EA 876 – Introdução a Software de Sistema

Prova 2 – 03/07/2009 – Turmas A e B

RA:_____ Nome: _____ Assinatura:_____

Questão 1 (3,0 pts.)

O algoritmo de troca de páginas LRU (Least Recently Used) procura eliminar da memória as páginas que não são referenciadas há mais tempo. Uma das maneiras de se efetuar este controle de idade da página é através de um contador em hardware que é incrementado cada vez que uma instrução é executada. Quando uma página é acessada, o valor corrente do contador é armazenado em um campo da tabela de páginas e assim o algoritmo tem dados para ordenar as páginas e decidir qual retirar da memória real. Este algoritmo requer o uso de hardware especial para ser implementado, o que o torna pouco prático. Responda às seguintes perguntas.

(a:1,0) Por que o contador deve ser implementado em hardware?

Resposta: se o contador é incrementado A CADA INSTRUÇÃO, então só mesmo implementando em hardware para que o tempo de incremento não tenha impacto no desempenho do processo. Se fosse implementado em software e incrementado a cada instrução, muito mais tempo seria gasto com esta atualização de contador que com trabalho útil.

(b:1,0) Como o LRU pode ser simulado em software?

Resposta: o campo da tabela de páginas é atualizado por uma rotina do S.O. quando há interrupção do relógio (troca de contexto). Esta atualização consiste apenas na soma do valor atual do campo com o bit R de referência de página. Para evitar que referências antigas a uma página tenham efeito acentuado no futuro, costuma-se dividir por 2 (deslocar os bits para a direita) o valor que está presente no campo da tabela de páginas antes de se somar o valor do bit de referência R.

(c:1,0) Por que a simulação em software de LRU é viável na prática?

Resposta: porque a atualização só ocorre a cada troca de contexto e não a cada instrução. Isto significa que a frequência de atualização do(s) contador(es) é muito menor, viabilizando a operação por software e eliminando a necessidade de hardware especial para o contador.

Questão 2 (3,0 pts.)

(a:1,0) Explique porque a concorrência de processos (competição pelo uso de recursos comuns) pode ser nociva.

Resposta: concorrência é a competição entre 2 ou mais processos (executando em uma ou mais CPUs) pelo uso de um certo recurso compartilhado (posição de memória, tabela, arquivo, porta de E/S, etc.). Se esta competição não for administrada, resultados inconsistentes podem surgir, dependendo da ordem em que os processos tiveram acesso ao recurso compartilhado (condição de corrida ou race condition). Existem várias maneiras de adminstrar esta competição, as quais implicam na garantia da mútua exclusão, isto é, quando um processo estiver tendo acesso ao recurso compartilhado, os demais devem ficar proibidos de fazer o mesmo. Os mecanismos de garantia de mútua exclusão se dividem naqueles

de espera ocupada (variáveis lock, instruções TSL, etc.) e espera bloqueada (sleep/wakeup, semáforos, etc.).

(b:1,0) Esta concorrência pode ser administrada também com operações sobre semáforos. Explique quais são estas operações e como elas funcionam.
Resposta: as operações sobre semáfaros são duas: DOWN(semáforo) e UP(semáforo), onde semáforo é uma variável inteira que acumula a quantidade de sinais de wake-up (despertar) relacionados a um determinado recurso compartilhado. Inicialmente, um semáfor que protege um recurso compartilhado que não está sendo utilizado tem um valor inicial de pelo menos 1. Quando um processo deseja usar este recurso, ele primeiro precisa executar uma operação DOWN(semáforo) para ver se tem ou não permissão para seguir em frente. Se o valor do semáforo for maior que zero, DOWN irá decrementar este valor e o processo terá permissão para seguir em frente e usar o recurso compartilhado. Se o semáforo for nulo, o processo é colocado para "dormir" (é bloqueado) antes que possa completar a operação DOWN (ele não chega a decrementar o semáforo). Posteriormente este processo será desbloqueado pelo sistema operacional (por meio de uma operação UP) e poderá completar o DOWN e usar o recurso compartilhado. Depois que um processo já usou o recurso compartilhado ele deve executar a operação UP(semáforo), a qual incrementa o valor do semáforo e desbloqueia um dos processos que porventura estejam bloqueados neste semáforo, permitindo que ele complete a operação DOWN que o fez bloquear. Ou seja, se houver algum processo bloqueado no semáforo, após um UP o valor do semáforo permanecerá o mesmo, pois ele será incrementado e decrementado logo em seguida pela operação DOWN que será executada pelo processo que for desbloqueado.

(c:1,0) Mostre como esta concorrência pode ser administrada com uma instrução tipo TSL, dando um exemplo de uso desta instrução.
Resposta: a instrução TSL (Test and Set Lock) realiza de forma atômica (indivisível) duas operações sobre uma variável de travamento que controla o uso de um recurso compartilhado. Estas operações são copiar o valor atual da variável para um registrador da CPU e setar (guardar o valor 1) nesta variável logo em seguida. Antes de usar o recurso, um processo precisa ter certeza de que conseguiu travar (setar) esta variável, ganhando acesso exclusivo ao recurso. Após usar o recurso, o processo só precisa destravar (ressetar) esta variável. Como as operações de TSL são indivisíveis, a mútua exclusão baseada em instrução TSL tem sua eficácia garantida. Os exemplos abaixo de código para entrada e para saída da região crítica (parte do processo que usa o recurso compartilhado) ilustram o uso desta instrução. Enquanto o processo que chama ENTRAR não conclui que foi ele que conseguiu setar a TRAVA, ele permanece em um loop (espera ocupada) e não entra na região crítica. O valor de REGISTRADOR comparado com 0 é aquele que estava guardado em TRAVA antes de TSL ser executada. Se TRAVA já for igual a 1, então TSL simplesmente escreve outro 1 por cima e nada se altera. Se ela for 0, então TSL a coloca em 1 e logo depois CMP descobre que o valor antigo era 0, o que permite concluir que a TRAVA estava aberta e agora está fechada (setada) pelo próprio processo. Logo ele pode executar RET, voltar ao seu código original e seguir em frente entrando na região crítica.

```
ENTRAR:   TSL   REGISTRADOR, TRAVA
          CMP   REGISTRADOR, #0
          JNE   ENTRAR
          RET
```

```
SAIR:   MOV  TRAVA, #0
        RET
```

Questão 3 (1,0 pt.)

Em sistemas de tempo real não basta que o sistema computacional responda de forma correta. É preciso também que ele responda dentro de um limite de tempo pré-determinado para que o resultado seja útil. Neste contexto é muito importante se conhecer o tempo máximo que determinadas operações irão gastar, para se saber se o sistema atende ou não as restrições de prazo que o seu uso impõe. Supondo um contexto com este, calcule o tempo máximo necessário para se carregar por completo do disco um arquivo de 1MB, considerando que o tempo de posicionamento da cabeça de leitura/gravação sobre uma trilha é de até 32 ms, que a velocidade de rotação mínima do disco é de 128 rps (rotações por segundo), que os cilindros só contêm uma trilha cada um, que as trilhas são formadas por 64 setores de 1KB e que os blocos de dados são de 1KB. Não é permitido uso de calculadora e você deve realizar os cálculos em potência de dois.

Resposta: T_max = num_leituras * (T_posicionamento_max + T_atraso_rotacional_max + T_leitura_max)

num_leituras = num_setores (como só há uma trilha, não há possibilidade de leituras simultâneas em paralelo)

num_setores = num_blocos (ambos são de 1 KB)

num_blocos = tamanho_arquivo / tamanho_bloco = $2^{2°}$ B / $2^{1°}$ B = $2^{1°}$ (como só há uma trilha, não há possibilidade de leituras simultâneas em paralelo)

T_posicionamento_max = 32 ms = 2^5 ms (do enunciado)

T_atraso_rotacional_max = T_uma_rotação_min = 1/128 = 2^{-7} s (pior caso: toda vez que a cabeça de leitura chega em uma trilha, o início do setor desejado acabou de passar)

T_leitura_max = T_leitura_um_setor_em_rotação_mínima = 2^{-7} s / 64 = 2^{-7} s / 2^6 s= 2^{-13} s

T_max = $2^{1°}$ * (2^5 ms + 2^{-7} s + 2^{-13} s)

Questão 4 (3,0 pts.)

(a:1,0) Qual é o tamanho máximo de um arquivo indexado por um i-node no Unix, sabendo que o disco tem 2^{32} blocos de 4 KBytes? Considere que, além dos atributos do arquivo, cada i-node pode guardar o endereço de dez blocos de dados e de três blocos indiretos: um simples, um duplo e um triplo. Justifique sua resposta. Não é permitido uso de calculadora e você deve realizar os cálculos em potência de dois.

Resposta: 1 bloco tem 2^{12} bytes = 2^{15} bits;

cada bloco precisa de 32=2^5 bits para ser endereçado;

portanto cada bloco pode conter 2^{15} / 2^5 = 2^{10} endereços de blocos;

cada i-node tem 10 endereços diretos para blocos e 3 indiretos, sendo 1 indireto simples, 1 indireto duplo e 1 indireto triplo;

portanto temos $(10 + 2^{10} + 2^{10} \times 2^{10} + 2^{10} \times 2^{10} \times 2^{10}$ blocos) x 4 KB/bloco = $(10 + 2^{10} + 2^{20} + 2^{30}) \times 4*2^{10}$ B = $40*2^{10} + 2^{22} + 2^{32} + 2^{42}$ bytes (> 4 TB)

(b:1,0) Apresente e discuta uma vantagem ou uma desvantagem de se gravar os dados de arquivos pequenos no próprio inode do arquivo.

Resposta: vantagens:

- o acesso aos dados será agilizado, principalmente no caso de arquivos pequenos;
- haverá economia de disco, pois o desperdício de blocos de dados que guardam poucos dados será menor;

desvantagens:

- passarão a existir duas maneiras de se procurar um bloco de dados no sistema: dados no bloco do inode e dados em blocos diretos ou indiretos (simples, duplo ou triplo), o que exigirá um esforço estra para determinar o método de busca adequado.

- se um arquivo pequeno que está gravado no i-node cresce e não cabe mais lá, será preciso copiar seu conteúdo para um bloco de dados para liberar o i-node para armazenar endereços de blocos de dados, o que exigirá um esforço de cópia extra.

(c:1,0) Após uma varredura no disco, um programa de verificação de consistência montou o seguinte mapa de bits, onde 0 representa "não" e 1 representa "sim".

Nº do bloco =>	0	1	2	3	4	5	6	7	8	9
Bloco usado =>	1	1	0	1	0	0	1	1	0	1
Bloco livre =>	0	1	0	1	1	1	1	0	0	0

Identifique e descreva todos os problemas encontrados, discuta suas consequências e proponha uma solução para cada um.

Resposta: os problemas são:

1. Blocos 1, 3 e 6 estão nas listas de usados e livres simultaneamente. Como estão na lista de livres poderão vir a ser usados por outro arquivo e haverá perda de dados. A solução é simples: basta retirá-los da lista de blocos livres.
2. Blocos 2 e 8 não aparecem como usados nem como livres. Eles estão perdidos e nunca iriam ser usados pelo sistema de arquivos. A solução consiste em colocá-los na lista de blocos livres.