Universidade Federal Rural de Pernambuco

Sistemas de Informação

Princípios de Software Básico

Lhaíslla Eduarda Cavalcanti Rodrigues da Silva

RECIFE-PE

Primeira atividade 2VA

Sistemas Arquivos

1. Dê cinco nomes de caminhos diferentes para o arquivo /etc./passwd. (Dica: lembre-se das entradas de diretório "." e "..").
    a. /etc/passwd

       /./etc/passwd
       /./../etc/passwd
       /./../../etc/passwd
       /etc/../etc/passwd
       /etc/../etc/../etc/passwd
       /etc/../etc/../etc/../etc/passwd
       /etc/../etc/../etc/../etc/../etc/passwd

2. No Windows, quando um usuário clica duas vezes sobre um arquivo listado pelo Windows Explorer, um programa é executado e dado aquele arquivo como parâmetro. Liste duas maneiras diferentes através das quais o sistema operacional poderia saber qual programa executar.
    a. Por meio do sistema de arquivos, é possível identificar qual a extensão responsável pela leitura e interpretação dos dados do arquivo. A nomeação de arquivos e a estruturação de arquivos. Quando um usuário clica duas vezes sobre o nome de um arquivo, o programa designado para essa extensão de arquivo é lançado com o arquivo como parâmetro.

3. Nos primeiros sistemas UNIX, os arquivos executáveis (arquivos a.out) começavam com um número mágico, bem específico, não um número escolhido ao acaso. Esses arquivos começavam com um cabeçalho, seguido por segmentos de texto e dados. Por que você acha que um número bem específico era escolhido para os arquivos executáveis, enquanto os outros tipos de arquivos tinham um número mágico mais ou menos aleatório como primeiro caractere?
    a. Essa abordagem é utilizada quando um mesmo pode lidar com vários tipos diferentes de arquivos. O fundamental para que um arquivo seja uma sequência de registros é a ideia de que a operação de leitura retorna um

registro e a operação de escrita sobrepõe ou anexa um registro.

4. A chamada de sistema open no UNIX é absolutamente essencial? Quais seriam as consequências de não a ter?
    a. Não é essencial. Se não estivesse presente, o sistema teria que abrir o arquivo como parte da primeira leitura ou gravação. Isso exigiria uma versão de leitura / gravação com o nome do arquivo.Para começar, se não houvesse "open", em todas as leituras seria necessário especificar o nome do arquivo a ser aberto. O sistema precisaria buscar o i-node para ele, embora isso pudesse ser armazenado em cache. Um problema que surge rapidamente é quando liberar o i-node de volta para o disco. Poderia expirar, no entanto. Seria um pouco desajeitado, mas pode funcionar.

5. Sistemas que dão suporte a arquivos sequenciais sempre têm uma operação para voltar arquivos para trás (rewind). Os sistemas que dão suporte a arquivos de acesso aleatório precisam disso, também?
    a. Não, Se você quiser ler o arquivo novamente, acesse aleatoriamente o byte 0

6. Alguns sistemas operacionais fornecem uma chamada de sistema rename para dar um nome novo para um arquivo. Existe alguma diferença entre usar essa chamada para renomear um arquivo e apenas copiar esse arquivo para um novo com o nome novo, seguido pela remoção do antigo?
    a. Para o usuário final não, logo que é um processo imperceptível, mas a nível computacional sim, logo que somente renomear de fato um arquivo somente será feita alterações nos blocos que contém este nome, e remover um arquivo e criar outro com o mesmo conteúdo é um processo mais “burocrático”. Também vale destacar que criando um novo arquivo ele estará em outra posição na memória secundária.

7. Em alguns sistemas é possível mapear parte de um arquivo na memória. Quais restrições esses sistemas precisam impor? Como é implementado esse mapeamento parcial?
    a. O mapeamento é possível por meio da implementação de diretórios, algumas restrições: manter vários atributos do arquivo, como o proprietário de cada um

e seu momento de criação, e armazená-los em algum lugar, limitando o nome do arquivo.

8. Um sistema operacional simples dá suporte a apenas um único diretório, mas permite que ele tenha nomes arbitrariamente longos de arquivos. Seria possível simular algo próximo de um sistema de arquivos hierárquico? Como?
    a. Sim, o usuário pode ter tantos diretórios quantos forem necessários para agrupar seus arquivos de maneira natural. Além disso, se múltiplos usuários compartilham um servidor de arquivos comum, como é o caso em muitas redes de empresas, cada usuário pode ter um diretório-raiz privado para sua própria hierarquia. É possível por meio do agrupamento de arquivos relacionados em um mesmo local.

9. No UNIX e no Windows, o acesso aleatório é realizado por uma chamada de sistema especial que move o ponteiro "posição atual" associado com um arquivo para um determinado byte nele. Proponha uma forma alternativa para o acesso aleatório sem ter essa chamada de sistema.
    a. Dois métodos podem ser usados para especificar onde começar a leitura. No primeiro, cada operação read fornece a posição no arquivo onde começar a leitura. No segundo, uma operação simples, seek, é fornecida para estabelecer a posição atual. Após um seek, o arquivo pode ser lido sequencialmente da posição agora atual. O segundo método é usado no UNIX e no Windows. Seek. Para arquivos de acesso aleatório, é necessário um método para especificar de onde tirar os dados. Uma abordagem comum é uma chamada de sistema, seek, que reposiciona o ponteiro de arquivo para um local específico dele. Após essa chamada ter sido completa, os dados podem ser lidos da, ou escritos para, aquela posição.

10. Considere a árvore de diretório da Figura 4.8. Se /usr/ jim é o diretório de trabalho, qual é o nome de caminho absoluto para o arquivo cujo nome de caminho relativo é ../ast/x?
    a. O caminho formado entre o diretório raiz e o arquivo.Cada usuário pode criar vários níveis de diretórios (ou subdiretórios), sendo que cada um pode conter

arquivos e subdiretórios. O componente de ".."move a pesquisa para /usr, então ../ast coloca em /usr/ast. Portanto ../ast/x é o mesmo que /usr/ast/x.

11. A alocação contígua de arquivos leva à fragmentação de disco, como mencionado no texto, pois algum espaço no último bloco de disco será desperdiçado em arquivos cujo tamanho não é um número inteiro de blocos. Estamos falando de uma fragmentação interna, ou externa? Faça uma analogia com algo discutido no capítulo anterior.
    a. Visto que o espaço de memória desperdiçado é entre as unidades de alocação (arquivos), e não dentro deles, esta é a fragmentação externa. Sendo análoga à fragmentação externa da memória principal que ocorre com um sistema de troca ou de um sistema usando a segmentação puro.

12. Descreva os efeitos de um bloco de dados corrompido para um determinado arquivo: (a) contíguo, (b) encadeado e (c) indexado (ou baseado em tabela).
    a. Se um bloco de dados for corrompido em um sistema de alocação contígua, somente esse bloco será afetado; o restante dos blocos do arquivo pode ser lido.No caso de alocação encadeada, o bloco corrompido não pode ser lido; Além disso, os dados de localização sobre todos os blocos iniciados neste bloco corrompido são perdidos. No caso de alocação indexada, apenas o bloco de dados corrompidos é afetado.

13. Uma maneira de usar a alocação contígua do disco e não sofrer com espaços livres é compactar o disco toda vez que um arquivo for removido. Já que todos os arquivos são contíguos, copiar um arquivo exige uma busca e atraso rotacional para lê-lo, seguido pela transferência em velocidade máxima. Escrever um arquivo de volta exige o mesmo trabalho. Presumindo um tempo de busca de 5 ms, um atraso rotacional de 4 ms, uma taxa de transferência de 80 MB/s e o tamanho de arquivo médio de 8 KB, quanto tempo leva para ler um arquivo para a memória principal e então escrevê-lo de volta para o disco na nova localização? Usando esses números, quanto tempo levaria para compactar metade de um disco de 16 GB?
    a. 5 ms, 4 ms, 80 MB/s = 8 e ^-6 = (1,953e+6 x 2)
14. Levando em conta a resposta da pergunta anterior, a compactação do disco faz algum sentido?

a. Sim, desempenho pode ser restaurado movendo os arquivos a fim de deixá-los contíguos e colocando todo (ou pelo menos a maior parte) o espaço livre em uma ou mais regiões contíguas no disco.

15. Alguns dispositivos de consumo digitais precisam armazenar dados, por exemplo, como arquivos. Cite um dispositivo moderno que exija o armazenamento de arquivos e para o qual a alocação contígua seria uma boa ideia.
    a. Os sistemas de arquivos Linux (especialmente ext2 e ext3) geralmente sofrem menos com a desfragmentação do que os sistemas Windows pela maneira que os blocos de discos são selecionados, então a desfragmentação manual raramente é exigida. Também, os SSDs não sofrem de maneira alguma com a fragmentação. Na realidade, desfragmentar um SSD é contraprodutivo. Não apenas não há ganho em desempenho, como os SSDs se desgastam; portanto, desfragmentá-los apenas encurta sua vida.

16. Considere o i-node mostrado na Figura 4.13. Se ele contém 10 endereços diretos e esses tinham 8 bytes cada e todos os blocos do disco eram de 1024 KB, qual seria o tamanho do maior arquivo possível?
    a. Há 10 ponteiros diretos para blocos de dados, 1 ponteiro indireto. O tamanho dos blocos de dados é de 1024 = 1KB, ou seja, Tamanho_do_Bloco = 1KB.Suponha que os números de bloco sejam representados como números inteiros não assinados de 8 bytes, ou seja, BlockNumberSize = 8b.O número máximo de bytes endereçados por 10 ponteiros diretos é = Número de ponteiros diretos , Tamanho do bloco =10*1KB = 10Kb , o número máximo de bytes endereçados por um ponteiro indireto é = NumeroEntradas x TamanhoBloco = (TamanhoBloco / TamanhodoEndereco) * TamanhoBloco =(1KB/8b)*1KB = 128 * 1 KB = 128Kb. O bloco indireto contém: 1024/8 = 128 endereços de bloco. Adicione os 10 endereços diretos para Total de 128 endereços de bloco 138 * 1024 = 138 KB (tamanho máximo do arquivo.

17. Para uma determinada turma, os históricos dos estudantes são armazenados em um arquivo. Os registros são acessados aleatoriamente e atualizados. Presuma que o histórico de cada estudante seja de um tamanho fixo. Qual dos três esquemas de alocação (contíguo, encadeado e indexado por tabela) será o mais apropriado?

Alocação por indexação, pois caso algum arquivo seja corrompido não afetará os demais arquivos.

18. Considere um arquivo cujo tamanho varia entre 4 KB e 4 MB durante seu tempo de vida. Qual dos três esquemas de alocação (contíguo, encadeado e indexado por tabela) será o mais apropriado? Alocação contígua, pois só precisa do Nome (e outros atributos) ,Bloco físico inicial e Tamanho do arquivo em blocos.Aborda execução do programa enviando mensagem de erro ao usuário Recopia o arquivo para uma porção maior (problema desempenho) Superdimensionar o espaço alocado no momento da criação Gera fragmentação interna se arquivo não usar todo o espaço pré alocado Usar extensões (lista de porções contíguas).

19. Foi sugerido que a eficiência poderia ser incrementada e o espaço de disco poupado armazenando os dados de um arquivo curto dentro do i-node. Para o i-node da Figura 4.13, quantos bytes de dados poderiam ser armazenados dentro dele?
    a. . Dado o i-node, é então possível encontrar todos os blocos do arquivo. A grande vantagem desse esquema sobre os arquivos encadeados usandouma tabela na memória é que o i-node precisa estar na memória apenas quando o arquivo correspondente estiver aberto. Se cada i-node ocupa n bytes e um máximo de k arquivos puderem estar abertos simultaneamente, a memória total ocupada pelo arranjo contendo os i- -nodes para os arquivos abertos é de apenas kn bytes. Apenas essa quantidade de espaço precisa ser reservada antecipadamente. Esse arranjo é em geral muito menor do que o espaço ocupado pela tabela de arquivos descrita na seção anterior. A razão é simples. A tabela para conter a lista encadeada de todos os blocos de disco é proporcional em tamanho ao disco em si. Se o disco tem n blocos, a tabela precisa de n entradas. À medida que os discos ficam maiores, essa tabela cresce linearmente com eles. Por outro lado, o esquema i-node exige um conjunto na memória cujo tamanho seja proporcional ao número máximo de arquivos que podem ser abertos ao mesmo tempo. Não importa que o disco tenha 100 GB, 1.000 GB ou 10.000 GB.

20. Duas estudantes de computação, Carolyn e Elinor, estão tendo uma discussão sobre i-nodes. Carolyn sustenta que as memórias ficaram tão grandes e baratas que, quando

um arquivo é aberto, é mais simples e mais rápido buscar uma cópia nova do i-node na tabela de i-nodes, em vez de procurar na tabela inteira para ver se ela já está ali. Elinor discorda. Quem está certa?

a. Carolyn esta certa, pois o i-node é um método para monitorar quais blocos pertencem a quais arquivos é associar cada arquivo a uma estrutura de dados chamada de i-node (index-node — nó-índice), que lista os atributos e os endereços de disco dos blocos do disco.

21. Nomeie uma vantagem de ligações estritas sobre ligações simbólicas e uma vantagem de ligações simbólicas sobre ligações estritas.

a. O problema com ligações simbólicas é a sobrecarga extra necessária. O arquivo contendo o caminho deve ser lido, então ele deve ser analisado e seguido, componente a componente, até que o i-node seja alcançado. Toda essa atividade pode exigir um número considerável de acessos adicionais ao disco.Há também outro problema introduzido pelas ligações, simbólicas ou não. Quando as ligações são permitidas, os arquivos podem ter dois ou mais caminhos. Programas que inicializam em um determinado diretório e encontram todos os arquivos naquele diretório e seus subdiretórios, localizaram um arquivo ligado múltiplas vezes.

22. Explique como as ligações estritas e as ligações flexíveis diferem em relação às alocações de i-nodes.

a. Quando uma nova entrada é feita na tabela de arquivos abertos, um ponteiro para o registro de cota do proprietário é atribuído a ela, a fim de facilitar encontrar os vários limites. Toda vez que um bloco é adicionado a um arquivo, o número total de blocos cobrados do proprietário é incrementado, e os limites flexíveis e estritos são verificados. O limite flexível pode ser excedido, mas o limite estrito não. Uma tentativa de adicionar blocos a um arquivo quando o limite de blocos estrito tiver sido alcançado resultará em um erro. Verificações análogas também existem para o número de arquivos a fim de evitar que algum usuário sobrecarregue todos os i-nodes.

23. Considere um disco de 4 TB que usa blocos de 4 KB e o método da lista de livres. Quantos endereços de blocos podem ser armazenados em um bloco?

a. Em cada bloco podem ser armazenados arquivos de no máximo 4 KB de acordo com o gerenciamento de blocos livres.

24. O espaço de disco livre pode ser monitorado usando-se uma lista de livres e um mapa de bips. Endereços de disco exigem D bits. Para um disco com B blocos, F dos quais estão disponíveis, estabeleça a condição na qual a lista de livres usa menos espaço do que o mapa de bits. Para D tendo um valor de 16 bits, expresse a resposta como uma percentagem do espaço de disco que precisa estar livre.
    a. Cabe ao usuário decidir qual arquivo colocar em qual unidade e monitorar o que está onde.

25. O começo de um mapa de bits de espaço livre fica assim após a partição de disco ter sido formatada pela primeira vez: 1000 0000 0000 0000 (o primeiro bloco é usado pelo diretório-raiz). O sistema sempre busca por blocos livres começando no bloco de número mais baixo, então após escrever o arquivo A, que usa seis blocos, o mapa de bits fica assim: 1111 1110 0000 0000. Mostre o mapa de bits após cada uma das ações a seguir:
    a. (a) O arquivo B é escrito usando cinco blocos.
       Depois escrito arquivo B: 1111 1111 1111 0000
    b. (b) O arquivo A é removido.
       Depois removido arquivo A: 1000 0001 1111 0000
    c. (c) O arquivo C é escrito usando oito blocos.
       Depois escrito arquivo C: 1111 1111 1111 1100
    d. (d) O arquivo B é removido.
       Depois removido arquivo B: 1111 1110 0000 1100

26. O que aconteceria se o mapa de bits ou a lista de livres contendo as informações sobre blocos de disco livres fossem perdidos por uma queda no computador? Existe alguma maneira de recuperar-se desse desastre ou é “adeus, disco”? Discuta suas respostas para os sistemas de arquivos UNIX e FAT-16 separadamente.
    a. O MS-DOS usa a FAT para monitorar blocos de disco livres. Qualquer bloco que no momento não esteja alocado é marcado com um código especial. Quando o MS-DOS precisa de um novo bloco de disco, ele pesquisa a FAT para uma entrada contendo esse código. Desse modo, não são necessários

nenhum mapa de bits ou lista de livres.

27. O trabalho noturno de Oliver Owl no centro de computadores da universidade é mudar as fitas usadas para os backups de dados durante a noite. Enquanto espera que cada fita termine, ele trabalha em sua tese que prova que as peças de Shakespeare foram escritas por visitantes extraterrestres. Seu processador de texto executa no sistema sendo copiado, pois esse é o único que eles têm. Há algum problema com esse arranjo?
    a. Sim, pois pode haver uma sobrescrita do conteúdo nas fitas. A forma mais simples de cópia incremental é realizar uma cópia (backup) completa periodicamente, digamos por semana ou por mês, e realizar uma cópia diária somente daqueles arquivos que foram modificados desde a última cópia completa. Melhor ainda é copiar apenas aqueles arquivos que foram modificados desde a última vez em que foram copiados. Embora esse esquema minimize o tempo de cópia, ele torna a recuperação mais complicada, pois primeiro a cópia mais recente deve ser restaurada e depois todas as cópias incrementais têm de ser restauradas na ordem inversa

28. .Discutimos como realizar cópias incrementais detalhadamente no texto. No Windows é fácil dizer quando copiar um arquivo, pois todo arquivo tem um bit de arquivamento. Esse bit não existe no UNIX. Como os programas de backup do UNIX sabem quais arquivos copiar?
    a. Uma cópia lógica começa em um ou mais diretórios especificados e recursivamente copia todos os arquivos e diretórios encontrados ali que foram modificados desde uma determinada data de base (por exemplo, o último backup para uma cópia incremental ou instalação de sistema para uma cópia completa). Assim, em uma cópia lógica, o disco da cópia recebe uma série de diretórios e arquivos cuidadosamente identificados, o que torna fácil restaurar um arquivo ou diretório específico mediante pedido. Tendo em vista que a cópia lógica é a forma mais usual, vamos examinar um algoritmo comum em detalhe usando o exemplo da Figura 4.25 para nos orientar. A maioria dos sistemas UNIX usa esse algoritmo.

29. Suponha que o arquivo 21 na Figura 4.25 não foi modificado desde a última cópia. De qual maneira os quatro mapas de bits da Figura 4.26 seriam diferentes?
    a. após a cópia é verificada a consistência dos arquivos,O programa então lê todos os i-nodes usando um dispositivo cru, que ignora a estrutura de arquivos e apenas retorna todos os blocos de disco começando em 0. A partir de um i-node, é possível construir uma lista de todos os números de blocos usados no arquivo correspondente. À medida que cada número de bloco é lido, seu contador na primeira tabela é incrementado. O programa então examina a lista de livres ou mapa de bits para encontrar todos os blocos que não estão sendo usados. Cada ocorrência de um bloco na lista de livres resulta em seu contador na segunda tabela sendo incrementado. Se o sistema de arquivos for consistente, cada bloco terá um 1 na primeira ou na segunda tabela, como ilustrado na Figura 4.27(a). No entanto, como consequência de uma queda no sistema, as tabelas podem ser parecidas com a Figura 4.27(b), na qual o bloco 2 não ocorre em nenhuma tabela. Ele será reportado como um bloco desaparecido

30. Foi sugerido que a primeira parte de cada arquivo UNIX fosse mantida no mesmo bloco de disco que o seu i-node. Qual a vantagem que isso traria?
    a. Quando o verificador tiver concluído, ele terá uma lista, indexada pelo número do i-node, dizendo quantos diretórios contém cada arquivo. Ele então compara esses números com as contagens de ligações armazenadas nos próprios i-nodes. Essas contagens começam em 1 quando um arquivo é criado e são incrementadas cada vez que uma ligação (estrita) é feita para o arquivo. Em um sistema de arquivos consistente, ambas as contagens concordarão. No entanto, dois tipos de erros podem ocorrer: a contagem de ligações no i-node pode ser alta demais ou baixa demais.

31. Considere a Figura 4.27. Seria possível que, para algum número de bloco em particular, os contadores em ambas as listas tivessem o valor 2? Como esse problema poderia ser corrigido?
    a. Quando uma contagem de i-node vai para zero, o sistema de arquivos marca como inutilizado e libera todos os seus blocos. Essa ação resultará em um dos diretórios agora apontando para um i-node não usado, cujos blocos logo

podem ser atribuídos a outros arquivos. Outra vez, a solução é apenas forçar a contagem de ligações no i-node a assumir o número real de entradas de diretório.

32. O desempenho de um sistema de arquivos depende da taxa de acertos da cache (fração de blocos encontrados na cache). Se for necessário 1 ms para satisfazer uma solicitação da cache, mas 40 ms para satisfazer uma solicitação se uma leitura de disco for necessária, dê uma fórmula para o tempo médio necessário para satisfazer uma solicitação se a taxa de acerto é h. Represente graficamente essa função para os valores de h variando de 0 a 1,0.
    a. O acesso ao disco é muito mais lento do que o acesso à memória. Ler uma palavra de 32 bits de memória pode levar 10 ns. A leitura de um disco rígido pode chegar a 100 MB/s, o que é quatro vezes mais lento por palavra de 32 bits, mas a isso têm de ser acrescentados 5-10 ms para buscar a trilha e então esperar pelo setor desejado para chegar sob a cabeça de leitura. Se apenas uma única palavra for necessária, o acesso à memória será da ordem de um milhão de vezes mais rápido que o acesso ao disco. Como consequência dessa diferença em tempo de acesso, muitos sistemas de arquivos foram projetados com várias otimizações para melhorar o desempenho. Nesta seção cobriremos três delas.

33. Para um disco rígido USB externo ligado a um computador, o que é mais adequado: uma cache de escrita direta ou uma cache de bloco?
    a. Cache de blocos, pois é uma coleção de blocos que logicamente pertencem ao disco, mas estão sendo mantidas na memória por razões de segurança. Vários algoritmos podem ser usados para gerenciar a cache, mas um comum é conferir todas as solicitações para ver se o bloco necessário está na cache. Se estiver, o pedido de leitura pode ser satisfeito sem acesso ao disco. Se o bloco não estiver, primeiro ele é lido na cache e então copiado para onde quer que seja necessário. Solicitações subsequentes para o mesmo bloco podem ser satisfeitas a partir da cache.

34. Considere uma aplicação em que os históricos dos estudantes são armazenados em um arquivo. A aplicação pega a identidade de um estudante como entrada e

subsequentemente lê, atualiza e escreve o histórico correspondente; isso é repetido até a aplicação desistir. A técnica de “leitura antecipada de bloco” seria útil aqui?

a. Sim, pois a técnica para melhorar o desempenho percebido do sistema de arquivos é tentar transferir blocos para a cache antes que eles sejam necessários para aumentar a taxa de acertos. Em particular, muitos arquivos são lidos sequencialmente. Quando se pede a um sistema de arquivos para obter o bloco k em um arquivo, ele o faz, mas quando termina, faz uma breve verificação na cache para ver se o bloco k + 1 já está ali. Se não estiver, ele programa uma leitura para o bloco k + 1 na esperança de que, quando ele for necessário, já terá chegado na cache. No mínimo, ele já estará a caminho.

35. Considere um disco que tem 10 blocos de dados começando do bloco 14 até o 23. Deixe 2 arquivos no disco: f1 e f2. A estrutura do diretório lista que os primeiros blocos de dados de f1 e f2 são respectivamente 22 e 16. Levando- -se em consideração as entradas de tabela FAT a seguir, quais são os blocos de dados designados para f1 e f2? f1(14,18); (15,17); f1(16,23); (17,21); f1(18,20); (19,15); f1(20, −1); (21, −1);f1 (22,19); (23,14). Nessa notação, (x, y) indicam que o valor armazenado na entrada de tabela x aponta para o bloco de dados y.

a. f1: (14,18); (16,23); (18,20);(20, −1) ; (22,19);
   f2: (15,17);(17,21); (19,15);(21, −1);(23,14).

36. . Considere a ideia por trás da Figura 4.21, mas agora para um disco com um tempo de busca médio de 6 ms, uma taxa rotacional de 15.000 rpm e 1.048.576 bytes por trilha. Quais são as taxas de dados para os tamanhos de blocos de 1 KB, 2 KB e 4 KB, respectivamente?

a. . Com arquivos de 4 KB e blocos de 1 KB, 2 KB ou 4 KB, os arquivos usam 4, 2 e 1 bloco, respectivamente, sem desperdício. Com um bloco de 8 KB e arquivos de 4 KB, a eficiência de espaço cai para 50% e com um bloco de 16 KB ela chega a 25%. Na realidade, poucos arquivos são um múltiplo exato do tamanho do bloco do disco, então algum espaço sempre é desperdiçado no último bloco de um arquivo. O que as curvas mostram, no entanto, é que o desempenho e a utilização de espaço estão inerentemente em conflito. Pequenos blocos são ruins para o desempenho, mas bons para a utilização do espaço do disco. Para esses dados, não há equilíbrio que seja razoável. O

tamanho mais próximo de onde as duas curvas se cruzam é 64 KB, mas a taxa de dados é de apenas 6,6 MB/s e a eficiência de espaço é de cerca de 7%, nenhum dos dois valores é muito bom. Historicamente, sistemas de arquivos escolheram tamanhos na faixa de 1 KB a 4 KB, mas com discos agora excedendo 1 TB, pode ser melhor aumentar o tamanho do bloco para 64 KB e aceitar o espaço de disco desperdiçado. Hoje é muito pouco provável que falte espaço de disco.

37. Um determinado sistema de arquivos usa blocos de disco de 4 KB. O tamanho de arquivo médio é 1 KB. Se todos os arquivos fossem exatamente de 1 KB, qual fração do espaço do disco seria desperdiçada? Você acredita que o desperdício para um sistema de arquivos real será mais alto do que esse número ou mais baixo do que ele? Explique sua resposta.
    a. 1KB/4KB = ¼ KB o desperdício para o sistema de arquivos real é mais alto que esse número. com um tamanho de bloco de 1 KB, apenas em torno de 30-50% de todos os arquivos cabem em um único bloco, enquanto com um bloco de 4 KB, a percentagem de arquivos que cabem em um bloco sobe para a faixa de 60-70%.

38. . Levando-se em conta um tamanho de bloco de 4 KB e um valor de endereço de ponteiro de disco de 4 bytes, qual é o maior tamanho de arquivo (em bytes) que pode ser acessado usando 10 endereços diretos e um bloco indireto?
    a. 64 KB a taxa de dados cresce quase linearmente com o tamanho do bloco (até as transferências demorarem tanto que o tempo de transferência começa a importar).

39. Arquivos no MS-DOS têm de competir por espaço na tabela FAT-16 na memória. Se um arquivo usa k entradas, isto é, k entradas que não estão disponíveis para qualquer outro arquivo, qual restrição isso ocasiona sobre o tamanho total de todos os arquivos combinados?
    a. No MS-DOS, os arquivos têm um nome base de 1-8 caracteres e uma extensão opcional de 1-3 caracteres. Na Versão 7 do UNIX, os nomes dos arquivos tinham 1-14 caracteres, incluindo quaisquer extensões. No entanto, quase todos os sistemas operacionais modernos aceitam nomes de arquivos

maiores e de tamanho variável.Para ler um arquivo, um programa de MS-DOS deve primeiro fazer uma chamada de sistema open para abri- -lo. A chamada de sistema open especifica um caminho, o qual pode ser absoluto ou relativo ao diretório de trabalho atual. O caminho é analisado componente a componente até que o diretório final seja localizado e lido na memória. Ele então é vasculhado para encontrar o arquivo a ser aberto. Embora os diretórios de MS-DOS tenham tamanhos variáveis, eles usam uma entrada de diretório de 32 bytes de tamanho fixo. O formato de uma entrada de diretório de MS-DOS é mostrado na Figura 4.30. Ele contém o nome do arquivo, atributos, data e horário de criação, bloco de partida e tamanho exato do arquivo. Nomes de arquivos mais curtos do que 8 + 3 caracteres são ajustados à esquerda e preenchidos com espaços à direita, separadamente em cada campo. O campo Atributos é novo e contém bits para indicar que um arquivo é somente de leitura, precisa ser arquivado, está escondido ou é um arquivo de sistema. Arquivos somente de leitura não podem ser escritos. Isso é para protegê-los de danos acidentais. O bit arquivado não tem uma função de sistema operacional real (isto é, o MS-DOS não o examina ou o altera). A intenção é permitir que programas de arquivos em nível de usuário o desliguem quando efetuarem o backup de um arquivo e que os outros programas o liguem quando modificarem um arquivo.

40. Um sistema de arquivos UNIX tem blocos de 4 KB e endereços de disco de 4 bytes. Qual é o tamanho de arquivo máximo se os i-nodes contêm 10 entradas diretas, e uma entrada indireta única, dupla e tripla cada?
    a. Uma entrada de diretório UNIX contém uma entrada para cada arquivo naquele diretório. Cada entrada é extremamente simples, pois o UNIX usa o esquema i-node ilustrado na Figura 4.13. Uma entrada de diretório contém apenas dois campos: o nome do arquivo (14 bytes) e o número do i-node para aquele arquivo (2 bytes), como mostrado na Figura 4.32. Esses parâmetros limitam o número de arquivos por sistema a 64 K. Assim como o i-node da Figura 4.13, o i-node do UNIX contém alguns atributos. Os atributos contêm o tamanho do arquivo, três horários (criação, último acesso e última modificação), proprietário, grupo, informação de proteção e uma contagem do número de entradas de diretórios que apontam para o i-node. Este último

campo é necessário para as ligações. Sempre que uma ligação nova é feita para um i-node, o contador no i- -node é incrementado. Quando uma ligação é removida, o contador é decrementado. Quando ele chega a 0, o i- -node é reivindicado e os blocos de disco são colocados de volta na lista de livres.

41. Quantas operações de disco são necessárias para buscar o i-node para um arquivo com o nome de caminho /usr/ ast/courses/os/handout.t? Presuma que o i-node para o diretório-raiz está na memória, mas nada mais ao longo do caminho está na memória. Também presuma que todo diretório caiba em um bloco de disco.
    a. Associar a cada arquivo uma estrutura de dados que relaciona os atributos e os endereços em disco dos blocos de arquivo.

42. Em muitos sistemas UNIX, os i-nodes são mantidos no início do disco. Um projeto alternativo é alocar um i- -node quando um arquivo é criado e colocar o i-node no começo do primeiro bloco do arquivo. Discuta os prós e contras dessa alternativa
    a. Quando um arquivo é aberto, o sistema deve tomar o nome do arquivo fornecido e localizar seus blocos de disco. Vamos considerar como o nome do caminho /usr/ ast/mbox é procurado. Usaremos o UNIX como exemplo, mas o algoritmo é basicamente o mesmo para todos os sistemas de diretórios hierárquicos. Primeiro, o sistema de arquivos localiza o diretório-raiz. No UNIX o seu i-node está localizado em um local fixo no disco. A partir desse i-node, ele localiza o diretório-raiz, que pode estar em qualquer lugar, mas digamos bloco 1. Em seguida, ele lê o diretório-raiz e procura o primeiro componente do caminho, usr, no diretório-raiz para encontrar o número do i-node do arquivo /usr. Localizar um i-node a partir desse número é algo direto, já que cada i-node tem uma localização fixa no disco. A partir desse i-node, o sistema localiza o diretório para /usr e pesquisa o componente seguinte, ast, nele. Quando encontrar a entrada para ast, ele terá o i-node para o diretório /usr/ast. A partir desse i-node ele pode fazer uma busca no próprio diretório e localizar mbox. O i-node para esse arquivo é então lido na memória e mantido ali até o arquivo ser fechado. O processo de busca está ilustrado na Figura 4.34. Nomes de caminhos relativos são procurados da mesma maneira que os absolutos, apenas partindo do diretório de trabalho em vez de do diretório-raiz. Todo diretório tem entradas para . e .. que são colocadas ali

quando o diretório é criado. A entrada . tem o número de i-node para o diretório atual, e a entrada para .. tem o número de i-node para o diretório pai. Desse modo, uma rotina procurando ../dick/prog.c apenas procura .. no diretório de trabalho, encontra o número de i-node do diretório pai e busca pelo diretório dick. Nenhum mecanismo especial é necessário para lidar com esses nomes. No que diz respeito ao sistema de diretórios, eles são apenas cadeias ASCII comuns, como qualquer outro nome. A única questão a ser observada aqui é que .. no diretório-raiz aponta para si mesmo.