Leonardo Vecchi Meirelles - 12011ECP002

Cap 36 - Questão 1)

O protocolo I/O canônico refere-se à abordagem tradicional usada para ler e gravar dados em dispositivos de armazenamento persistente, como discos. Nesse protocolo, o sistema operacional inicia uma operação de I/O e o aplicativo aguarda até que a operação seja concluída antes de continuar. Esse modelo de I/O síncrono garante a integridade dos dados, mas sofre problemas de desempenho, principalmente na presença de dispositivos de armazenamento lentos.

O problema de desempenho com o protocolo de I/O canônico é que ele introduz uma latência significativa e pode fazer com que o aplicativo pare enquanto aguarda a conclusão das operações de I/O. Essa natureza síncrona de I/O pode resultar em ciclos de CPU desperdiçados, pois o processador permanece ocioso durante as operações de I/O. Além disso, se vários solicitações de I/O forem emitidos sequencialmente, o desempenho geral de I/O poderá ser limitado pela operação de I/O mais lenta.

Leonardo Vecchi Meirelles - 12011ECP002

Cap 36 - Questão 2)

O método de interrupção resolve o problema de desempenho de busy waiting em operações de I/O. Em vez de a CPU pesquisar continuamente o dispositivo de I/O, o dispositivo envia um sinal de interrupção para a CPU quando a operação é concluída. Isso permite que a CPU execute outras tarefas enquanto aguarda a conclusão da operação de I/O, melhorando o desempenho geral do sistema e a utilização de recursos.

O método DMA resolve os problemas de desempenho associados à transferência de dados entre os dispositivos de I/O e a memória principal. Com o DMA, o dispositivo de I/O pode acessar diretamente a memória principal sem envolver a CPU, reduzindo o envolvimento da CPU e liberando seus recursos para outras tarefas. O DMA melhora o desempenho de I/O permitindo uma transferência de dados mais rápida e descarregando a tarefa de transferência de dados da CPU, resultando em eficiência geral do sistema

Leonardo Vecchi Meirelles - 12011ECP002

Cap 36 - Questão 3)

O protocolo básico para interagir com a interface IDE envolve a emissão de comandos e transferência de dados para se comunicar com dispositivos, como discos rígidos. O protocolo consiste em etapas como selecionar o dispositivo, especificar o comando e seus parâmetros, transferir dados entre o dispositivo e a memória e verificar o status da operação. Este protocolo utiliza o controlador IDE e seus registradores associados para controlar o fluxo de dados e coordenar a comunicação entre a CPU e o dispositivo. Ele permite ler e gravar dados de e para o dispositivo, permitindo armazenamento persistente e recuperação de informações.

Leonardo Vecchi Meirelles - 12011ECP002

Cap 37 - Questão 1)

No contexto de operações de I/O em uma unidade de disco rígido (HDD), existem três componentes de tempo principais:

- Tempo de busca (Seek time): é o tempo que o braço do disco leva para se posicionar sobre a trilha desejada. Envolve mover fisicamente o cabeçote de leitura/gravação para a posição correta do disco.
- Latência rotacional (Rotational latency): uma vez que o braço do disco é posicionado sobre a trilha desejada, a latência rotacional refere-se ao tempo que leva para o setor desejado girar sob o cabeçote de leitura/gravação. Depende da velocidade de rotação do disco, normalmente medida em RPM.
- Tempo de transferência (Transfer time): depois que o setor desejado é posicionado sob o cabeçote de leitura/gravação, o tempo de transferência é o tempo necessário para ler ou gravar os dados de ou para o disco. Depende de fatores como a taxa de transferência de dados, o tamanho dos dados sendo transferidos e a eficiência do sistema de I/O.

Leonardo Vecchi Meirelles - 12011ECP002

Cap 37 - Questão 2)

O algoritmo Shortest Seek Time First (SSTF) é um algoritmo de escalonamento de disco que visa minimizar o seek time, que é o tempo que o braço do disco leva para se mover para a trilha desejada.

No SSTF, a próxima requisição a ser atendida é escolhida com base na menor distância da posição atual do braço do disco. Seleciona a requisição que requer o menor movimento do braço, independente da ordem de recebimento das requisições. Isso garante que o tempo de busca seja minimizado, pois o braço do disco está sempre se movimentando para a solicitação mais próxima.

Ao selecionar continuamente a solicitação mais próxima, o algoritmo SSTF pode reduzir o tempo médio de busca em comparação com outros algoritmos de agendamento. No entanto, isso pode resultar na privação de algumas solicitações localizadas mais longe da posição atual, levando a possíveis problemas de imparcialidade (fairness).

Leonardo Vecchi Meirelles - J20JJECP002

Cap 37 - Questão 3)

O algoritmo SCAN (Elevator) e o algoritmo C-SCAN (Circular SCAN) são algoritmos de escalonamento de disco usados para otimizar o movimento do braço do disco e reduzir o tempo de busca.

No algoritmo SCAN, o braço do disco começa em uma extremidade do disco e se move em direção à outra extremidade, atendendo às solicitações ao longo do caminho. Ao chegar ao final, ele inverte a direção e volta, atendendo às solicitações na direção oposta. Este movimento imita o de um elevador, daí o nome.

O algoritmo C-SCAN é semelhante ao SCAN, mas fornece uma distribuição mais uniforme do tempo de serviço. Em vez de inverter a direção na extremidade do disco, ele salta para a extremidade oposta, criando efetivamente um caminho circular. Isso garante que as solicitações do outro lado do disco também sejam atendidos, evitando starving e fornecendo fairness.

Leonardo Vecchi Meirelles - 32011ECP002

Cap 37 - Questão 4)

Quando chega uma nova solicitação, o algoritmo Shortest Position to Target First (SPTF), seleciona a solicitação com a menor distância até a posição atual do braço do disco. Ele funciona verificando continuamente a fila de solicitações pendentes e selecionando a solicitação que requer o menor movimento do braço de disco para reduzir o tempo de busca. O algoritmo varre na direção das requisições pendentes e inverte a direção quando não há mais requisições naquela direção. O SPTF minimiza o tempo médio de busca e melhora o desempenho geral do disco. No entanto, pode levar a privação de solicitações (starvation) localizados mais longe da posição atual.

Leonardo Vecchi Meirelles - 120 11 ECP002

Cap 39 - Questão 1)

Quando um processo pai invoca a chamada de sistema fork(), o processo filho é criado como uma cópia do processo pai. Isso inclui duplicar os file descriptors do pai no processo filho. Portanto, os file descriptors abertos no pai também serão abertos no filho após a chamada fork(). Tanto o pai quanto o filho terão file descriptors tables independentes, mas farão referência aos mesmos arquivos abertos. Quaisquer modificações ou operações executados nos file descriptors no pai ou no filho afetarão os arquivos abertos compartilhados de acordo.

Leonardo Vecchi Meirelles - 12011ECP002

Cap 39 - Questão 2)

A principal diferença entre um hard link e um symbolic link é que um hard link é uma referência direta à localização física de um arquivo no sistema de arquivos, enquanto um symbolic link é um arquivo especial que contém o caminho para outro arquivo ou diretório

Um hard link cria vários entrados de diretório apontando para o mesmo arquivo físico e todos os hard links são tratados igualmente.

Um symbolic link, também conhecido como soft link, é um arquivo separado que atua como um ponteiro ou atalho para outro arquivo ou diretório e pode apontar para arquivos ou diretórios em diferentes sistemas de arquivos.

Se o arquivo original for movido ou renomeado, um hard link ainda apontará para o arquivo, enquanto um symbolic link será interrompido se o arquivo ou diretório de destino for movido ou renomeado.

Leonardo Vecchi Meirelles - 1201IECP002

Cap 39 - Questão 3)

Em sistemas do tipo Unix, existem três bits de permissão: read (r), write (w) e execute (x). Esses bits de permissão se aplicam a três entidades: ao proprietário do arquivo (owner), o grupo associado ao arquivo (group) e outros (others).

Os significados dos bits de permissão são os seguintes:

- Read (r): permite que a entidade leia o conteúdo do arquivo ou liste o conteúdo de um diretório.
- Write (w): permite que a entidade modifique o conteúdo do arquivo ou exclua o arquivo. Para diretórios, concede a capacidade de adicionar, remover ou renomear arquivos dentro do diretório.
- Execute (x): para arquivos, permite que a entidade execute o arquivo como um programa ou script. Para diretórios, concede permissão para acessar arquivos e diretórios dentro dele.

Os bits de permissão podem ser representados numericamente da seguinte forma: read (4), write (2) e execute (1). Os valores numéricos podem ser combinados para representar vários combinações de permissões para o owner, group e others.