PROVA 03 – 2012/01 (Turma B)

Observação: as respostas abaixo salientam apenas os principais conceitos e não se considera que elas sejam a resposta padrão. Na resposta a prova se espera que essas ideias sejam elaboradas de acordo com o solicitado e, respostas diferentes, bem fundamentadas, são avaliadas de acordo com sua correção e argumentação.

1ª Questão

(a) Operação bloqueante: quando um processo realiza uma chamada bloqueante ele é retirado do estado running e passa para estado blocked até que a operação seja concretizada em sua totalidade. Por exemplo, ao solicitar para ler 50 dados só retornará (e desbloqueia o processo) quando tiver sido lido os 50 dados.
Operação não-bloqueante: o processo realiza a chamada e ela retorna imediatamente com a quantidade de dados disponíveis naquele momento. Por exemplo, ao solicitar para ler 50 dados pode se retornar com o total de dados requisitados, menos ou mesmo nenhum.
Operação assíncrona: o processo requisita uma transferência de dados na sua totalidade, mas, no momento da chamada, o processo não é bloqueado. A operação terminará em algum momento no futuro e o processo deve ser informado da sua conclusão por um mecanismo como signal (interrupção de software) ou callback. Exemplo, um processo solicita a leitura de 50 bytes, retorna imediatamente e, no futuro, quando os 50 bytes estiverem disponíveis o processo de usuário será avisado.

(b) Em grandes passos. Considerando que o diretório corrente é D3. É preciso abrir o arquivo de diretório D3, procurar pela entrada que corresponde ao arquivo abc.txt, remover essa entrada, armazená-la em uma variável temporária qualquer, ler o arquivo de diretório D3 procurando pela entrada que corresponde ao diretório pai (D2), abrir o arquivo de diretório D2, procurar pela entrada correspondente a D4, abrir o arquivo de diretório D4, inserir nesse arquivo a entrada correspondente ao arquivo abc.txt. Sendo exemplificado com UNIX, cada entrada tem um i-node que permite o acesso aos blocos dos arquivos diretórios.

2ª questão

(a) $t_{acesso} = (10 + 4 + 2^3 x 2^{11}/2^{20})*64/2 = 448{,}5$ ms; onde 10 é o valor do tempo de seek, 4 é latência rotacional média (8/2), 2^3 (8) corresponde a latência computacional e é o tempo de uma volta completa no disco, 2^{11} a quantidade de bytes a serem lidos; e 2^{20} é a quantidade de bytes de uma trilha. O tempo de transferência é uma proporção entre tamanho_bloco/ e tamanho_trilha. Depois é preciso ver quantos blocos do disco devem ser lidos considerando o tamanho do executável (64 KB) e o tamanho do bloco (2K).

(b) $t_{acesso} = (10 + 4 + 2^3 x 2^{12}/2^{20})*64/4 = 224{,}5$ ms; mesmo raciocínio do anterior.

(c) $t_{acesso} = (10 + 4 + 2^3 x 2^{16}/2^{20})*64/64 = 14{,}5$ ms; mesmo raciocínio do anterior

(d) Para um executável que tem sempre 64 KB, são necessários 32 leituras de blocos de 4KB que contém uma página de 2KB (64/2=32). Há uma fragmentação interna de 50%. Se a página fosse de 4 KB seriam necessários ler 16 blocos para carregar todo o executável (64 KB), ou seja, 64/4 acessos a bloco, ou seja, 16 acessos. Não há fragmentação interna. Com páginas de 4KB, o executável estaria disperso em 16 paginas (64/4), o que também dá 16 acessos. Aqui também não há fragmentação interna.
Outro ponto a considerar é o desempenho. Nessa situação, páginas de 2 KB, levam a 32 operações de acesso ao disco, ao passo que páginas de 4 KB e de 64 KB, na situação do enunciado levam a 16 acessos a disco, ou seja, o mesmo tempo.

3ª Questão

(a) Contígua: no diretório há uma entrada que associa o bloco lógico do arquivo (primeiro bloco é o zero) com um bloco físico que é corresponde ao primeiro bloco do arquivo. Como são contíguos, o bloco i lógico corresponde ao bloco_físico_inicial + i.
Encadeada: no diretório, cada entrada associa o primeiro bloco lógio do arquivo para um bloco físico qualquer, o próximo bloco lógico do arquivo corresponde ao bloco físico apontado pelo ponteiro de endereço dado na última entrada do bloco físico, e assim, sucessivamente.
Indexado: no diretório, a entrada corresponde a um bloco de dados que possui endereços dos blocos lógicos, portanto, a i-ésima entrada do bloco corresponde ao bloco lógico i. Se a capacidade de endereçamento for ultrapassada, a última entrada do bloco possui um endereço de próximo bloco que possui, novamente, uma entrada para cada bloco lógico. Portanto, dado um bloco x qualquer deve-se descobrir a qual bloco de índices ele pertence e obter a entrada correspondente.

Número de acesso para ler o bloco 4:
- contígua: 1 acesso (endereço do bloco inicial + 4) Ou 5 acessos (ler 0, 1, 2, 3, 4)
- encadeada: 5 acessos (ler 0,1,2,3 e 4). É preciso ler o 0 porque ele tem o ponteiro para o 1, o 1 porque tem o ponteiro para o 2, etc...
- indexada: 1 acesso (que corresponde ao bloco da entrada i=4)
OBS: como não foi explicitado na questão que a numeração dos blocos iniciava em zero (que é o lógico e o esperado) foi aceito as respostas que consideraram o inicio em 1, MAS, várias questões estavam incoerentes: a primeira parte considerava zero e a segunda parte considera 1 como primeiro bloco.

(b) Ao usar hardlink uma ou mais entradas em um diretório apontam para o mesmo descritor de arquivos (i-node no caso do Unix). Dessa forma, o arquivo original e seus hardlinks não são distinguíveis entre si. Já um softlink tem um descritor próprio (i-node) que informa qual é o path do arquivo que ele, softlink, aponta.

Ao se remover um hardlink deve-se ter o cuidado para não liberar os blocos do arquivo ele aponta porque podem haver outros hardlinks apontando para esse mesmo conjunto de blocos (arquivos). Para contornar esse problema, os sistemas operacionais usam um contador de referência no próprio descritor (i-node). Quando um arquivo é criado esse contador é posto em 1. Para cada hardlink criado ele é incrementado de uma unidade. Na remoção, decrementa-se o contador de referências. Se zero deve-se remover a entrada do diretório (hardlink) e liberar os blocos do arquivo, caso contrário, remove-se apenas o hardlink. Um softlink, por seu um arquivo próprio, pode ser removido sem os mesmo cuidados. Remover o softlink é a mesma coisa que remover um arquivo. Essa remoção não altera o arquivo original apontado pelo softlink. Se o arquivo original for removido, o softlink aponta para um path inválido.

Os hardlinks não podem ser usados entre partições, os sistemas operacionais impedem a criação de hardlinks para arquivos do tipo diretório, mas são mais eficientes que os softlinks já que não há distinção entre o arquivo original e o softlink. O softlink não tem as restrições do hardlink, mas tem um desempenho um pouco pior já que é necessário abrir um arquivo para recuperar a referência (path) do arquivo apontado.

4ª questão

(a) Como existem 2^{32} blocos e em um bit map é necessário 1 bit por bloco para indicar quais blocos estão livres e quais estão ocupados, é preciso 2^{32} bits. Como cada bloco tem 4096x8 bits (2^{15} bits), o número total de blocos ocupados pelo bitmap é 2^{17} blocos. Cada arquivo ocupado um i-node. O menor arquivo ocupa 1 bloco, portanto, teoricamente, é possível haver 2^{32} arquivos de 1 bloco, ou seja, é preciso, no pior caso, 2^{32} bits para representar i-nodes livres e ocupados. Também são necessários 2^{17} blocos.

(b) Um endereço de bloco possui 4 bytes, portanto, em um bloco se tem 1024 endereços. A capacidade de armazenamento direto corresponde a 10 blocos (40 bytes/4bytes), o que dá, 40 KB (10x40KB) que devem ser somados com a capacidade armazenamento indireto simples. Essa capacidade equivale UM endereço indireto, ou seja, aponta para um bloco. Esse bloco tem 1024 endereços de bloco, portanto, a capacidade indireta é 1024 x4 KB (2^{22} bytes). Maior arquivo Panoramix (direto+indireto) é: 40KB + 1024 x 4 KB.

(c) Um arquivo de diretório tem tamanho máximo de 40 KB (10x4096), cada entrada do diretório possui 256 bytes, portanto, cada arquivo de diretório possui 10x4096/256 entradas, ou seja, 160. Como cada entrada está associada a um arquivo, tem-se 160 arquivos.

(d) Logo após a formatação todos os blocos estão livres, então é preciso armazenar:
   a. Solução 1: endereço do primeiro bloco livre (4 bytes) e o número de blocos livres (4 bytes), ou seja, se ocupado apenas 1 bloco.
   b. Solução 2: necessário armazenar 2^{32} blocos, como cada endereço possui 4 bytes, são precisos 2^{36} bytes. Um bloco tem 2^{12} bytes, portanto, se precisa de 2^{22} blocos (desconsiderando os blocos necessários ao encadeamento – como dito no enunciado).

5ª questão

(a) Garantir a consistência de um sistema de arquivos através do registro, em um arquivo especial (log ou journal) das operações a serem feitas no sistema de arquivos. As operações são feitas baseadas em transações e, em caso de panes no sistema, a reexecução ou cancelamento dessas transações leva o sistema de arquivos a um estado consistente. A vantagem é desempenho, já que essa operação é baseada apenas no arquivo de log e não necessita varrer todo o sistema de arquivo como ocorre com os aplicativos do tipo fsck e chkdsk em sistemas de arquivos não jornalizados.

(b) Escalonamento de E/S consiste, basicamente, em reordenar as requisições de E/S provenientes dos processos de usuários para atendê-las em uma ordem que aumente o desempenho de acesso a um determinado dispositivo. No disco rígido, por exemplo, o objetivo é minimizar o tempo de seek e de latência rotacional; para isso existem vários algoritmos como o FCFS, C-LOOK, C-SCAN, etc.