CAP. 4

1) Como o MS-DOS implementa o acesso aleatório aos arquivos?

Por uma tabela de alocação de arquivos na memória principal. A entrada do diretório contém o número do primeiro bloco de arquivos. Este número é usado de índice para uma tabela FAT na memória principal, com a qual se acha o resto do arquivo, pois dentro desta tabela se guarda o endereço de memória de onde está outro índice de outra parte do arquivo, e esta guarda o endereço de outra e assim sucessivamente até chegar ao fim do arquivo, marcado geralmente por um caractere inválido.

2) A chamada de sistema *open* no Unix é essencial? Quais seriam as consequências de não tê-la?

Não, mas seria necessário especificar o nome do arquivo a cada chamada de leitura, e o sistema teria que buscar o *inode* responsável pelo arquivo. Alguns problemas podem decorrer disto (“quando retornar o *inode?”* e “como melhorar a performance ao carregá-lo?”), mas tudo é contornável, embora não tenham boa performance.

3) Sistemas que dão suporte a arquivos sequenciais têm uma operação para rebobiná-los. Os sistemas de acesso aleatório precisam disso também?

Não; se desejar acessar um arquivo novamente, basta acessar aleatoriamente o *byte* inicial (0.)

4) Alguns sistemas operacionais fornecem a chamada de sistema *rename*. Há diferença entre usar essa chamada ou copiar um arquivo com novo nome e então deletar o antigo?

Sim. *Rename* não altera atributos como a data de criação ou de última modificação. Copiar um novo e deletar o antigo, sim, pois estas informações tornam-se a data atual. Ademais, se o disco estiver quase cheio, pode ser que seja impossível copiar um arquivo.

5) Um certo sistema de arquivos usa blocos de disco de 4KB. O tamanho médio do arquivo é de 1KB. Se todos os arquivos forem de exatamente 1KB, qual a fração de espaço em disco que será desperdiçada? Você acha que o desperdício para um sistema de arquivos real será mais alto ou mais baixo que este? Justifique.

Como é impossível guardar mais de um arquivo no mesmo bloco, desperdiçar-se-ia (4-1)KB, ou seja, ¾ do tamanho do bloco. Um sistema real usa tanto arquivos grandes quanto pequenos, o que compensa o tamanho dos blocos, diminuindo bastante o desperdício.

6) Quantas operações em disco são necessárias para buscar o *inode* do arquivo /usr/ast/cursos/os/handout.t? Suponha que só o *inode* para o diretório raiz esteja na memória. Suponha que todos os diretórios caibam em um único bloco de disco.

É necessário operar sobre cada um dos diretórios, ou seja, cinco: raiz, usr, ast, cursos, os. Depois, é necessário pegar o *inode* para cada um destes diretórios e para o arquivo final, handout.t. Como o raiz está na memória, não é necessário buscá-lo em disco. Ao todo, são 5 *inodes* buscados. Logo, é necessário 5+5=10 operações.

7) Cite uma vantagem de *hard links* sobre *symbolic links*, e vice-versa.

Os *hard links* não exigem nenhum gasto extra de espaço no disco, apenas um contador no *inode* para poder saber quantos *hard links* há para aquele arquivo. *Links* simbólicos precisam de espaço para guardar o nome do arquivo para o qual eles apontam.

Os *links* simbólicos podem apontar para arquivos em outras máquinas, inclusive na internet. Já os *hard links* são limitados a apontar para arquivos na mesma partição em que estão.

8) 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 3 esquemas de alocação (contíguo, encadeado e indexado por tabela) será o mais apropriado?

Como o acesso é aleatório, tanto indexado por tabela quanto alocação contígua são apropriados. A encadeada não é recomendada porque precisa visitar todos os registros antes de chegar ao desejado.

CAP. 5

1) O MBR não influencia qual sistema operacional será carregado?

Falso. O gerenciador de *boot* encontra-se no MBR, e é ele o responsável por iniciar o devido sistema operacional a ser carregado.

1) Na alocação contígua dos arquivos, o número de blocos independe do tamanho do arquivo.

Falso. Para um tamanho de bloco X, e um tamanho de arquivo T, haverá um número de T/X blocos em disco.

1) Uma das funções de um S.O. é gerenciar quem pode acessar determinado arquivo.

Verdadeiro. Gerenciar o acesso a arquivos é parte do sistema de arquivos.

2) Um RAID nível 3 é capaz de corrigir erros de bit único usando somente um disco de paridade. Qual é o propósito do RAID nível 2? Afinal, também só pode corrigir um erro e gasta mais discos para fazê-lo.

O RAID 2 pode se recuperar não só de discos quebrados, mas também de erros transientes indetectados. Se um *driver* entregar até um *bit* ruim, o RAID nível 2 irá corrigi-lo; o de nível 3, não.

3) Quais são os itens que compõem um sistema de arquivos?

São arquivos e diretórios, e todo mais necessário para utilizá-los: as permissões de acesso, os tipos de arquivos, suas estruturas, nomenclaturas, atributos, operações, suas implementações físicas.

3) Qual a função principal de um sistema de diretórios?

Mapear o nome do arquivo em ASCII na informação necessária para a localização de dados.

3) Quais são as duas formas de armazenar atributos dos arquivos em um diretório?

Ou estes atributos são armazenados diretamente no próprio diretório, junto à lista de entradas; ou, caso sejam utilizados *inodes*, nestes.

5) Cite 5 funções dos drivers de dispositivos de E/S.

Aceitar solicitações abstratas de leitura e de escrita; inicializar o dispositivo; gerenciar necessidades de energia; registrar eventos; enfileirar requisições se o dispositivo estiver em uso.

5) Qual a principal vantagem de um driver ser implementado no espaço do usuário?

O fato de que as chamadas de sistema são feitas por rotinas de bibliotecas.

6) O que são *inodes*?

Os *inodes* são uma estrutura de dados que listam os atributos e os endereços de discos dos blocos do disco. Por meio de um *inode*, é possível encontrar todos os blocos do arquivo.

6) Uma das formas de implementação de arquivos é a alocação por lista encadeada, em que a primeira palavra de cada bloco é usada como ponteiro para o próximo bloco, e o restante do bloco para dados. Cite uma vantagem e uma desvantagem deste tipo de implementação.

Uma vantagem é que não há fragmentação senão a interna no último bloco. Uma desvantagem é que o acesso aleatório é muito lento, pois é necessário visitar sequencialmente todos os antecessores antes de chegar ao bloco desejado.

6) Cite uma vantagem da tabela de alocação de arquivos (FAT) sobre uma lista encadeada.

O acesso aleatório é facilitado, pois são colocadas as palavras do ponteiro de cada bloco de disco em uma tabela na memória, cujo fim é demarcado por um valor qualquer inválido. Assim, o encadeamento está inteiramente na memória, e pode ser seguido sem fazer referências ao disco.

4) As requisições de um disco chegam ao driver do disco na seguinte ordem dos cilindros: 10, 22, 20, 2, 40, 6 e 38. Um posicionamento leva 6ms por cilindro movido. Quanto tempo é necessário para cada um dos algoritmos (FCFS, SSF e elevador), considerando que o braço está, inicialmente, no cilindro 20? Dica: fazer o cálculo em número de cilindros e multiplicar por 6ms.

No FCFS (first-come, first-served, pega os cilindros à medida que são requisitados):

Como ele está no cilindro 20, ele precisa descer 10 cilindros ao 10; então 12 cilindros para ir do 10 ao 22; então 2 para ir do 22 ao 20; 18 do 20 para o 2; 38 do 2 para o 40; 34 do 40 para o 6; por fim, 32 do 6 para o 38. Ao todo, ele se move por 146 cilindros. Seria necessário, então, 146*6=876ms.

No SSF (shortest seek first, pega a primeira requisição; as demais que chegam, pega qual for a mais próxima):

Devido à natureza deste algoritmo, a ordem de visita aos cilindros será do 20 inicial para: 22-20-10-6-2-38-40. Visita-se, portanto, 60 cilindros. O tempo necessário é 60*6=360ms.

No elevador:

Este algoritmo trabalha com direções a partir de um ponto. Ou pode-se subir, ou descer. Se o elevador está subindo, ele continuará subindo (ou seja, acessando os cilindros depois do cilindro inicial) até que todas as requisições para cima tenham acabado; ele então desce (começa a acessar os cilindros abaixo do último cilindro visitado), e o processo se repete.

Por começar no 20º cilindro, ele subirá para: 22, 38 e 40. Então descerá para: 20, 10, 6 e 2. Ou seja, sai-se do 20, para: 22-38-40-20-10-6-2. Visita-se 58 cilindros, levando-se 58*6=348ms.

CAP. 6

1) Na prevenção de impasses, uma das opções é atacar a condição de posse e espera, tentando impedir que processos que detenham algum recurso possam esperar por outros recursos. Uma solução é fazer com que os processos aloquem todos os recursos dos quais precisem de uma só vez. Quais os dois problemas que existem em fazê-lo?

Muitos processos não sabem determinar a quantidade de recursos que eles usarão até que comecem a executar. Outro problema é que, assim, os recursos não serão usados de maneira otimizada.

2) Quais as quatro estratégias usadas para lidar com impasses?

A trajetória de recursos, estados seguros e inseguros, o algoritmo do banqueiro para um único recursos, e o algoritmo do banqueiro com múltiplos recursos.

3) Qual a estratégia usada para prevenir impasses?

Atacar: a exclusão mútua, usando *spool* em tudo; a posse e espera, requisitando todos os recursos necessários no início; a não-preempção, retomando os recursos alocados; a espera circular, ordenando numericamente os recursos.

4) Descreva uma das duas soluções para evitar a condição de espera circular?

Fornecer uma numeração global de todos os recursos. Processos podem solicitar recursos sempre que quiserem, mas todas as solicitações precisam ser feitas em ordem numérica.

6) Um sistema tem 2 processos e 3 recursos idênticos. Cada processo precisa de um máximo de 2 recursos. Um impasse é possível? Explique.

Não. Os dois processos alocam, cada um, seu primeiro recurso. O primeiro que precisar pegar o seu segundo, que é o terceiro do total de recursos disponíveis, pegará, e o outro aguardará por este. Como os recursos são idênticos, não faz sentido que um processo trave por querer o recurso do outro e vice-versa; quando o que tem dois recursos liberar um deles, o segundo processo continuará, só.

5) Um sistema tem 4 processos e 5 recursos alocáveis. A alocação atual e as necessidades máximas são conforme a imagem abaixo. Qual o menor valor de x para o qual esse estado é seguro?

	Alocado	Máximo	Disponível
Processo A	10211	11213	00*x*11
Processo B	20110	22210	
Processo C	11010	21310	
Processo D	11110	11221	

Neste caso, deve-se fazer uma matriz que é a subtração dos valores máximos pelos alocados, e então ver quais das linhas dessa matriz é inferior aos dos recursos que constam disponíveis.

```
      MATRIZ: MÁXIMO - ALOCADO                    RESULTADO

[(1-1)  (1-0)   (2-2)   (1-1)   (3-1)]          [0 1 0 0 2]
[(2-2)  (2-0)   (2-1)   (1-1)   (0-0)]          [0 2 1 0 0]
[(2-1)  (1-1)   (3-0)   (1-1)   (0-0)]          [1 0 3 0 0]
[(1-1)  (1-1)   (3-1)   (2-1)   (1-0)]          [0 0 2 1 1]
```

A única linha da matriz resultante que pode chegar próxima ao de recursos disponíveis é a segunda, para x=2. Pegamos então os valores alocados por essa última linha (a do processo D), somamos eles aos recursos disponíveis [0+1, 0+1, 2+1, 1+1, 1+0], e verificamos qual a próxima linha da tabela que podemos esvaziar. Esta linha é a do processo C, pois é a única que pode ser atendida pelo máximo de recursos disponíveis. Enfim, o processo repete-se até que todas as linhas tenham sido visitadas e todos os recursos sejam livres.

Caso x=2 não resolva o problema, deve-se tentar com x=3, x=4…, até chegar ao resultado. Neste caso, x=2 é a resposta correta. Parecerá que faltará um recurso (o da última coluna) para o processo A, mas, na verdade, como só há um processo rodando, então o sistema está em estado seguro, é impossível um impasse quando não há outro processo travando recurso.

PROVA 2 DADA EM AULA

2) c) As requisições de um disco chegam ao *driver* do disco na seguinte ordem dos cilindros: 10, 21, 20, 11, 42, 6 e 38. Um posicionamento leva 6ms por cilindro movido. Quanto tempo é necessário para cada um dos algoritmos abaixo, considerando que o braço está, inicialmente, no cilindro 20? Faça usando o algoritmo do elevador com ele se movendo apenas para cima.

Este algoritmo é semelhante ao abordado na questão 4 do capítulo 5. A única diferença é que ele começará do 20, e se movimentará somente em uma direção, ou seja, visitará: 20, 21, 38 e 42, e então irá para o 6, o 10 e o 11, como se fosse um círculo (é como se o 6 estivesse imediatamente após o 42.) Por conta disso, ele visitará 28 cilindros, logo, seu tempo será de 28*6=168ms.

3) Cite as desvantagens da implementação de arquivos por alocação por lista encadeada, e como podem ser eliminadas.

O acesso aleatório é extremamente lento, pois, começando de uma posição, ele vai ter que percorrer vários elementos da lista, sequencialmente, até chegar ao desejado. A quantidade de dados que um bloco armazena deixa de ser também uma potência de dois, porque parte do espaço é gasto com o ponteiro para o próximo endereço, o que confere, ao bloco, tamanhos peculiares, o que é menos eficiente, dado que programas costumam escrever em blocos cujo tamanho é em potência de dois.

Pode-se eliminar estes problemas colocando a palavra de cada ponteiro para o próximo elemento em uma tabela na memória (FAT.)

3) Qual a vantagem de I-NODES sobre arquivos encadeados que usam tabela na memória?

A vantagem é que o *i-node* precisa estar carregado na memória somente enquanto o arquivo estiver aberto.

4) Quais as duas soluções possíveis para resolver o problema de compartilhamento de arquivos? Descreva resumidamente como cada uma dela funciona.

Pode-se usar *i-nodes*. Neste caso, os blocos de disco não são listados em diretórios, mas sim a essa estrutura, que é associada com o arquivo em si.

Outra solução é criar um arquivo do tipo *link* (simbólico), que cria um arquivo do tipo *link* que aponta para o arquivo desejado. O sistema, ao ler este arquivo *link*, vê a qual arquivo ele se refere e então lê o arquivo referido.

6) Qual a principal característica de uma E/S programada e qual sua principal desvantagem?

O uso contínuo da CPU, que é, também, sua maior desvantagem. Este método toma a CPU por todo tempo enquanto a operação de E/S não tiver terminado.

6) Qual a principal diferença entre a E/S programada e a E/S usando DMA?

Teoricamente, ambas fazem E/S programada. A primeira ocupa a CPU, a outra, o DMA.

6) Qual a vantagem da E/S usando DMA sobre a E/S orientada à interrupção?

Interrupções são custosas para se lidar, demandando tempo. Com o DMA, a CPU fica livre do problema de ter que lidar com interrupções constantes.

7) Quais as duas funções básicas de um software de E/S independente de dispositivo?

Executar funções de E/S comuns para todos os dispositivos, e fornecer uma interface uniforme para o *software* em nível de usuário.

7) Qual a melhor maneira de eliminar os problemas causados por *drivers* defeituosos?

Isolar o núcleo do sistema dos *drivers*, colocando-os para rodar somente no espaço de usuário.