Sistemas Operacionais
Sétima Lista de Exercícios – Solução

Norton Trevisan Roman
Clodoaldo Aparecido de Moraes Lima

15 de novembro de 2012

1. Não. Basta acessar o byte 0

2. Sim. Há todo um processo de alocação de blocos e liberação de outros, além de i-nodes ou entradas na FAT, envolvidos na criação/remoção. Sem contar que, ao renomear um arquivo, os demais atributos não são mudados, enquanto que copiá-lo mudaria o tempo de criação e última modificação.

3. Use o nome como se fosse o caminho. Ex: "/meu/arquivo"

4. A 8MB/s, 8KB são transferidos em $\frac{8KB}{8MB} = \frac{1}{1024}s \Rightarrow \frac{1000}{1024}ms$. A cópia de um arquivo envolve 1 posicionamento, 1 atraso e a transferência total de seus 8KB, tanto para sua leitura, quanto escrita. O tempo total de transferência será então $2 \times (5ms + 4ms + \frac{1000}{1024}ms) = 19,95ms$.

   Metade dos 16GB (8GB) conterão $\frac{8GB}{8KB} = 1.048.576$ arquivos. Como cada arquivo leva 19,95ms, temos $1.048.576 \times 19,95 = 20.919.091,2ms \Rightarrow \frac{20.919.091,2}{1000} \approx 20.919,09s \Rightarrow \frac{20.919,09}{3600} \approx 5,8h$

5. Se feito corretamente, sim, dado o ganho em tempo de se ter um arquivo com blocos consecutivos. Contudo, e pelo tempo despendido, essa operação não deve ser feita com grande frequência.

6. Hard links não ocupam espaço extra, enquanto que soft links ocupam. Além disso, se o arquivo for apagado, o link não é quebrado. Soft links podem se ligar a arquivos em outras partições, HDs, ou mesmo máquinas.

7. Temos 10 blocos diretos $+\frac{1024}{4} = 256$ blocos indiretos, num total de 266 blocos de 1024B $\Rightarrow$ 266KB

8. Serão 9 endereços (os 8 diretos e o espaço para o indireto) de 4B $\Rightarrow 9 \times 4B = 36B$

9. Depende realmente de como essa lista é implementada (se conterá um endereço de bloco e um endereço para o próximo elemento na lista, ou conterá os endereços blocos livres agrupados como em um arranjo. Considerando o último modelo, podemos dizer que cada bloco livre na lista ocupará D bits (ou seja, seu próprio endereço). Assim, com F blocos livres a lista terá um tamanho de DF bits.

   O bitmap, por outro lado, mapeia o disco inteiro, possuindo um total de B bits.

   Para que a lista ocupe menos espaço que o bitmap temos que $DF < B$. Com $D = 16$ bits, teremos que $16F < B \Rightarrow \frac{F}{B} < 1/16 = 0,0625 = 6,25\%$. Assim, a lista de livres é menor se 6,25% ou menos do disco estiver livre

10. Considerando 1 como ocupado e 0 livre, temos, após A, 1111 1110 0000 0000

    (a) 1111 1111 1111 0000

    (b) 1000 0001 1111 0000

    (c) 1111 1111 1111 1100

(d) 1111 1110 0000 1100

11. Não saberia de onde tirar espaço para alocar mais arquivos. Basta no entanto reconstruir, a partir dos i-nodes, a lista de blocos livres → sabe-se os ocupados, então o que sobrar (e não estiver danificado) é livre. No caso da FAT, o problema não ocorreria, pois não há lista de livres (a FAT reflete todo o HD, tanto com blocos livres quanto ocupados). Mas mesmo que houvesse, bastaria varrer a FAT, buscando entradas livres.

12. Em n requisições, temos $h = \frac{n_h}{n}$ e $m = \frac{n_m}{n}$ (onde h é a taxa de acertos e m de falhas). Assim, em n requisições teremos $n \times h$ hits e $n \times (1-h)$ misses (pois $m = 1-h$). Como um acerto leva 1ms e uma falha leva 40ms, o tempo total, de n requisições, será de $n \times h + n \times (1-h) \times 40$ms.

    O tempo médio será então

    $$\frac{n \times h + n \times (1-h) \times 40}{n} = h + 40 - 40h = 40 - 39h$$

    O gráfico tempo médio × h será uma reta descendente, em que $y = 40$ quando $x = 0$ e $y = 1$ quando $x = 1$

13. A 15.000rpm, o disco dá $\frac{15.000}{60} = 250$ voltas/s, ou uma volta em $\frac{1}{250} = 0,004 = 4ms$. A leitura de um bloco aleatório, contudo, envolve também um posicionamento e uma latência (atraso rotacional). O posicionamento é de 8ms. Para a latência, podemos considerar que, em média, meia trilha será buscada ⇒ 2ms ($\frac{1}{2}$ volta, sendo que a volta leva 4ms). O tempo total de transferência de um bloco será então $T_{total} = T_{pos} + T_{lat} + T_{transf}$, restando-nos calcular o tempo de transferência do bloco.

    Blocos de 1KB: Com 1KB (1024B), cabem 262.144/1024 = 256 blocos em uma trilha. Como a trilha leva 4ms para ser lida completamente, a transferência de um único bloco levará $4/256 = 0,015625$ms. O tempo total será então $T_{total} = 8 + 2 + 0,015625 = 10,015625$ms. Com 1024B em cada bloco, temos uma taxa de $1024 \times 1000/10,015625 \approx 102.240$B/s

    Blocos de 2KB: Temos 262.144/2048 = 128 blocos em uma trilha. A transferência de um único bloco levará $4/128 = 0,03125$ms. O tempo total será então $T_{total} = 8 + 2 + 0,03125 = 10,03125$ms. Com 2048B em cada bloco, temos uma taxa de $2048 \times 1000/10,03125 \approx 204.162$B/s

    Blocos de 4KB: Temos 262.144/4096 = 64 blocos em uma trilha. A transferência de um único bloco levará $4/64 = 0,0625$ms. O tempo total será então $T_{total} = 8 + 2 + 0,0625 = 10,0625$ms. Com 4096B em cada bloco, temos uma taxa de $4096 \times 1000/10,0625 \approx 407.056$B/s

14. O tempo para ler cada bloco é dado por posicionamento + latência + transferência. Em ambos os casos, a latência será de 100ms e a transferência de 25ms ⇒ 125ms. O posicionamento, contudo, varia.

    Para o primeiro caso, cada bloco exige que o braço corra 13 cilindros, levando $6 \times 13 = 78$ms, num total de $125 + 78 = 203$ms. Assim, e supondo que o primeiro bloco levou o mesmo tempo dos demais, 100 blocos levariam $100 \times 203 = 20,3s$

    Para o segundo caso, cada bloco exige que o braço corra 2 cilindros, levando $6 \times 2 = 12$ms, num total de $125 + 12 = 137$ms. Assim, 100 blocos levariam $100 \times 137 = 13,7s$

15. Como cada bloco terá um desperdício de 50%, esse será o desperdício total. Em um sistema real, raramente os arquivos serão todos assim, reduzindo consideravelmente o desperdício.

16. Com 15 bits, o SO conseguiria endereçar $2^{15} = 32K$ blocos. Se fizesse o bloco desse tamanho (32KB, valor máximo usado na FAT16), teria 32K entradas para blocos de 32KB ⇒ $32K \times 32K = 1GB$

17. A soma dos tamanhos de todos os arquivos continua com a restrição de não ultrapassar o tamanho do disco. Nada muda.

18. Com 1KB, cada bloco consegue armazenar $\frac{1024}{4} = 256$ entradas de 4B. Então

Tipo	Número de Entradas	Bytes
Direta	10	$10 \times 1KB = 10KB$
Indireta Simples	256	$256 \times 1KB = 256KB$
Indireta Dupla	$256 \times 256 = 64K$	$64K \times 1K = 64MB$
Indireta Tripla	$256 \times 256 \times 256 = 16M$	$16M \times 1KB = 16GB$

O total será então $10 + 256 + 64K + 16M = 16.843.018$ blocos de 1KB $\Rightarrow 16.06GB$

19. Com o i-node para o / na memória temos:

Bloco /

I-node usr

Bloco usr

I-node ast

Bloco ast

I-node cursos

Bloco cursos

I-node so

Bloco so

I-node ep.java

Num total de 10 acessos

20. (a) Bloco 4: missing block. Adicione-o à lista de livres.
Blocos 8 e 14: embora não relatada em aula, não é difícil descobrir o que fazer. Dá-se prioridade ao bloco em uso, e reconstrói-se a lista de blocos livres.

(b) O segundo tipo. O fato de estar na lista de livres permite que seja usado por outros arquivos, criando uma inconsistência. Se esse bloco estava sendo usado por um diretório (ou pior ainda, pelo diretório raiz), o sistema de arquivos pode ficar inconsistente.

21. Com blocos de 4KB, a partição de 400MB terá 100K blocos (102.400 blocos).

(a) Com 16b, a FAT não mapeará todos os blocos do diretório. O máximo que ela mapeia são $2^{16} = 65.536$ blocos. A solução é agrupar blocos (criando clusters, na nomenclatura da FAT). Como os blocos são potência de 2, o próximo, além do 4K é 8K. Com blocos de 8KB, a partição armazena 50K blocos (51.200). Nessas condições, o programa de 45KB ocupará $45/8 \approx 5,6 \rightarrow 6$ clusters e, consequentemente, 6 posições na FAT

(b) Com blocos de 4KB, o programa ocupa $45/4 = 11,25 \rightarrow 12$ blocos. Desses, 10 são armazenados diretamente no i-node, e os outros 2 ficam no bloco indireto simples

(c) Com 16 bits, $2^{16} = 65.536 = 65K$ entradas. Como cada entrada, por sua vez, também tem 16 bits (2B, pois guarda um endereço de bloco), serão $65K \times 2 = 130KB$

(d) Com blocos de 4KB e endereços de 2B, cada bloco pode armazenar 2K (2048) endereços. O número total de blocos será então 10 (diretos) + 2048 (indireto simples) + 2048^2 (indiretos duplos) = 4.196.362 blocos. Como cada bloco tem 4KB, o arquivo armazena $4.196.362 \times 4 = 16.785.448KB \approx 16GB$