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 - Primeiro Semestre de 2008

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) Descreva como multiplexar (i) por tempo e (ii) por espaço um dado recurso do computador, destacando em quais casos cada um destes é aplicado.

   Resp.: Na multiplexação por tempo, cada processo executando algum programa usa exclusivamente o recurso por um determinado intervalo de tempo, ou seja, dois ou mais processos não usarão o recurso ao mesmo tempo. Já na multiplexação por espaço, o recurso é dividido em diversas partes, e cada processo usa uma (ou mais) destas partes, sendo que somente um processo usa exclusivamente a parte do recurso alocada a ele. Com isso, neste último caso, podemos ter dois ou mais processos usando concorrentemente um mesmo recurso, desde que eles usem partes diferentes deste recurso. Um uso da multiplexação por tempo ocorre no gerenciamento do processador do computador pelo sistema operacional, pois cada processo executa exclusivamente no processador por um dado intervalo de tempo. Já a multiplexação por espaço é usada no gerenciamento da memória pelo sistema operacional, pois a memória em geral é compartilhada pelos códigos e dados de alguns dos processos em execução, que não ocupam simultaneamente a mesma parte da memória.

2. (1.5) Discuta a relação existente entre os seguintes elementos do sistema operacional: processos, arquivos e diretórios e chamadas ao sistema operacional.

   Resp.: A relação que existe entre os processos e os arquivos e diretórios é a de que os dados usados pelos processos (como, por exemplo, os arquivos de configuração) são, em geral, lidos de arquivos. Além disso, as saídas com os resultados das computações dos processos também são, em geral, salvas em arquivos. Também, o próprio código do processo, assim como o de outros processos, é armazenado em um arquivo. Os processos podem usar os diretórios para organizar os arquivos de dados que eles usam, ou para organizar as saídas geradas por sua computação. Agora, para poder criar os arquivos e os diretórios e, para poder e ler e salvar dados nestes arquivos, os processos devem usar as chamadas ao sistema operacional que tratam do gerenciamento do sistema de ar-

quivos. Um processo também pode criar um novo processo através de uma chamada ao sistema operacional que lê o arquivo com o código do processo e que depois cria este processo e o coloca no estado pronto. Um processo não acessa diretamente os arquivos e os diretórios por dois motivos. Primeiro, porque o processo desconhece como os arquivos e os diretórios são salvos no disco (isso dependerá do tipo do sistema de arquivos). Segundo, porque o acesso direto à versão abstrata do disco, disponibilizada pelo sistema operacional, é somente permitido aos processos que compõem o sistema operacional. Em relação a executar diretamente os processos, além do processo desconhecer como os arquivos são salvos no disco e como o código é armazenado em um arquivo, tem o fato de que somente o sistema operacional poder criar novos processos.

3. (1.5) O grande problema do sistema monolítico é a falta de estruturação do seu núcleo. Descreva sucintamente como os modelos baseados em anéis e em micronúcleo estruturam o núcleo do sistema operacional.

   Resp.: Nos modelos baseados em anéis, assim como ocorre nos modelos baseados em camadas, o núcleo do sistema operacional é dividido em uma hierarquia em camadas, chamadas de anéis, sendo que cada uma delas trata do gerenciamento de alguma parte do hardware. Um anel oferece, aos anéis superiores a ele na hierarquia, uma versão abstrata e mais fácil de usar da parte do hardware que ele gerencia. Com isso, um anel fornece uma interface de acesso que independe da parte do hardware que ele gerencia. Além disso, cada anel somente pode acessar a versão abstrata do hardware fornecida pelos anéis inferiores, não podendo acessar diretamente a parte do hardware gerenciada por um destes anéis. Já quando usamos um modelo baseado em micronúcleo, todas as funções do sistema são implementadas no modo usuário por processos servidores. Estes processos gerenciam os recursos do hardware e implementam os diversos serviços oferecidos pelas chamadas ao sistema operacional nos outros modelos. O micronúcleo, que executa no modo supervisor, somente trata da troca de mensagens entre os diversos processos (dos usuários ou do sistema) executando no modo usuário, e do envio de comandos às controladoras dos dispositivos físicos. Estes comandos são passados ao micronúcleo através de mensagens especiais

enviadas pelos processos servidores que gerenciam estes dispositivos.

4. (1.5) A multiprogramação permite que vários processos executem realmente em parelelo no computador, isto é, os processos são todos executados simultaneamente? Justifique a sua resposta.

   Resp.: Depende. A multiprogramação permite que vários processos compartilhem os processadores do computador, fazendo com que os processos executem alternadamente, isto é, *concorram* por estes processadores, por um certo intervalo de tempo. Se o número de processadores for maior ou igual do que o de processos no estado pronto, então teremos o que chamamos de um paralelismo real. Neste caso, todos os processos prontos estarão executando paralelamente no computador, um em cada processador diferente. Porém, se o número de processadores for menor, então o número de processos paralelos não será igual ao número processos prontos, e sim, ao de processadores, e os processos alternarão as suas execuções nestes processadores. Neste último caso o paralelismo é aparente (pseudoparalelismo), pois temos a impressão de que todos os processos estão em execução paralela, por causa do pequeno tempo em que um processo executa em um dos processadores antes de cedê-lo a um outro processo.

5. (2.0) Suponha que temos uma pilha com n posições, compartilhada por um conjunto de processos. Esta pilha possui duas operações: *empilha*, para inserir um elemento no topo da pilha, e *desempilha*, para remover o elemento do topo da pilha. Como estas operações devem ser implementadas, usando semáforos, para garantir a exclusão mútua ao acessar a pilha e a correta sincronização dos processos? Justifique a sua resposta.

   Resp.: Vamos precisar de dois semáforos de contagem, *cheio* e *vazio*, e de um semáforo binário, *acesso*. O semáforo *cheio* irá bloquear um processo que deseja inserir um elemento no topo da pilha quando ela estiver cheia, ou seja, se possuir n elementos. Já o semáforo *vazio* irá bloquear um processo que deseja remover o elemento do topo da pilha caso ela esteja vazia, isto é, sem nenhum elemento. Finalmente, o semáforo *acesso* será usado para garantir o acesso exclusivo à pilha.

Como não sabemos quantos elementos temos inicialmente na pilha, vamos supor que a pilha possui a elementos, $0 \leq a \leq n$. Então, os valores iniciais dos semáforos *cheio* e *vazio* serão de, respectivamente, $n - a$ e a. Já o valor inicial do semáforo *acesso* será de 1, pois inicialmente nenhum processo está usando a pilha. A seguir damos os códigos das funções *Empilha*(*elemento*) e *Desempilha*(*elemento*), sendo que mostramos as posições corretas onde as operações sobre os semáforos devem ser usadas. A função *Empilha*(*elemento*) implementa a operação *empilha*, e insere o elemento dado por *elemento* no topo da pilha. Já a função *Desempilha*(*elemento*) implementa a operação *desempilha*, que remove o elemento do topo da pilha, colocando-o em *elemento*. Finalmente, a função *InsereTopo*(*elemento*) insere *elemento* no topo da pilha e a função *RemoveTopo*() remove e retorna o elemento no topo da pilha.

```
void Empilha(elemento)
{
  P(cheio);
  P(acesso);
  InsereTopo(elemento);
  V(acesso);
  V(vazio);
}
```

```
void Desempilha(elemento)
{
  P(vazio);
  P(acesso);
  elemento = RemoveTopo();
  V(acesso);
  V(cheio);
}
```

6. (2.0) Suponha que três processos A, B, e C acabaram de ser iniciados e são os únicos processos em execução no sistema. Suponha ainda que A executa por 2 quanta, B por 1 quantum e C por 3 quanta. Se usarmos o algoritmo de escalonamento por *round-robin*, qual seria a seqüência de execução dos processos no processador, se A foi o primeiro processo escolhido pelo escalonador e C foi o segundo?

Resp.: Como A executa por 2 quanta, B por 1 quantum e C por 3 quanta, então A, B e C serão escolhidos para executar pelo escalonador por, respectivamente, 2, 1 e 3 vezes. Agora, como as duas primeiras escolhas do escalonador foram, respectivamente, A e C, então a ordem de execução será a seguinte: A, C, B, A, C e C.