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
AP1 - Segundo Semestre de 2006

Nome -
Assinatura -

Observações:

1. Prova sem consulta e sem uso de máquina de calcular.
2. Use caneta para preencher o seu nome e assinar nas folhas de questões e nas folhas de respostas.
3. Você pode usar lápis para responder as questões.
4. Ao final da prova devolva as folhas de questões e as de respostas.
5. Todas as respostas devem ser transcritas nas folhas de respostas. As respostas nas folhas de questões não serão corrigidas.

---

1. (1.5) Sobre a classificação de sistemas operacionais (SO), descreva os possíveis tipos de SO quanto a:

   (a) (0.75) Número de usuários;

   Resp.: Um sistema operacional é dito monousuário se somente um único usuário puder acessar o sistema de uma vez, isto é, um outro usuário somente poderá usar o sistema quando o usuário atual deixar de usar o sistema. Já em um sistema multiusuário, vários usuários podem acessar simultaneamente o sistema operacional, isto é, um usuário não precisa esperar um outro usuário deixar de usar o sistema para usá-lo.

   (b) (0.75) Número de programas.

   Resp.: Um sistema monoprogramado permite que somente um programa seja executado por vez no sistema, isto é, um novo programa somente poderá ser executado após o programa atualmente em execução terminar. Já em um sistema multiprogramado, podemos ter vários programas em execução no sistema, sendo que um programa não precisará esperar o término do outro para começar a ser executado (os programas poderão ou não executar simultaneamente, dependendo do número de processadores do computador).

2. (1.5) Qual é o objetivo das chamadas ao sistema operacional? Dê uma descrição de como estas funcionam, evidenciando qual o papel das bibliotecas.

   Resp.:-As chamadas ao sistema operacional objetivam fornecer uma interface entre os processos executando no modo usuário e as diversas funções oferecidas pelo sistema operacional.
   -Quando um processo deseja fazer uma chamada ao sistema, este deverá fazer uma chamada a uma das funções da biblioteca que tratam dos detalhes de executar esta chamada. O motivo de usarmos a biblioteca é porque além de esta fornecer funções que facilitam ou estendem a funcionalidade de cada chamada ao sistema, esta também facilita a execução da própria chamada, pois o modo de implementar uma chamada depende do hardware em que o sistema operacional está exe-

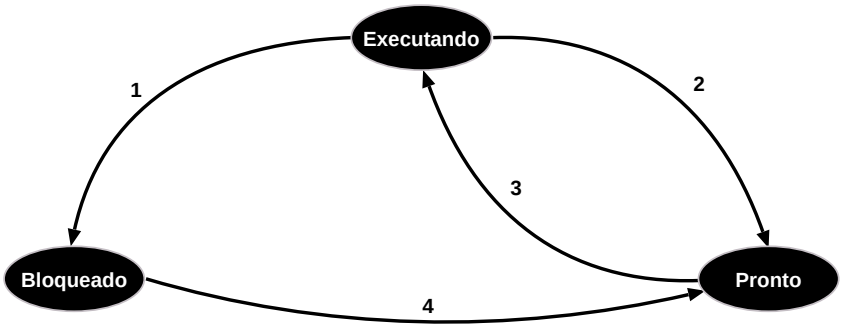
cutando. Antes de chamar a função da biblioteca, o processo deverá colocar na pilha os parâmetros requeridos por esta função. Depois de o processo chamar a função, esta determinará qual chamada ao sistema deverá ser executada, colocará os parâmetros desta chamada no local em que o sistema operacional espera que estes estejam (um identificador para a chamada, e os parâmetros da chamada), e executará uma instrução TRAP (cuja implementação dependerá do hardware) para passar o controle ao sistema operacional (comutando o processador do modo usuário para o modo supervisor). Depois de o sistema operacional ser chamado, este determinará, através do identificador, o endereço da chamada que foi executada (usando este identificador como um índice em uma tabela com os endereços de todas as chamadas), e saltará para este endereço. O código para o qual o sistema saltou, chamado de o tratador da chamada, executará então as tarefas necessárias à execução desta chamada ao sistema, bloqueando o processo que fez a chamada à biblioteca se for necessário esperar a ocorrência de algum evento externo. Caso este tratador termine sem bloquear o processo, o sistema operacional executará uma instrução de retorno de uma TRAP, que fará o processador retornar ao modo usuário, e continuar a execução a partir da instrução imediatamente posterior à instrução TRAP executada. A função da biblioteca então devolverá o controle ao processo, que deverá remover os parâmetros da pilha para finalizar corretamente a chamada à função da biblioteca.

3. (1.5) Qual é a principal vantagem do modelo de micronúcleo considerando múltiplos processadores?

   Resp.: No modelo de micronúcleo a maior parte das funcionalidades do sistema são implementadas no modo usuário, por processos servidores. O micronúcleo, que executa no modo supervisor, somente trata do acesso direto aos dispositivos de E/S (que não pode ser feito no modo usuário), e dos detalhes necessários à troca de mensagens entre os processos do sistema, pois os serviços do sistema operacional, como as chamadas ao sistema, devem ser solicitados através do envio de mensagens aos servidores que executam estes serviços. A vantagem é exatamente o fato de usarmos mensagens para que o sistema execute tarefas para os processos (chamados de clientes), pois nada impede que os servidores estejam em processadores ou computadores

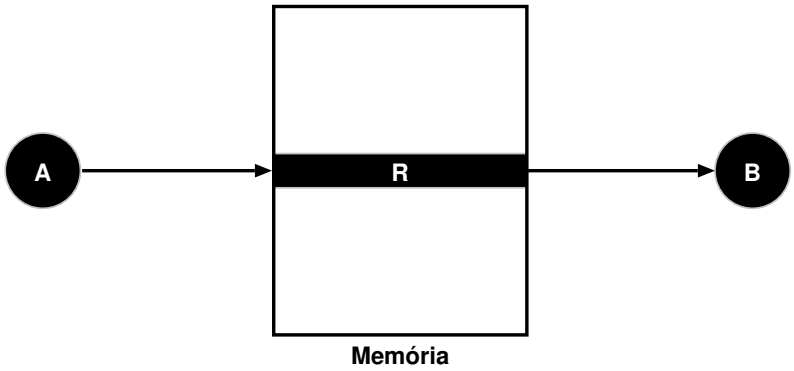
diferentes, sendo que no primeiro caso as mensagens podem ser trocadas por uma rede especializada que conecta todos os processadores, e no segundo caso por uma rede de computadores comum interligando os computadores.

4. (2.0) Descreva os estados de um processo e as ações levadas a cabo em cada uma das possíveis transições entre estes estados como descrito na figura a seguir:



Resp.: Um processo pode estar em três estados: executando, quando este está sendo executado pelo processador; pronto, quando este processo está esperando para ser executado no processador (pois algum outro processo está em execução no processador); e bloqueado, quando o processo não pode executar no processador até a ocorrência de algum evento externo. A Transição 1, do estado executando para o bloqueado, ocorre quando um processo em execução descobre que somente poderá continuar a executar após a ocorrência de um certo evento externo à sua execução. A Transição 2, do estado executando para o pronto, ocorre quando o escalonador determinou que o processo atualmente em execução já executou por muito tempo no processador. A Transição 3, do estado pronto para o executando, ocorre quando o escalonador determinou que é a vez deste processo, que estava esperando pelo uso do processador, de executar no processador por algum tempo. Finalmente, a Transição 4, do estado bloqueado para o pronto, significa que o evento externo pelo qual este processo estava esperando ocorreu, e com isso, o processo poderá agora ser escolhido pelo escalonador para ser futuramente executado pelo processador.

5. (2.0) Suponha que dois processos, A e B, se comunicam usando a palavra da memória R, sendo que R é usada para transferir dados do processo A para o processo B, como descrito pela figura a seguir:



(a) (0.75) O acesso irrestrito à palavra R pelos processos A e B gera condições de corrida. Descreva estas condições.

Resp.: As condições de corrida ocorrerão quando o processo A tentar escrever na palavra R quase ao mesmo tempo em que o processo B tentar ler um valor desta palavra. Temos dois casos: no primeiro, A pode escrever novamente na posição R antes de B ler o valor escrito anteriormente por A, o que fará com que este valor seja perdido; e no segundo, B tentará ler um valor da posição R antes que A coloque um novo valor nesta posição, o que fará com que B leia novamente o mesmo valor.

(b) (1.25) Explique quais são os semáforos necessários para a correta implementação da comunicação entre os processos A e B. Dê os valores iniciais desses semáforos para o caso em que a palavra R está inicialmente ocupada por um dado e preencha as lacunas nos códigos abaixo indicando os nomes dos semáforos apropriados.

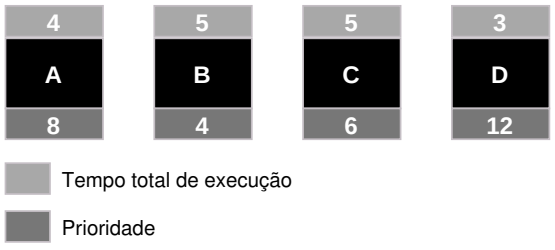
Resp.: Precisamos de dois semáforos binários, *dado_disponivel* e *dado_ja_lido*. O semáforo *dado_disponivel* irá controlar o acesso do processo B à palavra R, somente permitindo o acesso quando R possuir um novo valor, colocado por A, a ser lido por B. Como

inicialmente a palavra R está ocupada por um dado, o valor inicial do semáforo *dado_disponivel* será 1. O semáforo *dado_ja_lido* irá controlar o acesso do processo A à palavra R, somente permitindo o acesso após B ler o valor atual de R. Como R está inicialmente ocupada, o valor inicial do semáforo *dado_ja_lido* deverá ser 0. Para que a comunicação dos processos seja correta e sincronizada, o processo A deverá executar a operação **P** sobre *dado_ja_lido* antes de escrever um valor em R, e executar a operação **V** sobre *dado_disponivel* logo após colocar um novo valor em R. Por sua vez, o processo B deverá executar a operação **P** sobre *dado_disponivel* antes de ler um valor de R, e deverá executar a operação **V** sobre *dado_ja_lido* logo após ler um valor de R. A seguir damos o código obtido ao preencher as lacunas de acordo com o que foi descrito anteriormente.

```
void ProcessoA(void)
{
  int Valor;
  while (1) {
    Valor = GerarDado();
    P(dado_ja_lido);
    R = Valor;
    V(dado_disponivel);
  }
}
```

```
void ProcessoB(void)
{
  int Valor;
  while (1) {
    P(dado_disponivel);
    Valor = R;
    V(dado_ja_lido);
    UsarDado(Valor);
  }
}
```

6. (1.5) Suponha que os processos A, B, C e D estão atualmente em execução no sistema, como na figura a seguir, onde são dados, para cada processo, o seu tempo total de execução e a sua prioridade. Quanto tempo será necessário para que cada processo termine a sua execução, se:



(a) (0.75) Usarmos o algoritmo de escalonamento por *round robin*, com um quantum igual a uma unidade de tempo.

Resp.: Na execução usando o algoritmo por *round robin*, cada um dos processos executará em seqüência, um após o outro, até o término do quantum, ou da sua execução. Supondo que os processos estão na fila na ordem dada na figura, ou seja, na ordem A, B, C e D, teremos a seguinte seqüência de execução: A, B, C, D, A, B, C, D, A, B, C, D, A, B, C, B e C. Como cada quantum equilave a uma unidade de tempo, então A terminará a sua execução após 13 unidades de tempo, B terminará a sua execução após 16 unidades de tempo, C terminará a sua execução após 17 unidades de tempo, e D terminará a sua execução após 12 unidades de tempo.

(b) (0.75) Usarmos o algoritmo de escalonamento por prioridades, com redução de prioridades, supondo que as prioridades são sempre reduzidas de 1, dentro de uma unidade de tempo?

Resp.: Como vimos na Aula 6 que o processo de maior prioridade executa até que a sua prioridade seja menor do que a do processo com a segunda maior prioridade, podemos ter as seguintes ordens de execução (para cada processo damos, entre parênteses, a prioridade que este processo possui ao ser executado):

- D (12), D (11), D (10), A (8), A (7), A (6), C (6), C (5), A (5), B (4), C (4), C (3), B (3), B (2), C (2), B (1), B (0): neste caso, A terminará a sua execução em 9 unidades de tempo, B terminará a sua execução em 17 unidades de tempo, C terminará a sua execução em 15 unidades de tempo, e D terminará a sua execução em 3 unidades de tempo.
- D (12), D (11), D (10), A (8), A (7), A (6), C (6), C (5), A (5), C (4), B (4), B (3), C (3), C (2), B (2), B (1), B (0): neste caso, A terminará a sua execução em 9 unidades de tempo, B terminará a sua execução em 17 unidades de tempo, C terminará a sua execução em 14 unidades de tempo, e D terminará a sua execução em 3 unidades de tempo.