Guilherme Vasco da Silva : 1603019

Henrique Moura Bini : 2046326

Juan Felipe da Silva Rangel :2046385

Entrada/Saída

Lista - 08

Campo mourão 03/2021

1. O sistema operacional exige de todos os drivers de dispositivo a mesma interface padrão para que o restante do SO não precise ter consciência das peculiaridades de cada dispositivo, mas possa tratá-lo por famílias ou classes. Não seria mais apropriado deixar cada driver de dispositivo definir as rotinas de interface que fazem sentido para aquele tipo específico de dispositivo, pois isso complicaria o trabalho do SO, tendo que tratar cada dispositivo de forma diferente, não sendo possível tratar todos os dispositivos de uma mesma forma genérica, complicando também a construção de aplicações e das camadas mais elevadas do próprio sistema operacional, pois, nesse caso, impossibilitando agrupá-los em classes ou famílias com características semelhantes.

2. Os três componentes que formam o tempo de acesso ao disco são: tempo de busca, latência rotacional e tempo de transferência.

3. No mapeamento em espaço de memória o módulo de entrada/saída opera dentro de um espaço de endereçamento da memória, utilizando apenas um conjunto de endereços reservados, ocasionando o processador a tratar os registradores de status e dados do módulo de entrada e saída como qualquer outra posição de memória, utilizando operações comuns de leitura e escrita.

Esse tipo de mapeamento oferece uma maior simplicidade e disponibilidade de um conjunto maior de operações (instruções), mesmo sendo reduzido em espaço de memória, por causa da reserva de endereços para portas de I/O.

Já no mapeamento em espaço de entrada e saída, existe um espaço de endereçamento independente para os dispositivos de entrada/saída. Neste caso as portas de I/O são acessíveis apenas por operações especiais de entrada/saída.

Diferentemente, nesse tipo de mapeamento temos uma maior segurança nas operações que envolvem memória ou entrada/saída, e também um maior espaço de endereçamento, devido a organização um mais complexa e um número menor de instruções .

4.

* E/S Programada: A vantagem dessa técnica é que, entre todas as estratégias de entrada/saída, essa é a mais simples. Porém ela também apresenta algumas desvantagens, nesse estratégia, o controlador pode ficar esperando por novos dados, pois ele só precisa trabalhar de verdade quando há dados para serem processados, o que parece uma vantagem, entretanto, manter o processador esperando até que a operação seja concluída não é exatamente algo que nós desejamos, principalmente se a operação que for solicitada demore muito tempo para ser terminada, gerando uma espera ocupada. Além disso, podemos ser prejudicados na execução de outras atividades importantes do sistema, considerando que o processador estará esperando pela resposta de um controlador lento.
* Interrupção: A vantagem apresentada nesta técnica é que ela resolve o problema de espera ocupada que tínhamos anteriormente, utilizando as interrupções, ou seja, quando o processador envia um comando para o módulo de entrada/saída, o mesmo consegue começar a executar uma tarefa diferente, sem ter a necessidade de ficar monitorando o módulo que acabou de ser adicionado.
* Acesso direto a memória: A vantagem de utilizar o acesso direto a memória é que ele permite transferências diretas entre a memória principal e os controladores de entrada/saída, melhorando o uso do processador, já que agora não é preciso fazer transferências adicionais, pois nessa estratégia ele não é usado para intermediar o tratamento de operações de entrada/saída. Uma outra vantagem é que o processador fica livre após acessar as portas do controlador associado ao dispositivo desejado, deixando que o controlador DMA e o controlador do disco se encarregue da transferência dos dados.

Uma desvantagem dessa estratégia é que ela não se dá tão bem com transferências pequenas de dados, já que a configuração de cada operação do DMA é complexo, sendo mais simples e rápido utilizar o processador principal. Podemos concluir então que, se o seu objetivo é lidar com operações de entrada/saída envolvendo dispositivos que produzem grandes volumes de dados, utilizar o acesso direto à memória é uma boa escolha.

5. Cada dispositivo de E/S ligado a um computador precisa de algum código específico do dispositivo para controlá-lo. Esse código, chamado driver do dispositivo, geralmente é escrito pelo fabricante do dispositivo e fornecido junto com ele. Tendo em vista que cada sistema operacional precisa dos seus próprios drivers, os fabricantes de dispositivos comumente fornecem drivers para vários sistemas operacionais populares. Com o advento dos computadores pessoais, com sua miríade de dispositivos de E/S, um modelo onde é necessário recompilar o núcleo com cada driver novo é inviável. Poucos usuários são capazes de recompilar ou religar o núcleo, mesmo que eles tenham o código-fonte ou módulos-objeto, o que nem sempre é o caso. Em vez disso, sistemas operacionais, começando com o MS-DOS, se converteram em um modelo no qual os drivers são dinamicamente carregados no sistema durante a execução. Sistemas diferentes lidam com o carregamento de drivers de maneiras diferentes.

6. a - Drivers do dispositivo

b - Drivers do dispositivo

c - Software do sistema operacional independente do dispositivo

d - Software de E/S no nível do usuário

7. Durante o spooling é criado um processo especial, chamado daemon, e um diretório especial, chamado de “diretório de spooling”. Para imprimir um arquivo, um processo primeiro gera o arquivo inteiro a ser impresso e o coloca no diretório de spooling. Cabe ao daemon, que é o único processo com permissão de usar o arquivo especial da impressora,

imprimir os arquivos no diretório. Ao proteger o arquivo especial contra o uso direto pelos usuários, o problema de ter alguém mantendo-o aberto por um tempo desnecessariamente longo é eliminado. Isso evita, por exemplo, que um processo amarre a impressora, o que resultaria em alguns caracteres serem impressos e depois dormir por horas.

8. **FCFS**

| Posição Inicial | Posição final | Blocos percorridos |
| --- | --- | --- |
| 50 | 95 | 45 |
| 95 | 164 | 69 |
| 164 | 36 | 128 |
| 36 | 68 | 32 |
| 68 | 17 | 51 |
| 17 | 115 | 98 |

| Total de blocos percorridos | 423 |
| --- | --- |

**SSTF**

| Posição Inicial | Posição final | Blocos percorridos |
| --- | --- | --- |
| 50 | 36 | 14 |
| 36 | 17 | 19 |
| 17 | 68 | 51 |
| 68 | 95 | 27 |
| 95 | 115 | 20 |
| 115 | 164 | 49 |

| Total de blocos percorridos | 180 |
| --- | --- |

**SCAN**

| Posição Inicial | Posição final | Blocos percorridos |
| --- | --- | --- |
| 50 | 68 | 18 |
| 68 | 95 | 27 |
| 95 | 115 | 20 |
| 115 | 164 | 49 |
| 164 | 199 | 35 |
| 199 | 36 | 167 |
| 36 | 17 | 19 |

| Total de blocos percorridos | 335 |
| --- | --- |

**C-SCAN**

| Posição Inicial | Posição final | Blocos percorridos |
| --- | --- | --- |
| 50 | 68 | 18 |
| 68 | 95 | 27 |
| 95 | 115 | 20 |
| 115 | 164 | 49 |
| 164 | 199 | 35 |
| 199 | 0 | 199 |
| 0 | 17 | 17 |
| 17 | 36 | 19 |

| Total de blocos percorridos | 384 |
| --- | --- |

**LOOK**

| Posição Inicial | Posição final | Blocos percorridos |
| --- | --- | --- |
| 50 | 68 | 18 |
| 68 | 95 | 27 |
| 95 | 115 | 20 |
| 115 | 164 | 49 |
| 164 | 36 | 128 |
| 36 | 17 | 19 |

| Total de blocos percorridos | 261 |
| --- | --- |

**C-LOOK**

| Posição Inicial | Posição final | Blocos percorridos |
| --- | --- | --- |
| 50 | 68 | 18 |
| 68 | 95 | 27 |
| 95 | 115 | 20 |
| 115 | 164 | 49 |
| 164 | 17 | 147 |
| 17 | 36 | 19 |

| Total de blocos percorridos | 280 |
| --- | --- |

9.

| Estratégia | Velocidade de leitura | Velocidade de escrita | Espaço | Falhas |
| --- | --- | --- | --- | --- |
| RAID 0 (striping) | até 4 | até 4 | 32 TBytes | 0 |
| RAID 1 | até 4 | 1 (requer atualizar todas as cópias) | 8 TBytes | 3 |
| RAID 5 | até 4 (acessa discos distintos em paralelo) | 1 (lê blocos de todos os discos para calcular e atualizar bloco de paridade) | 24 TBytes | 1 |

10. O escalonamento de disco é necessário pois ele reduz o tempo gasto na busca de registros através da criação de uma fila de requisições. Após isso ele reorganiza a fila de modo que ele consiga atendê-las da forma mais eficiente possível e com o mínimo de movimento mecânico.

11. O uso de cache é importante para a gerência de entrada e saída, já que os discos são dispositivos lentos, causando operações de leitura e escrita terem latências altas, e usando a cache é possível melhorar esse desempenho, mantendo o conteúdo de blocos lidos ou escritos, para que seja possível consultá-los em acessos posteriores. Por exemplo, poderíamos fazer um caching de leitura, começando por ler um bloco ou vários blocos na memória, e em seguida, armazenar esses blocos lidos anteriormente do disco, na memória cache, com intuito de otimizar as próximas leituras.

Outra questão para o software de E/S é a utilização de buffer. Muitas vezes, dados provenientes de um dispositivo não podem ser armazenados diretamente em seu destino final. Por exemplo, quando um pacote chega da rede, o sistema operacional não sabe onde armazená-lo definitivamente até que o tenha colocado em algum lugar para examiná-lo. Também, alguns dispositivos têm severas restrições de tempo real (por exemplo, dispositivos de áudio digitais), portanto os dados devem ser colocados antecipadamente em um buffer de saída para separar a taxa na qual o buffer é preenchido da taxa na qual ele é esvaziado, a fim de evitar seu completo esvaziamento. A utilização do buffer envolve consideráveis operações de cópia e muitas vezes tem um impacto importante sobre o desempenho de E/S.

Em uma E/S programada, por exemplo, consideremos um processo de usuário que quer imprimir a cadeia de oito caracteres “ABCDEFGH” na impressora por meio de uma interface serial. O processo do usuário requisita a impressora para escrita fazendo uma chamada de sistema para abri-la. Se a impressora estiver atualmente em uso por outro processo, a chamada fracassará e retornará um código de erro ou bloqueará até que a impressora esteja disponível, dependendo do sistema operacional e dos parâmetros da chamada. Uma vez que ele tenha a impressora, o processo do usuário faz uma chamada de sistema dizendo ao sistema operacional para imprimir a cadeia de caracteres na impressora.

O sistema operacional então (normalmente) copia o buffer com a cadeia de caracteres para um vetor no espaço do núcleo, onde ele é mais facilmente acessado. Ele então confere para ver se a impressora está disponível no momento. Se não estiver, ele espera até que ela esteja. Tão logo a impressora esteja disponível, o sistema operacional copia o primeiro caractere para o registrador de dados da impressora, nesse exemplo usando a E/S mapeada na memória. Essa ação ativa a impressora.

Tão logo copiado o primeiro caractere para a impressora, o sistema operacional verifica se ela está pronta para aceitar outro. Geralmente, a impressora tem um segundo registrador, que contém seu estado. O ato de escrever para o registrador de dados faz que o estado se torne "indisponível". Quando o controlador da impressora tiver processado o caractere atual, ele indica a sua disponibilidade marcando algum bit em seu registrador de status ou colocando algum valor nele.

Nesse ponto, o sistema operacional espera que a impressora fique pronta de novo. Quando isso acontece, ele imprime o caractere seguinte. Esse laço continua até que a cadeia inteira tenha sido impressa. Então o controle retorna para o processo do usuário.

12. I/O com bloqueio deve ser usado caso o usuário deseje realizar a impressão de vários arquivos em uma única impressora. Também deve ser utilizado quando o usuário deseja fazer alguma alteração no disco rígido. Outro caso em que se deve utilizar I/O com bloqueio é a alteração nos dados de um dispositivo de armazenamento conectado a uma porta USB.

I/O sem bloqueio deve ser usado caso haja uma aplicação de interface para o usuário onde ao receber entradas do teclado ou do mouse e exibe dados na tela. Outro caso seria em uma aplicação de vídeo que lê quadros de um arquivo em disco enquanto, simultaneamente, descomprime e exibe a saída na tela. Outro caso em que se deve utilizar I/O com bloqueio são a que o próprio kernel e aplicações que não devem ser suspensas enquanto esperam que uma operação I/O seja concluída.

Não se deve implementar apenas I/O sem bloqueio pois isso pode gerar um sobrecarregamento da CPU com operações de I/O impedindo que a CPU execute outros processos que não sejam I/O.

Patos voam!