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 2009

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.

Nome -
Assinatura -

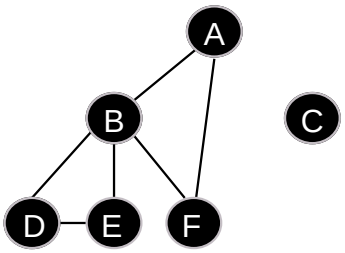
---

1. (1.5) Suponha que dois programas, A e B, estejam para serem executados no processador. O programa A executa por 6s, sendo que 20% deste 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 2s no processador. Se o sistema operacional não implementa o conceito de multiprogramação, o processador poderá ficar ocioso? Em caso afirmativo, qual será o tempo de ociosidade do processador? Justifique a sua

resposta.

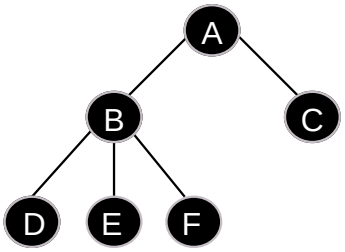
Resp.: -Sim, pois sem a multiprogramação o processador ficará ocioso quando o programa atualmente em execução fizer uma operação de E/S, até que esta operação termine e o programa possa continuar a sua execução.
-Pelo enunciado da questão, o programa A executa por 6s e 20% deste tempo é usado pela operação de E/S. Logo, o processador ficará ocioso por $0,2 \times 6 = 1,2$s. Note que o programa B não afeta o tempo de ociosidade do processador, pois o sistema não usa a multiprogramação.

2. (1.5) 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 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: os dois ciclos da hierarquia e o processo C, que está no segundo nível da hierarquia e não tem um pai. O processo C ou deveria ser filho do processo A (se ele continuasse no segundo nível), ou deveria estar no primeiro nível. Em relação ao ciclo que envolve os processos A, B e F, o processo A não poderia ser pai de F, pois A está no primeiro nível e F está no terceiro nível. Finalmente, para o ciclo envolvendo os processos B, D e E, D não poderia ser pai de E (ou E não poderia ser pai de D), pois ambos estão no terceiro nível. Na figura a seguir

mostramos uma possível hierarquia relacionando os processos A, B, C, D, E e F.



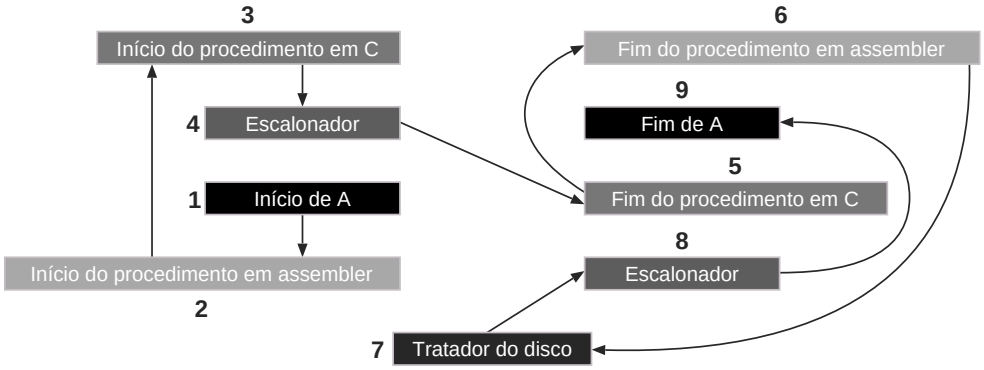
3. (1.5) Suponha que um processo execute em 5s e que durante a sua execução sejam geradas, pelo sistema operacional, 500 operações de E/S. Suponha ainda que o sistema operacional esteja executando sobre o hardware real, e que uma operação de E/S execute em 2ms. Se o sistema operacional agora executar sobre uma máquina virtual que reduz a velocidade das operações de E/S em 25%, e cuja velocidade do processador virtual é 80% da velocidade do processador real, qual será o novo tempo de execução do processo?

   Resp.: Pelo enunciado, o processo executa por 5s, ou seja, 5000ms. Como durante a execução do processo são executadas 500 operações de E/S, e como cada operação de E/S demora 2ms para ser executada, então 1000ms deste tempo serão gastos pelas operações de E/S. Com isso, o processo executará no processador do hardware real por 4000ms. Quando o processo executar no sistema operacional sobre a máquina virtual, que reduz a velocidade das operações de E/S em 25%, cada operação de E/S gastará agora $2 + 0,25 \times 2 = 2,5$ms e, com isso, as 500 operações de E/S executarão agora em $2,5 \times 500 = 1250$ms. Além disso, como o processador virtual possui 80% da velocidade do processador real, então o programa executará no processador virtual por $4000 + 0,25 \times 4000 = 5000$ms (note que o tempo de execução será 25% maior do que no processador real). Logo, o tempo total de execução do processo no sistema operacional sobre a máquina virtual será de $5000 + 1250 = 6250$ms, ou seja, 6,25s.

4. (1.5) Suponha que tenha ocorrido uma interrupção do disco enquanto o processo A estava em execução no sistema. Na figura a seguir damos

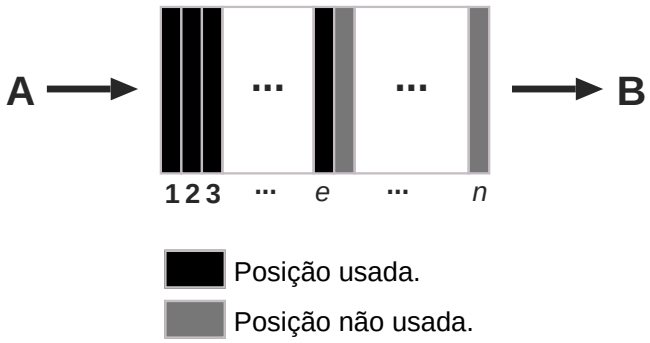
as ações que ocorrem quando uma interrupção é tratada pelo sistema operacional. Dê a ordem em que as ações são executadas no tempo.

Resp.: Na figura a seguir damos a ordem correta em que as ações são executadas. Note que o processo tratador do disco, executado devido à primeira chamada ao escalonador, somente será executado após o tratamento da interrupção ser finalizado. Já a segunda chamada ao escalonador fará com que A imediatamente inicie a sua execução no ponto em que parou anteriormente (note que, neste caso, A precisa ser o processo mais prioritário).



5. (2.0) Suponha que dois processos, A e B, estejam gerenciando uma fila, inicialmente vazia, que possui 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. O relacionamento entre os processos A e

B pode ser visto na figura dada a seguir. Se o número de elementos atualmente na fila for dado pela variável e, como os semáforos binários poderão ser usados para garantir o correto funcionamento dos processos A e B?



Resp.: Precisaremos usar três semáforos binários para garantir o correto funcionamento dos processos A e B: *acesso*, *cheio* e *vazio*. O semáforo *acesso* garantirá o acesso exclusivo à fila, e será inicializado com 1, pois inicialmente nenhum dos processos está acessando a fila. Já o semáforo *cheio* será usado para bloquear A caso a fila esteja cheia, e será inicializado também com 1, pois inicialmente a fila não está cheia. Finalmente, o semáforo *vazio* será usado para bloquear B quando a fila estiver vazia, e será inicializado com 0, pois a fila está inicialmente vazia. 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, e será desbloqueado por B quando este remover um elemento da fila. Note também que B será bloqueado quando a fila estiver vazia, e será 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(acesso);
  if (e == 0) // Fila vazia.
```

```
  {
    V(acesso);
    P(vazio);
    P(acesso);
  }
  Primeiro = RemoveFila();
  e = e − 1;
  if (e == n − 1) // Desbloqueia B, se ele estiver bloqueado,
    V(cheio);      // pois a fila estava cheia.
  V(acesso);
  return Primeiro;
}

void Inserir(elemento Novo)
{
  P(acesso);
  if (e == n) // Fila cheia.
  {
    V(acesso);
    P(cheio);
    P(acesso);
  }
  InsereFila(Novo);
  e = e + 1;
  if (e == 1)  // Desbloqueia A, se ele estiver bloqueado,
    V(vazio); // pois a fila estava vazia.
  V(acesso);
}
```

6. (2.0) Suponha que o sistema operacional use o algoritmo por prioridades ao escalonar os processos. Suponha ainda que a prioridade do processo em execução seja reduzida a cada unidade de tempo e que ele execute enquanto a sua prioridade é a maior. Será possível executarmos os processos do sistema de acordo com o algoritmo do trabalho mais curto primeiro, se soubermos o tempo de execução de todos os processos a serem executados? Justifique a sua resposta.

Resp.: Sim, será possível, se atribuirmos as prioridades corretas aos processos. Pelo enunciado da questão, vemos que a priori conhecemos quantos processos vão executar no sistema operacional. Vamos supor que existem n processos, e que P_1, P_2, $\cdots$, P_n são os processos ordenados segundo uma ordem não-decrescente de acordo com os seus tempos de execução. Vamos supor também que o tempo de execução do processo P_i, $1 \leq i \leq n$, desta ordenação seja t_i unidades de tempo. Seja T a soma destes tempos, isto é, $T = \sum_{1 \leq i \leq n} t_i$. Ao processo P_1 vamos atribuir a prioridade $q_1 = T$, e ao processo P_i, $1 < i \leq n$, a prioriodade $q_i = T - \sum_{1 \leq j \leq i-1} t_j$. Usando estas prioridades, os processos executarão de acordo com o algoritmo do trabalho mais curto primeiro, como veremos a seguir através de um exemplo. Suponha que 5 processos A, B, C, D e E, cujos tempos de execução são de, respectivamente, 3, 2, 4, 2 e 1 unidades de tempo, vão ser executados no sistema operacional. Ordenando estes processos em ordem não-decrescente de acordo com os tempos de execução, obteremos a ordem E, B, D, A e C para os processos (a ordem também poderia ser E, D, B, A e C). Logo $P_1 = E$, $P_2 = B$, $P_3 = D$, $P_4 = A$ e $P_5 = C$. A soma T dos tempos será $1 + 2 + 2 + 3 + 4 = 12$. As prioridades atribuídas aos processos serão $q_1 = 12$, $q_2 = 12 - 1 = 11$, $q_3 = 12 - (1 + 2) = 12 - 3 = 9$, $q_4 = 12 - (1 + 2 + 2) = 12 - 5 = 7$ e $q_5 = 12 - (1 + 2 + 2 + 3) = 12 - 8 = 4$ e, com isso, obteremos a ordem de execução dada na tabela a seguir. Nesta tabela damos as prioridades na primeira linha, os processos na segunda linha, e as unidades de tempo na última linha. Como podemos notar por esta tabela, cada processo executará até o seu término sem ser interrompido por um outro processo. Logo, as prioridades foram corretamente escolhidas, pois a ordem de execução dos processos, segundo o algoritmo do trabalho mais curto primeiro, será E, B, D, A e C, que será exatamente a ordem em que os processos serão executados pelo algoritmo de prioridades.

12	11	10	9	8	7	6	5	4	3	2	1
E	B	B	D	D	A	A	A	C	C	C	C
0	1	2	3	4	5	6	7	8	9	10	11