes de B, C e D, que C executa pela primeira vez antes de B e D, e que D executa pela primeira vez antes de B?

Resp.: Como vimos na aula 6, o algoritmo *round robin* executa os processos **Prontos** de modo alternado, sendo que cada processo executa por um tempo igual ao **quantum**. Além disso, um processo somente volta a executar, se necessário, por mais um **quantum**, quando todos os outros processos no estado **Pronto** também já tiverem executado por um **quantum**. Pelo enunciado, vemos que a ordem de execução dos processos é como a dada na tabela a seguir. Nesta tabela mostramos como os processos são escolhidos pelo algoritmo, sendo que cada coluna mostra a execução de um processo dando o tempo de início de cada quantum e o processo correspondente. Pela tabela, vemos que os tempos de término dos processos A, B, C e D são de, respectivamente, 10, 22, 4 e 16 unidades de tempo.

0	2	4	6	8	10	12	14	16	18	20
A	C	D	B	A	D	B	D	B	B	B

(b) (1.0) o algoritmo de prioridades, supondo que a prioridade é reduzida em 2 unidades a cada 1 unidade de tempo, e que um processo executa até existir um outro processo cuja prioridade seja maior?

Resp.: A execução do algoritmo é como a dada na tabela a seguir, mostrada em duas partes. Como existem diversas possibilidades devido à prioridade dos processos ainda em execução se tornarem

iguais após algum tempo, a tabela somente mostra uma dessas possibilidades. Cada coluna da tabela refere-se a um processo, mostrando a unidade de tempo em que executou (primeira linha) e a prioridade com que executou (terceira linha). Como podemos ver pela tabela, alguns dos possíveis tempos de término são de, respectivamente, 16, 22, 2 e 19 unidades de tempo.

0	1	2	3	4	5	6	7	8	9	10
C	C	B	B	D	D	A	B	B	D	A
7	6	5	4	4	3	3	3	2	2	2

11	12	13	14	15	16	17	18	19	20	21
A	B	D	D	A	B	B	D	B	B	B
1	1	1	0	0	0	-1	-1	-2	-3	-4