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
AP3 - Segundo Semestre de 2007

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. (1.0) Qual é a diferença essencial que existe ao executarmos um sistema operacional sobre uma máquina virtual em relação a executarmos sobre o hardware do computador? Justifique a sua resposta.

   Resp.: Quando executarmos o sistema operacional sobre o hardware do computador, ele será o responsável por gerenciar os diversos componentes deste hardware. Logo, quando o sistema operacional executar uma operação de E/S sobre um dispositivo físico, ele interagirá diretamente com a controladora deste dispositivo. Além disso, o sistema irá usar a memória do hardware, e será o responsável pelo compartilhamento do processador do hardware entre os processos do sistema. Agora, se o sistema operacional executar sobre uma máquina virtual, todos os componentes da máquina virtual serão cópias idênticas, porém virtuais, dos componentes reais do hardware. Neste caso, quando uma operação de E/S for executada, o sistema irá agora interagir com a controladora virtual do dispositivo, e não com a controladora real. Do mesmo modo, o sistema acessará a memória da máquina virtual e gerenciará o processador virtual dela. Note que o acesso direto aos dispositivos do hardware e o seu compartilhamento entre as diversas máquinas virtuais será feito pelo monitor de máquina virtual, e não pelo sistema operacional executando na máquina virtual.

2. (2.0) Suponha que dois processos A e B executam cooperativamente uma tarefa do seguinte modo: A coloca itens em um vetor R com n posições, e B retira itens de R. A ordem de processamento dos itens colocados por A não é importante, mas todos os itens devem sempre ser processados por B. O vetor R está inicialmente vazio. Responda:

   (a) (1.0) Como podemos usar os semáforos para garantir que os processos executem corretamente a tarefa cooperativa, se supusermos que a exclusão mútua ao acessar o vetor R já está garantida?

   Resp.: Como estamos supondo que a exclusão mútua ao acessar o vetor R já está garantida, então precisaremos somente garantir a correta sincronização dos processos A e B. Neste caso, precisaremos bloquear A sempre que não existir espaço para colocar mais um item em R, e bloquear B sempre que R estiver vazio. Para

fazer isso, precisaremos criar dois semáforos de contagem, *cheio* e *vazio*, que contarão, respectivamente, o número de entradas de R disponíveis e com itens. Como R está inicialmente vazio, então os semáforos *cheio* e *vazio* deverão ser inicializados com, respectivamente, os valores n e 0. A seguir mostramos os fragmentos de código dos processos A e B, sendo que em cada fragmento mostramos a ordem correta em que as operações **P** e **V** devem ser executadas sobre os semáforos *cheio* e *vazio*. Como no enunciado não foi especificado quantos itens serão colocados pelo processo A em R, então vamos supor que A e B executam para sempre e, com isso, os fragmentos de código dados a seguir possuem laços infinitos. Também poderíamos supor que A coloca um número fixo de itens em R antes de terminar a sua execução, e que B retira este mesmo número de itens antes de terminar a sua execução.

```
void ProcessoA(void)
{
  while (1) {
    ⋮
    P(cheio);
    \\ Insere um item em R.
    ⋮
    V(vazio);
    ⋮
  }
}
```

```
void ProcessoB(void)
{
  while (1) {
    ⋮
    P(vazio);
    \\ Remove um item de R.
    ⋮
    V(cheio);
    ⋮
  }
}
```

(b) (1.0) Suponha que o sistema operacional usa o escalonamento por *round-robin* e que, em cada quantum, A coloca dois itens em R e B retira um item de R. Por quantos quanta B deverá executar se supusermos que A executa por 5 quanta, e que $n > 10$? Justifique a sua resposta.

Resp.: Como o vetor R está inicialmente vazio, se o processo B executar antes do processo A, ele será bloqueado até A executar e, com isso, não executará por um quantum completo. Supondo

então que A começa a execução antes de B, então os processos executarão alternadamente até que A termine a sua execução. Note que o processo A nunca será bloqueado, pois ele coloca, durante os 5 quanta de execução, 10 itens em R, e o tamanho de R é maior do que 10. O processo B também não será bloqueado (a não ser inicialmente, se escolhido pela primeira vez antes de A), pois A sempre coloca dois itens em cada quantum e B retira somente um item em cada quantum. Agora, como B somente retira um item em cada quantum, o número de quanta que ele executará será igual ao número de itens colocados por A em R, isto é, B executará por 10 quanta.

3. (2.0) Suponha que temos n processos P_i, $1 \leq i \leq n$, executando no computador, e que também temos n recursos R_i, $1 \leq i \leq n$, no computador. Suponha ainda que o processo P_i precisa dos recursos R_i e R_{i+1} se $i < n$ e R_1 e R_n se $i = n$, e que P_i obtém um recurso de cada vez, sem liberar um outro recurso que tenha obtido anteriormente. O que ocorerrá se cada processo P_i obtiver o recurso R_i, e se:

   (a) (1.0) todos os recursos forem não-preemptivos?

   Resp.: Se todos os recursos forem não-preemptivos, teremos um impasse, pois cada processo P_i possui o recurso R_i, e precisa do recurso R_{i+1} alocado a P_{i+1} se $i < n$ ou R_1 alocado a P_1 se $i = n$. Somente quando o processo P_{i+1} se $i < n$ ou P_1 se $i = n$ liberar o seu recurso é que P_i poderá ser desbloqueado, mas isso nunca ocorrerá, pois o outro processo também está bloqueado esperando por um recurso.

   (b) (1.0) pelo menos um dos recursos for preemptivo?

   Resp.: Como vimos na resposta da questão anterior, se todos os recursos forem não-preemptivos, teremos um impasse. Porém, se pelo menos um recurso, digamos R_i, for preemptivo, poderemos tirá-lo do processo P_i que o possui, e não teremos mais um impasse, como veremos a seguir. Neste caso, o processo P_{i-1} se $i > 1$ ou P_n se $i = 1$, que estava bloqueado esperando por R_i poderá ser

desbloqueado. Depois de este processo executar e liberar os seus recursos, os dois processos que estavam esperando pelos recursos que ele possuía poderão agora ser desbloqueados. Após estes dois processos executarem e liberarem os recursos, mais dois processos, que dependiam dos recursos que acabaram de ser liberados, poderão agora executar, e assim sucessivamente. Eventualmente, os recursos R_i e R_{i+1} se $i < n$ ou R_1 se $i = n$ que P_i precisava serão liberados, e P_i poderá continuar a sua execução.

4. (3.0) Suponha que um computador tem uma memória virtual de 4GB e uma memória física de 1GB. Suponha ainda que o sistema operacional usa uma tabela de páginas com somente um nível. Responda:

   (a) (1.0) Se o computador possuir 2^{14} molduras de página, quantas páginas virtuais teremos? Como será dividido o endereço virtual neste caso?

   Resp.:-Como o computador tem 1GB de memória, isto é, uma memória com 2^{30} bytes, então cada endereço físico terá 30 bits. Além disso, como temos 2^{14} molduras de página que ocupam 14 bits neste endereço físico, então o tamanho de cada moldura será de 2^{16} bytes, pois os 16 bits restantes do endereço físico definirão o deslocamento dentro da moldura. Agora, como o computador tem 4GB de memória virtual, isto é, 2^{32} bytes, então cada endereço virtual terá 32 bits. Como o tamanho da página virtual é igual ao tamanho da moldura de página, 16 bits deste endereço serão usados para definir o deslocamento dentro da página virtual e os 16 bits restantes serão usados para definir esta página. Logo, teremos $2^{16} = 65536$ páginas virtuais.
   -Pelo que vimos anteriormente na resposta desta questão, o endereço virtual, com 32 bits, será dividido do seguinte modo: os seus 16 bits superiores indicarão a página virtual, e os seus 16 bits inferiores o deslocamento dentro desta página.

   (b) (1.0) Quantas entradas deveriam existir na TLB para que a taxa de acerto fosse de 5%, supondo que todas as páginas virtuais possuem igual probabilidade de serem acessadas?

Resp.: Na questão anterior, vimos que temos 2^{16} páginas virtuais. Logo, teremos 2^{16} entradas na tabela de páginas. Agora, como queremos garantir que a taxa de acerto seja de 5%, e como todas as páginas possuem igual probabilidade de serem acessadas, então a TLB deverá ter um número de entradas não menor do que $5 \times 2^{16}/100$. Como o valor $5 \times 2^{16}/100 = 2^{15}/10 = 3276,8$ não é exato, então a TLB deverá ter 3277 entradas, para garantir que a taxa de acerto seja de aproximadamente 5%.

(c) (1.0) Suponha que o sistema alocou três molduras de página para um processo, e que duas molduras não estão vazias, possuindo as páginas 1 e 7, sendo que a 7 foi a última a ser acessada. Suponha ainda que o processo acessou depois as páginas 0, 2, 1, 2, 7 e 4, nesta ordem. Se usarmos o algoritmo LRU, uma página será substituída quando o processo acessar novamente a página 2?

Resp.: Se o sistema operacional usar o algoritmo LRU, as páginas serão escolhidas como na tabela dada a seguir. Em cada uma das linhas da tabela, as páginas estão ordenadas, em ordem crescente, de acordo com tempo em que foram acessadas pela última vez. Com isso, a primeira página (em negrito) será a escolhida ao ocorrer uma falha de página. A primeira linha da tabela mostra o estado das molduras antes de acessarmos as páginas. As outras linhas da tabela mostram o estado das molduras após acessarmos, em ordem, as páginas dadas no enunciado. Como, pela última linha da tabela, vemos que a página 2 ainda está na memória, então nenhuma página será substituída ao acessarmos novamente a página 2.

Página	Molduras			Falha de página?
-	**1**	7		-
0	**1**	7	0	Sim
2	**7**	0	2	Sim
1	**0**	2	1	Sim
2	**0**	1	2	Não
7	**1**	2	7	Sim
4	**2**	7	4	Sim

5. (2.0) Suponha que o computador possui um disco com 32MB. Suponha ainda que um mapa de bits, ocupando exatamente 8KB da memória do computador, é usado para gerenciar os blocos livres do disco. Responda:

   (a) (1.0) Quantos blocos existem no disco, e qual é o tamanho de cada um deles?

   Resp.: Como o mapa de bits ocupa 8KB da memória principal, isto é, 8192 bytes, o mapa terá 65536 bits. Logo, o disco tem 65536 blocos. Agora, como o disco tem 32MB de tamanho, isto é, 2^{25} bytes (ou 33554432 bytes), e como temos 65536 blocos, ou seja, 2^{16} blocos, o tamanho de cada bloco do disco será de $2^{25}/2^{16} = 2^9$ bytes (ou 33554432/65536 = 512 bytes).

   (b) (1.0) Suponha que usamos a alocação contígua, e que dois Arquivos A e B foram armazenados no disco, com tamanhos de, respectivamente, 20MB e 10MB. Se A foi armazenado no início do disco, onde B poderia ser armazenado para que pudéssemos criar um arquivo com 2MB sem usarmos a compactação do disco?

   Resp.: O arquivo que desejamos armazenar terá 4096 blocos de 512 bytes, pois o tamanho deste arquivo é de 2MB, ou seja, 2097152 bytes. Após armazenarmos os arquivos A e B no disco, teremos somente 2MB de espaço disponível nele, que é exatamente o tamanho do arquivo que desejamos armazenar. Agora, como estes 4096 blocos devem ser consecutivos no disco, então poderemos

ou armazenar B imediatamente após A no disco, ou poderemos armazenar B nos últimos blocos do disco. No primeiro caso, o arquivo com 2MB seria armazenado nos últimos 4096 blocos consecutivos do disco e, no último caso, ele seria armazenado nos 4096 blocos consecutivos entre o bloco final do arquivo A e o bloco inicial do arquivo B.