EXERCÍCIOS DO LIVRO

1) Quais as duas principais funções de um sistema operacional?

Gerenciar recursos de hardware e fornecer abstrações limpas destes ao usuário.

3) Qual a diferença entre sistemas de compartilhamento de tempo e de multiprogramação?

Em um sistema de tempo compartilhado, múltiplos usuários podem acessar e executar seus programas simultaneamente, usando seus próprios terminais. Sistemas multiprogramáveis permitem ao usuário executar vários programas simultaneamente. Todos os sistemas de tempo compartilhado são multiprogramação, mas o contrário nem sempre pode ser dito.

4) Para usar a memória de cache, a memória principal é dividida em linhas de cache, em geral de 32 a 64 bytes de comprimento. Uma linha inteira é capturada em cache de uma só vez. Qual é a vantagem de fazer isso com uma linha inteira em vez de um único byte ou palavra de cada vez?

Porque as chances da cache acertar, pelo menos na segunda tentativa, aumentam. No mais, hardware moderno pode fazer uma transferência em blocos de 32 ou 64 bytes para uma linha de cache mais rápido do que se lesse as palavras destes mesmos dados individualmente.

6) Instruções relacionadas ao acesso a dispositivos de E/S são tipicamente instruções privilegiadas, isto é, podem ser executadas em modo núcleo, mas não em modo usuário. Dê uma razão do por que elas serem privilegiadas.

Devido ao fato destas permissões serem decididas pelo administrador do sistema. Assim, estas políticas de acesso devem ser reforçadas para que programas em nível de usuário não interfiram com elas.

9) Há várias metas de projeto (recursos, robustez) na construção de um SO. Dê um exemplo de duas metas de projeto que contradizem uma à outra.

Processos em tempo real e tempo de execução justa. O primeiro exige que os recursos sejam alocados baseando-se no tempo no qual diferentes processos precisem completar suas execuções. A segunda baseia-se em um tempo de execução igual para todos os processos. Assim, um processo em tempo real poderia ter uma quantidade desproporcional dos recursos necessários a sua execução.

10) Qual a diferença entre modo usuário e núcleo? Explique como ter dois modos distintos ajuda no projeto de um SO.

O modo núcleo dá total acesso às instruções do S.O. e aos recursos de hardware do computador. O modo usuário é um conjunto limitado de instruções com um acesso limitado aos recursos de software. Ter dois modos ajuda a manter a segurança, impedindo o usuário de ter acesso a instruções ou recursos críticos.

12) Quais das instruções a seguir devem ser deixadas somente em modo núcleo: desabilitar todas as interrupções; ler o relógio da hora do dia; configurar o relógio da hora do dia; mudar o mapa de memória.

Somente ler o relógio da hora do dia deve ser permitido em modo usuário. As demais devem rodar estritamente em modo núcleo.

15) Considere um sistema de computador que tem uma memória de cache, memória RAM e disco, e um SO que usa memória virtual. É necessário 1ns para acessar uma palavra da cache, 10ns para acessar uma palavra da RAM e 10ms para acessar uma palavra do disco. Se o índice de acerto da cache é de 95% e o índice de acerto da RAM após um erro da cache é de 99%, qual é o tempo médio para acessar uma palavra?

O tempo médio é a soma do tempo de acesso de cada um dos dispositivos multiplicados pela chance de acerto e, caso seja de algum dispositivo que pressuponha um erro anterior, deve-se multiplicar por essa margem de erro. Assim, o tempo médio é igual a:

$$T = (0.95 * 1) + (0.05 * 0.99 * 10) + (0.05 * 0.01 * 10^7) \Rightarrow T = 50001445\,ns.$$

16) Na leitura de disco, quando ocorre uma interrupção, quem chamou a leitura deverá ser bloqueado, pois ainda não há dados disponíveis para ele. E quanto a escrever para o disco? Quem chamou a escrita precisa ser bloqueado esperando o término da transferência de disco?

Talvez seja necessário bloquear, talvez não. Se quem chamou retomar o controle e imediatamente sobrescrever os dados, quando a escrita finalmente ocorrer, os dados errados serão escritos. Todavia, se o driver primeiro copiar os dados para um buffer privado antes de retorná-los, então quem chamou a leitura de disco poderá continuar imediatamente.

18) Por que a tabela de processos é necessária em um sistema de compartilhamento de tempo? Ela também é necessária em um SO de computador pessoal monousuário?

Ela é necessária para guardar o estado de um processo que está atualmente suspenso, seja em estado pronto ou bloqueado.

19) Há algum motivo para que se monte um sistema de arquivos em um diretório não-vazio? Se sim, por quê?

Montar um sistema de arquivos em um diretório faz com que quaisquer arquivos neste se tornem inacessíveis. No entanto, um administrador de sistema pode querer copiar alguns dos arquivos mais importantes normalmente localizados no diretório montado de forma com que eles possam ser encontrados em seu caminho natural em um caso de emergência, quando o dispositivo montado estiver sendo reparado.

22) A chamada *count* = *write(fd, buffer, nbytes);* pode retornar qualquer valor em *count* fora *nbytes*? Se sim, por quê?

Se a chamada der erro, ela retorna -1. Se encerrar normalmente, retornará *nbytes*.

25) Qual a diferença entre um arquivo especial de bloco e um de caractere?

Arquivos especiais em bloco consistem de blocos enumerados, e cada um destes pode ser lido ou escrito de forma independente dos outros. Isso não é possível em arquivos especiais de caractere.

26) É fundamental que uma rotina de sistema tenha o mesmo nome que sua equivalente rotina de biblioteca? Se não, qual a mais importante?

Não, pois rotinas de sistema, na verdade, nem mesmo possuem nomes; eles só existem para fins documentacionais. No sistema, rotinas de sistemas são representadas por números. Já o nome das rotinas de biblioteca são muito importantes.

27) SOs modernos desacoplam o espaço de endereçamento do processo da memória física da máquina. Liste duas vantagens desse projeto.

Isto permite que um programa seja carregado em diferentes partes da memória da máquina e em diferentes execuções. Além disto, permite que o tamanho do programa exceda o da memória da máquina.

31) Explique como a separação da política e do mecanismo ajuda na construção de SOs baseados em micronúcleos.

Com esta separação, os designers de sistema podem implementar apenas um pequeno número de instruções primitivas no núcleo. Estas instruções são simples, porque independem de alguma política específica. Então estas instruções primitivas podem ser combinadas para implementar mecanismos e políticas mais complexas em nível de usuário.

PRIMEIRO QUESTIONÁRIO

1) Por que threads são úteis no modelo de processo?

Por questões de desempenho, uma vez que estas são mais simples de criar ou destruir do que um processo, além de compartilharem os dados e o mesmo espaço de endereçamento com o processo que as criou.

2) Quais as maiores vantagens e desvantagens de implementar threads no espaço do usuário? E quanto à implementação no núcleo?

A maior vantagem é que, se implementadas no espaço do usuário, pode-se usar *threads* mesmo onde não se as suporte, além de não precisar desviar o controle para o núcleo, e de terem escalonamento mais rápido. A maior desvantagem é que, se ocorrer um bloqueio de thread ou uma falta de página, pode-se parar todas as threads em execução ou mesmo o próprio processo, além de não possuírem controle do relógio, devendo ceder voluntariamente suas vezes de execução.

No modo núcleo, a maior vantagem é que, se ocorrer uma falta de página, o núcleo verifica facilmente se o processo tem threads para executar, e as executa enquanto aguarda a página requisitada. A maior desvantagem é a sobrecarga elevada, seja para operações frequentes de criação e destruição de threads, seja porque chamadas bloqueantes ao thread são chamadas ao sistema.

3) Um thread pode sofrer preempção por uma interrupção de relógio? Em caso afirmativo, sob que circunstâncias? Do contrário, por que não?

Em modo usuário, as threads não podem sofrer preempção. Já as implementadas em modo núcleo sofrem, individualmente. Neste último caso, se um thread operar por tempo demais, o relógio pode mesmo interrompê-lo, ou ainda a seu processo.

4) Quando uma interrupção ou uma chamada de sistema transfere o controle para o sistema operacional, geralmente é usada uma área da pilha do núcleo separada da pilha do processo interrompido. Por quê?

Porque, durante uma interrupção, é o sistema o responsável por liberar outro processo. Logo, se se utilizar a mesma área da pilha do processo interrompido, este novo processo será interrompido juntamente.

5) Seria possível estabelecer uma medida sobre quanto um processo é limitado a CPU ou limitado a E/S analisando o código-fonte? Como isso poderia ser determinado em tempo de execução?

Observando-se o código: um processo é limitado à E/S caso ele apresente muitas operações de leitura e escrita. Por sua vez, se o processo possuir muitas operações lógico-aritméticas, ele é limitado pela CPU.

Em tempo de execução: um processo que é limitado à E/S está constantemente interrompido, aguardando o término de suas leituras e escritas. Os limitados à CPU simplesmente executarão, em seu tempo.

6) Uma falta de página é suficiente para bloquear um processo quando se usa o modelo de *threads*? Por quê?

Sim, caso ele tenha sido implementado em modo usuário, porque o núcleo desconhece a existência dos threads e, com isso, bloqueia todo o processo até que a E/S do disco termine.

7) Um sistema computacional tem espaço suficiente para conter 5 programas em sua memória principal. Esses programas estão ociosos, esperando por E/S, metade do tempo. Qual fração do tempo da CPU é desperdiçada?

A utilização (U) da CPU para um certo tempo de espera de E/S (p) e uma determinada quantidade (n) de programas se dá pela fórmula: $U = 1 – p^n$.

Assim, sendo 5 os programas e metade (½) o tempo, $U = 1 – (½)^5$, ou seja, $U = 31/32$, o que dá, aproximadamente, 97%. Logo, a fração de tempo de CPU desperdiçada é de aproximadamente 3%.

8) No texto, descrevemos um servidor Web multithread, mostrando porque ele é melhor que um servidor de thread único e do que um servidor de máquina de estados finitos. Existe alguma circunstância na qual um servidor de thread único possa ser melhor? Se existir, dê um exemplo e explique o porquê.

Sim. Servidores de thread único são melhores quando seus processos são limitados à CPU, e não à E/S, não se justificando a complexidade de implementar um sistema multithread.

9) Considere um sistema multiprogramado com grau 6 (seis programas na memória ao mesmo tempo.) Presuma que cada processo passe 40% do seu tempo esperando pelo dispositivo de E/S. Qual será a utilização da CPU?

Esta é a mesma fórmula do item 7: $U = 1 - p^n$. Aplicando-a, temos que a variável $U = 1 - (2/5)^6$, ou seja U (utilização da CPU) é aproximadamente 99,5%.

10) Um computador tem 4GB de RAM da qual o SO ocupa 512MB. Os processos ocupam 256MB cada, e têm as mesmas características quanto ao tempo que ficam esperando por E/S. Se a meta é atingir uma utilização da CPU de 99%, qual a porcentagem de espera de E/S máxima que os processos devem ter?

Tendo-se 4GB ($2^2 * 2^{30}$), e o SO ocupando 512MB ($2^9 * 2^{20}$), pode-se dizer que este ocupa 1/8 da RAM. Se um processo tem 256MB ($2^8 * 2^{20}$), pode-se dizer que cada um deste ocupa 1/16 da RAM.

Porém, da RAM, está disponível apenas 14/16, pois 2/16 (1/8) dela foi gasto pelo SO.

Assim, se cada processo gasta apenas 1/16 de RAM e há 14/16 disponíveis desta, pode haver até 14 processos de 256MB. Assim sendo, aplica-se, mais uma vez, a fórmula $U = 1 - p^n$ para determinar a porcentagem (p) máxima de espera de E/S que estes 14 processos devam ter para U = 99%.

$$U = 1 - p^n \Rightarrow \frac{99}{100} = 1 - p^{14} \Rightarrow \sqrt[14]{\frac{100}{100} - \frac{99}{100}} = p \Rightarrow p \approx \sqrt[14]{\frac{1}{100}}$$

SEGUNDO QUESTIONÁRIO

1) Para cada um dos seguintes endereços virtuais decimais, calcule o número da página virtual e o deslocamento, para: a) uma página de 4KB; b) uma página de 8KB. Os endereços são: 20000, 32768 e 60000.

Ora, 4KB são 4096 bytes. Sabendo disso, o número da página virtual será o resultado da divisão de cada um dos endereços por 4096 acrescido de 1, e o deslocamento será o resto da divisão. Assim, para o endereço de 20000, tem-se uma página virtual de 5 (4+1) e um deslocamento de 3616. Para 32768, o número da página é 9, o deslocamento é 0. Para 60000, o número de página é 15 e o deslocamento é 2656.

O mesmo vale para uma página de 8KB; a única diferença é que 8KB são 8192 bytes.

2) Considere um sistema de troca de processos entre a memória e o disco no qual a memória é constituída dos seguintes tamanhos de lacunas em ordem na memória: 10KB, 4KB, 20KB, 18KB, 7KB, 9KB, 12KB e 15KB. Qual lacuna é tomada pelas solicitações sucessivas do segmento de (para o first, best, worst e next fit):

a) 12KB b) 10 KB c) 9KB

a) 12KB, no first fit, ocupará o espaço de 20KB, pois é o primeiro em que ele cabe. No best fit 12KB, pois é o que desperdiça menos espaço. No worst fit, ele ocupa 20KB, pois é o que mais desperdiça espaço. O next fit, na primeira alocação, funciona como um first fit. Na segunda, ele vai procurar a que primeiro encaixa a partir da posição em que ele parou. Por exemplo: os 12KB vão ficar em 20KB por ser o first fit. Já os próximos 10KB, no next fit, vão ficar em 18KB, por ser o primeiro espaço que suporta os 10KB em sua totalidade.

b) FF: 10KB; BF: 10KB; WF: 20KB; NF: 18KB.

c) FF: 10KB; BF: 9KB; WF: 20KB; NF: 9KB.

3) Um computador com um endereçamento de 32 bits usa uma tabela de páginas de dois níveis. Os endereços são quebrados em um campo de 9 bits para a tabela de páginas de nível 1, um campo de 11 bits para a tabela de páginas de nível 2 e um deslocamento. Qual o tamanho das páginas e quantas existem no espaço de endereçamento citado? Explique sua resposta.

Se há um campo de 9 + 11 bits para as respectivas páginas de nível 1 e 2, e o computador é de 32 bits, então o deslocamento será de 32 – 11 – 9 bits, ou seja, 12 bits. O tamanho das páginas é 2 elevado à quantidade de bits do deslocamento, logo, é 2^{12} ou 4KB.

O número de páginas disponíveis é 2 elevado ao resultado da soma do número de bits dos campos das páginas de nível 1 (9 bits) e de nível 2 (11 bits), logo, o número de páginas disponíveis é 2^{9+11} ou 2^{20} páginas.

4) Uma máquina tem um endereçamento virtual de 48 bits e um endereçamento físico de 32 bits. As páginas são de 8KB. Quantas entradas são necessárias para a tabela de páginas? Por quê?

Para saber quantas entradas são necessárias na tabela de páginas, deve-se subtrair, da quantidade de bits do endereço, o valor do deslocamento, e elevar dois a esse resultado.

O valor do deslocamento diz qual o tamanho da página; se o tamanho é 8KB (2^{13}), então o deslocamento é de 13 bits.

Logo, o número de entradas exigido é $2^{(32-13)}$ ou 2^{19}.

O mesmo faz-se ao endereçamento virtual.

8) No algoritmo WSClock da figura (fig. 3.20(c) do livro), o ponteiro do relógio aponta para a página com R=0. Se τ = 400, essa página será removida? Por quê? O que acontecerá se τ = 1000? Por quê?

Como a idade da página (2204-1213=991) é maior do que τ (400), ela será removida. No entanto, para τ = 1000, ela será mantida, pois sua idade é menor.

5) Se o algoritmo de substituição FIFO é usado com quatro molduras de página e oito páginas virtuais, quantas faltas de página ocorrerão com a cadeia de referências 0-1-7-2-3-2-7-1-0-3 se os quatro quadros estão inicialmente vazios? Agora repita este problema para LRU. Explique sua resposta.

FIFO significa First-In, First-Out, ou seja, o primeiro a entrar é o primeiro a sair. Se há oito páginas virtuais, há oito números, de 0 a 7. No entanto, só poderão ficar, por vez, quatro destes, pois só há quatro quadros. Os quatro primeiros valores inseridos (0, 1, 7, 2) não fazem nada, logo, ficarão conforme na figura 1.

Ao se tentar inserir o valor 3, como ele ainda não existe nos quadros, o primeiro valor inserido neles deverá ser removido, ou seja, o valor 0 (figura 2.)

Como os valores seguintes (2, 7, 1) já existem nos quadros, nada acontecerá. Ao se tentar inserir o valor 0 mais uma vez, porém, o valor 1 terá de ser removido, pois é o mais velho ainda no quadro (figura 3.)

Assim, conclui-se que, pelo FIFO, seriam necessárias duas trocas de páginas.

Figura 1

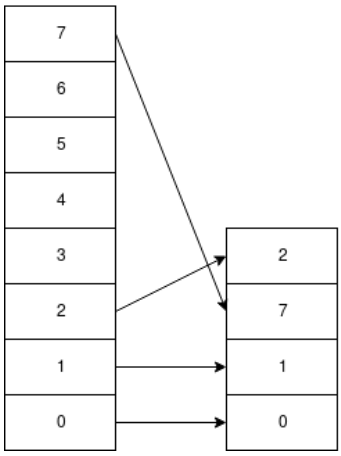


Figura 2

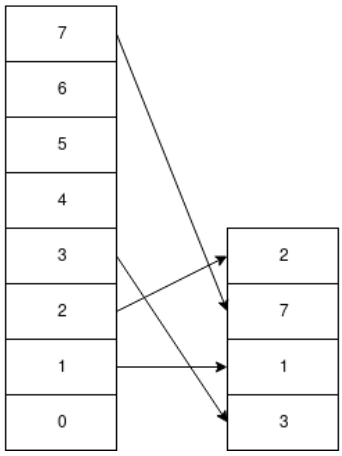
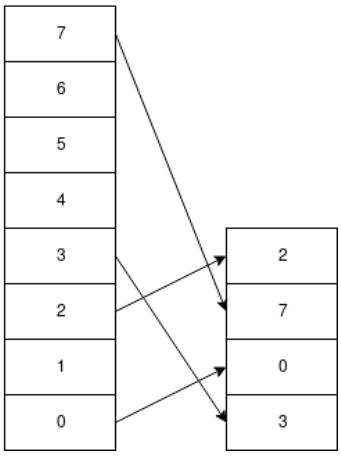


Figura 3



Para o LRU, o princípio é semelhante. No entanto, o LRU funciona da seguinte forma: o que foi usado menos recentemente, será excluído. Assim, deve-se observar quando números repetidos são inseridos, porque isso significa que aquele número foi usado recentemente. Levando em conta isso, são necessárias três trocas de página, conforme as figuras abaixo:

Figura 1

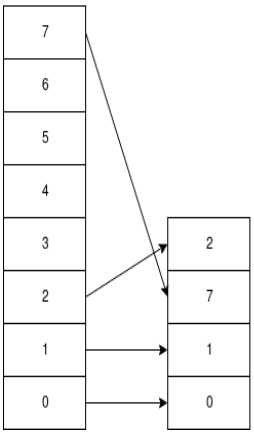


Figura 2

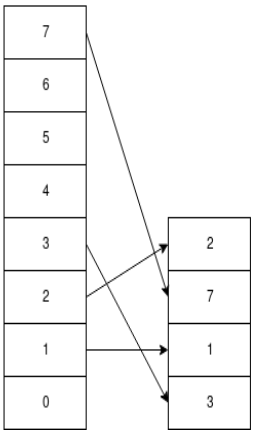


Figura 3

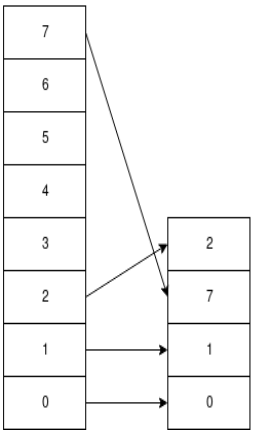
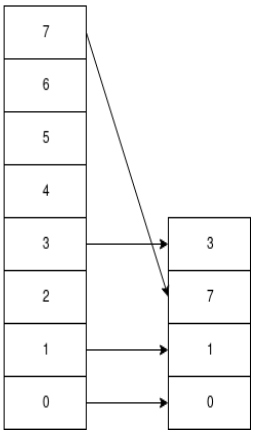
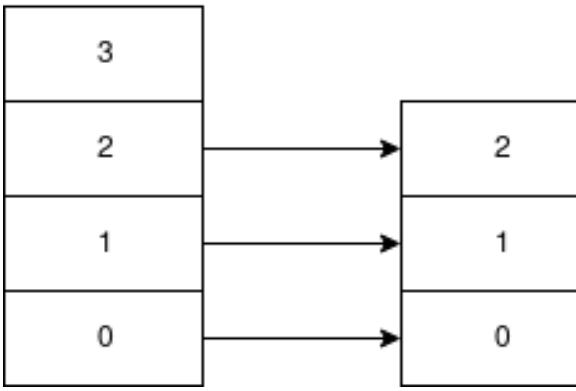


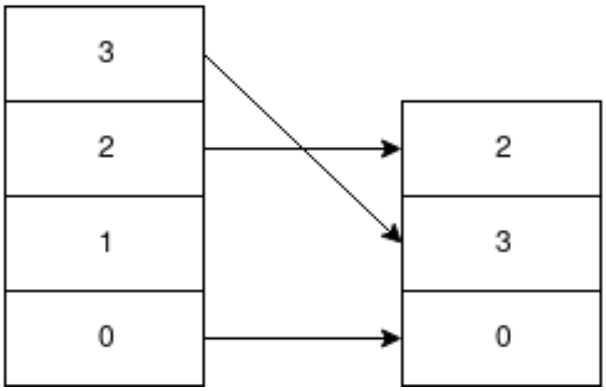
Figura 4

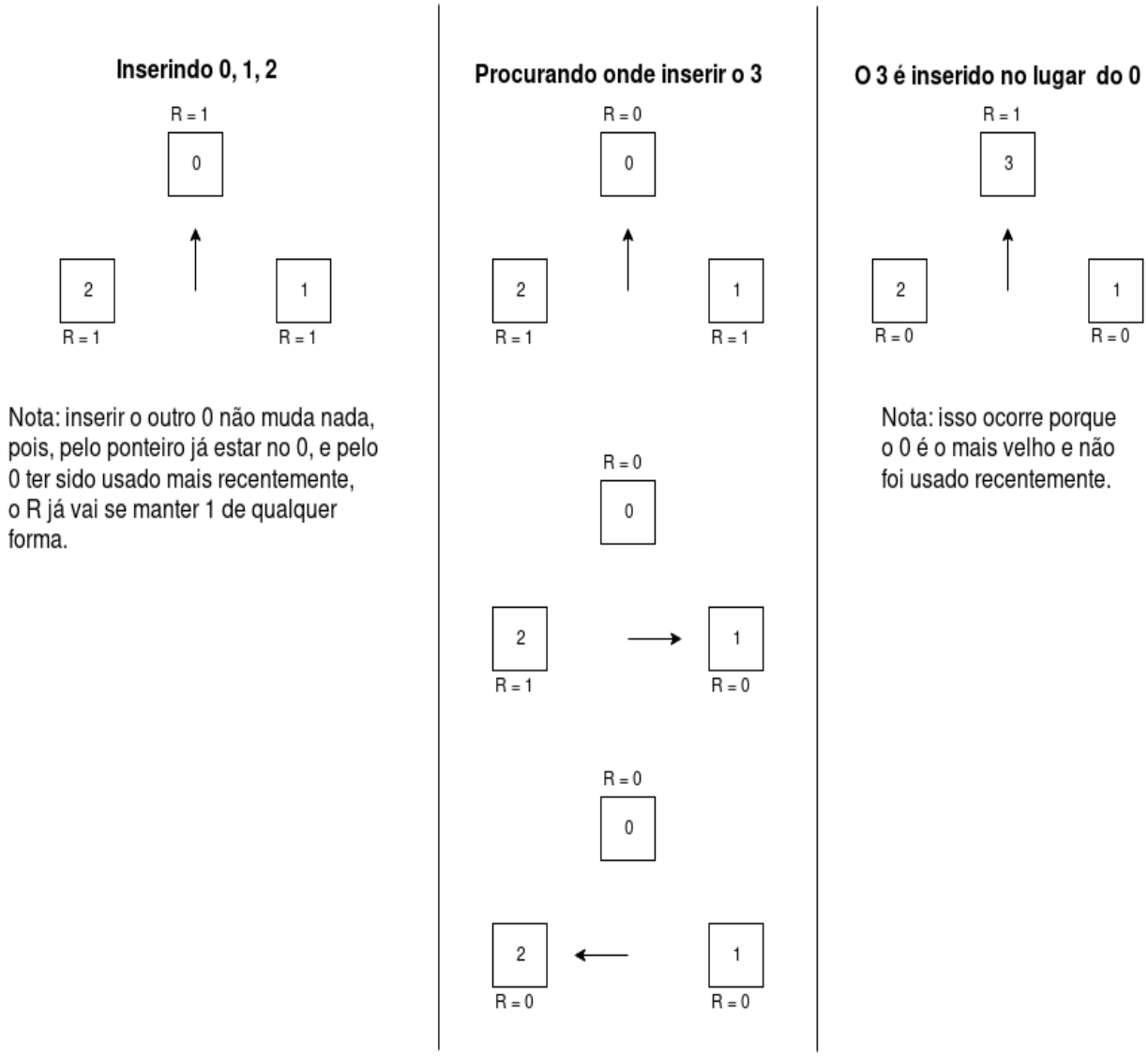


7) Dê um exemplo simples de uma sequência de referência de páginas onde a primeira página selecionada para substituição seja diferente para os algoritmos de substituição de página relógio e LRU. Suponha que em um processo sejam alocadas 3 molduras e que a cadeia de referência contenha números e páginas do conjunto 0, 1, 2 e 3.

A sequência é 0, 1, 2, 0 e 3. No LRU e no relógio, respectivamente:







6) Um computador pequeno tem quatro molduras de páginas. No primeiro tique de relógio, os bits R são 0111 (página 0 é 0, as demais são 1.) Nos tiques subsequentes, os valores são, respectivamente: 1011, 1010, 1101, 0010, 1010, 1100 e 0001. Se o algoritmo do envelhecimento (*aging*) é usado com um contador de 8 bits, quais os valores dos quatro contadores após o último tique? Por quê?

Se é usado um contador de 8 bits e há quatro molduras de páginas, então são quatro números de oito bits cada. Logo, para um bit R = 0111, o primeiro bit do primeiro número recebe o valor 0; o primeiro bit do segundo número recebe o valor 1 e assim por diante. A tabela completa pode ser vista abaixo:

0111	1011	1010	1101	0010	1010	1100	0001
00000000	10000000	11000000	11100000	01110000	10111000	11011100	01101110
10000000	01000000	00100000	10010000	01001000	00100100	10010010	01001001
10000000	11000000	11100000	01110000	10111000	11011100	01101110	00110111
10000000	11000000	01100000	10110000	01011000	00101100	00010110	10001011

TERCEIRO QUESTIONÁRIO

1) Por que a atomicidade dos semáforos é fundamental para resolver problemas de sincronização e evitar condições de corrida?

Porque ações atômicas garantem que, uma vez iniciada uma operação de semáforo, nenhum outro processo poderá acessá-lo até que a operação tenha sido concluída.

5) Se um sistema tem apenas 2 processos, tem sentido usar uma barreira para sincronizá-los? Por quê?

Se o programa trabalha em fases e nenhum processo puder ir à fase seguinte até que ambos tenham terminado a fase atual, faz sentido usar uma barreira.

7) Cinco tarefas estão esperando para serem executadas. Seus tempos de execução previstos são 9, 6, 3, 5 e X. Em que ordem elas deveriam ser executadas para minimizar o tempo médio de resposta? (Depende de X.)

Para x <= 3: X, 3, 5, 6, 9.

Para 3 <= x <= 5: 3, X, 5, 6, 9.

Para 5 <= x <= 6: 3, 5, X, 6, 9.

Para 6 <= x <= 9: 3, 5, 6, X, 9.

Para x => 9: 3, 5, 6, 9, X.

8) Explique por que o escalonamento em 2 níveis é bastante usado.

Porque ele evita que o processador fique ocioso, uma vez que o escalonador escolhe o processo de maior prioridade e de menos tempo, e coloca-o na memória principal, enquanto os demais ficam alocados em disco.

9) Um sistema de tempo real precisa controlar duas chamadas de voz, cada uma delas executada a cada 5ms e consumindo 1ms do tempo da CPU por surto, além de um vídeo de 25 quadros/s, cada quadro requerendo 20ms do tempo da CPU. Esse sistema pode ser escalonado? Por quê?

Sendo o consumo do tempo da CPU a variável C, e o período de execução de cada processo a variável P, para um sistema ser escalonável, a soma de todos os C/P dos m eventos deve ser menor que 1, ou seja:

$$\sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1$$

Como temos dois eventos acontecendo neste caso, deve-se primeiro descobrir C_1 e P_1. Pelo próprio enunciado, sabe-se que C_1 = 1ms, e que P_1 = 5ms. Pelo enunciado, sabe-se também que C_2 = 20ms. No entanto, para determinar P_2, precisa-se dividir 1000ms (que equivalem a 1s) por 25 (já que são 25 quadros por s.) Assim, acha-se que P_1 = 40ms. Agora, basta montar as frações e resolver. Se o resultado der menor que 1, o sistema é escalonável; senão, ele não é. Atente-se que C_1/P_1 deve ser somado duas vezes, porque há DUAS chamadas de voz acontecendo simultaneamente.

10) O algoritmo de envelhecimento (*aging*) com a = ½ está sendo usado para prever tempos de execução. As quatro execuções anteriores, da primeira a mais recente, são: 40, 20, 40 e 15ms. Qual a previsão da próxima execução?

A fórmula de envelhecimento, para um termo m qualquer, é:

$$T_m = \frac{T_0}{2^{m-1}} + \sum_{i=1}^{m-1} \frac{(a^{-1}-1)*T_i}{(a^{-1})^{(m-i)}}$$

Assim, se queremos T_4 (pois T_0 = 40) para a = ½:

$$T_4 = \frac{T_0}{2^3} + \frac{T_1}{2^3} + \frac{T_2}{2^2} + \frac{T_3}{2^1}$$

$$T_4 = \frac{40}{8} + \frac{20}{8} + \frac{40}{4} + \frac{15}{2} = 25\,ms$$

QUARTO QUESTIONÁRIO

1) Um computador tem um endereçamento de 32 bits e usa uma tabela de páginas de dois níveis. O tamanho das páginas é de 8KB. Existem 256 tabelas de páginas de nível 2. Os endereços são quebrados em um campo de Y bits para a tabela de páginas de nível 1, um campo de Z bits para a tabela de páginas de nível 2 e X bits para o deslocamento. Quanto valem X, Y e Z? Justifique.

O tamanho das páginas é 8KB, ou seja, 2^{13}. O número 2 elevado ao número de bits (X) do deslocamento é o valor deste tamanho de páginas, logo, X = 13.

Se há 256 tabelas de nível 2, e 256 é 2^8, então o campo da tabela de páginas de nível 2 é de 8 bits, ou seja, Z = 8.

Como o sistema é de 32 bits, a soma dos campos das tabelas de páginas de nível 1 e 2 e do deslocamento deve dar 32 bits. Logo, para saber quantos bits (Y) há para o campo da tabela de página de nível 1, basta subtrair, de 32, os valores 8 e 13. Assim, encontra-se Y = 11.

2) Como podemos garantir uma grande quantidade de molduras de página disponíveis na memória? Quais os dois motivos que fazem melhorar o desempenho quando isto acontece? Por quê?

Por meio do *daemon* de paginação, que atua quando um número pequeno de quadros de página estiver disponível. Ele seleciona páginas a serem removidas por meio de um algoritmo de substituição. Se estas foram modificadas desde que foram carregadas, elas serão escritas para o disco.

Melhora-se o desempenho porque, se uma das páginas removidas ser necessária novamente antes de seu quadro ser sobreposto por uma nova página, ela pode ser reobtida, retirando-a do conjunto de quadros de páginas disponíveis.

Outra melhora de desempenho ocorre porque manter uma oferta de quadros de páginas disponíveis é melhor do que usar toda memória e só então tentar encontrar um novo quadro, no momento em que ele for necessário.

5) O que é uma instrução TRAP? Explique seu uso em sistemas operacionais. Qual a diferença fundamental entre uma TRAP e uma interrupção?

É uma instrução que indica uma exceção durante a execução de um processo, como uma divisão por 0 ou um acesso inválido à memória. É por meio dela, também, que se faz uma exceção a um processo e permite com que ele faça uma chamada de sistema a algum dos recursos do *kernel*, ou seja, é a instrução TRAP a responsável por chavear do modo usuário para o modo núcleo. A instrução TRAP, portanto, a despeito de tratar-se de exceções, ocorre sempre de forma planejada, esperada. Já uma interrupção é um evento que ocorre a partir do *hardware*, e que é imprevisível; a CPU apenas lida com elas, passivamente.

7) Cinco tarefas em lote, A, B, C, D e E, chegam a um centro de computação quase ao mesmo tempo. Elas têm tempos de execução estimados em 12, 8, 4, 6 e 10 minutos, respectivamente. Determine o instante em que cada tarefa é executada, usando escalonamento FIFO e usando tarefa mais curta primeiro.

FIFO: A = 12m; B = 20m (8 + 12); C = 24m (4 + 20); D = 30m (6 + 24); E = 40m (10 + 30.)

Mais curta primeiro: C = 4m; D = 10m; B = 18m; E = 28m; A = 40m.

ANOTAÇÕES

Escalonador

- **Preemptivo:** escolhe um processo e o deixa executar por um tempo máximo previamente estipulado. Exige uma interrupção do relógio ao fim deste tempo, para que se possa devolver o controle da CPU de volta ao escalonador.
- N**ão-preemptivo:** escolhe um processo para executar e o deixa, até que seja bloqueado ou libere, voluntariamente, o uso da CPU.

Escalonamento em lotes

- **Primeiro a chegar, primeiro a ser servido:** é não-preemptivo. Os processos são executados à medida que chegam na CPU.
- **Tarefa mais curta primeiro:** com o tempo previamente conhecido, executa-se o processo mais curto primeiro, porque assim se reduz o tempo médio de espera.

Escalonamento de sistemas interativos

- **Chaveamento circular (round-robin):** é dado um *quantum*, uma fatia de tempo, na qual os processos devam executar. Estourado o *quantum* ou bloqueado o processo, este retorna ao fim da fila, e o próximo na fila é executado em seu lugar.

- **Escalonamento por prioridades:** a cada processo é dada uma prioridade; os que a tiverem mais elevada terão permissão de executar primeiro. Para evitar que processos rodem indefinidamente, pode-se ou se lhes estipular um *quantum*, ou diminuir-lhes a prioridade a cada tique do relógio. Em geral, é bom dar alta prioridade a um processo limitado à E/S porque, desta forma, desperdiça-se menos CPU. Isto ocorre porque, enquanto o processo espera sua E/S, a CPU pode ocupar-se de processar outro processo.
- **Escalonamento garantido:** para n usuários conectados, cada um recebe $1/n$ da CPU para trabalhar. O mesmo pode ser aplicado a processos de apenas um usuário, sendo n o número de processos deste usuário, é dado $1/n$ de tempo de CPU para cada processo executar nela.
- **Escalonamento por loteria:** é altamente responsivo. Dá bilhetes aos processos. A CPU escolhe um bilhete ao acaso e será executado o processo que tiver tal bilhete. Processos de maior prioridade podem receber mais bilhetes. Processos cooperativos podem ceder bilhetes entre si.
- **Escalonamento por fração justa:** garante uma proporção justa na execução de processos de diferentes usuários, dando, a cada um destes, uma fração similar de acesso à CPU, para que esta rode os processos específicos de cada usuário de forma justa. Ou seja, se um usuário tem 10 processos e outro somente 2, a CPU organizará, de forma justa, que o segundo usuário não espere tanto para que se execute algum de seus 2 processos.
- **Escalonamento de processo mais curto em seguida:** estipula, por meio de um coeficiente de envelhecimento, um tempo mínimo de execução para os processos. Assim, pode-se organizá-los de forma que os de tempo mais curto sejam executados primeiro, diminuindo o tempo médio de resposta. Assim, para um coeficiente de envelhecimento fracionário a qualquer, se quisermos prever o tempo T de execução de um m-ésimo processo, pode-se usar esta fórmula:

$$T_m = \frac{T_0}{2^{m-1}} + \sum_{i=1}^{m-1} \frac{\left(a^{-1}-1\right)*T_i}{\left(a^{-1}\right)^{(m-i)}}$$

Escalonamento em sistemas de tempo real

- Um sistema de tempo real pode ser crítico, em que há prazos absolutos que devem ser cumpridos, ou não-crítico, em que falhas são toleráveis, ainda que indesejáveis.
- Um sistema de tempo real só será escalonável para *m* processos se, somando a divisão do tempo de surto da CPU pelo tempo do período de repetição, este resultado for menor que 1.

Memória virtual: cada programa tem seu próprio espaço de endereçamento, dividido em blocos, chamados páginas, as quais são séries contíguas de endereços. Páginas são mapeadas na memória física, mas não precisam estar todas, ao mesmo tempo, na memória.

M.M.U.: dispositivo responsável por mapear endereços virtuais em endereços reais.

Espaço de endereçamento virtual: são unidades de tamanhos fixos, chamadas páginas.

Quadros de páginas: são as unidades, do espaço de endereçamento virtual, correspondentes na memória física.

Tabelas de páginas multiníveis: há duas tabelas de página; a de nível 1 funciona como um índice, com o qual se acessará a página de nível 2, onde estão mapeados os quadros de página, isto é, os correspondentes na memória física.

AS CONTAS DO MAL

- Um sistema de X bits precisa ter bits de nível 1 e de nível 2 que, somando-os com o deslocamento, dê a mesma quantidade X de bits.
- Se um deslocamento tem Y bits, as páginas são de tamanho 2^Y.
- Se o tamanho do campo de endereçamento do primeiro nível da página é V bits, e é W bits o da página de segundo nível, há um total de $2^{(V+W)}$ páginas disponíveis.

ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINA

Algoritmo ótimo de substituição de páginas:

- É impossível de implementá-lo, porque ele parte do pressuposto que se conhece em quantas instruções futuras uma página será referenciada novamente. Aquela que só for utilizada novamente muito mais tarde será a removida.

Substituição de páginas não usadas recentemente (NRU):

- Há um bit R, usado para determinar se uma página foi referenciada ou não. Este bit é setado sempre que uma página for lida ou escrita.
- Há um bit M, usado sempre que uma página for modificada, ou seja, este bit é setado sempre que uma página for escrita.
- Quando um processo é iniciado, os bits R e M de suas páginas são setados para 0. Periodicamente, o bit R é limpo, indicando que as páginas não foram usadas recentemente. O bit M nunca é limpo.
- Com a informação destes dois bits, tem-se 4 classes: 0 (R=0;M=0), 1 (R=0;M=1), 2(R=1;M=0) e 3 (R=1;M=1.)
- O NRU removerá, ao acaso, uma página de classe mais baixa que não esteja vazia.

FIFO (primeiro a entrar, primeiro a sair):

- Irá simplesmente remover a página que está há mais tempo na memória.

Segunda chance:

- No FIFO, páginas referenciadas recentemente podem acabar por ser removidas. Por conta disto, fez-se esse algoritmo.
- Antes de remover a página mais velha, observa-se seu bit R. Se ele for 1, ele é zerado, e a página volta para o início da fila, como se fosse recém-chegada.
- No entanto, se todas as páginas tiverem R=1, este algoritmo funciona igual a um FIFO puro.

Relógio:

- O mesmo que o anterior, a diferença é que usa uma lista circular em vez de uma encadeada, para melhor desempenho.
- Se R=1, ele zera o R e passa à próxima página até que encontre uma com R=0.

Páginas usadas recentemente (LRU):

- É o mais próximo do algoritmo ótimo. Quando ocorre uma falta de página, o LRU substituirá uma que não foi utilizada há mais tempo.
- É muito difícil de implementar, pois é custoso: a cada referência a uma página, toda a lista encadeada deve ser rearranjada a fim de colocar a mais recentemente usada ao final da lista.

Não usada frequentemente (NFU):

- A cada tique do relógio, o contador de cada uma das páginas é deslocado um bit à direita. Depois, se uma página foi recentemente referenciada, o bit menos significativo de seu contador recebe 1; se não, recebe 0. Isto se designa algoritmo de envelhecimento.
- Quando ocorre uma falta de página, a página de contador mais baixo é a removida.
- Um dos problemas deste algoritmo é a impossibilidade de comparar referências mais velhas do que o mapeado pelos bits limitados do contador.
- Também é impossível saber, dentro de um intervalo, qual página foi referenciada primeiro, isto é, se duas páginas são referenciadas dentro de um mesmo intervalo, é difícil saber qual foi a mais recentemente referenciada. Logo, se se precisar remover uma destas, pode-se acabar por remover a mais recentemente usada.

Conjunto de trabalho:

- Conjunto de trabalho é o conjunto de páginas que o processo está usando atualmente.
- Faz-se uma estimativa de qual conjunto de trabalho um processo exigirá e, então, antes de carregá-lo, carrega-se as páginas (pré-paginação é o nome disto) que lhe serão necessárias à execução. Tudo isto é para evitar, ao máximo, faltas de página.

WSClock:

- Implementa o algoritmo anterior em formato de relógio, isto é, de lista circular. É muito utilizado na prática. Dentro de cada nó da lista (ou seja, cada página), consta o instante do último uso, o bit R e o bit M.
- Quando ocorre uma falta de página, a que estiver sendo apontada pelo ponteiro é examinada primeiro.
    - Se R = 1, ele é setado para 0 e o algoritmo avança o ponteiro à próxima página. Se o bit R = 0 e a idade da página é maior que τ e a página está limpa, então ela não está no conjunto de trabalho e há uma cópia sua em disco. Neste caso, o quadro é reivindicado e a nova página é colocada lá.
    - Se R = 0 e a idade é maior que τ mas a página não está limpa, ela não pode ser reivindicada porque não há cópia em disco. Então o ponteiro segue se movendo até encontrar uma limpa. Se não encontrar nenhuma, a atual será removida.

Tamanho de página ótimo: o tamanho de página (P) ótimo se dá por P = √(SE), sendo *S* o tamanho médio, em bytes, do processo, e *E* a quantidade, em bytes, que cada entrada de página exige.