

Entrada e Saída (I/O)

Princípios básicos de hardware

É necessário que algum tipo de mecanismo exista para que possamos informar problemas ao computador e recuperar uma sua solução. Esse mecanismo constitui o que chamamos genericamente de dispositivos de Entrada e Saída (I/O).

De acordo com o sentido do fluxo de dados entre o computador e o dispositivo, esses podem ser divididos em periféricos de entrada, periféricos de saída, ou ainda periféricos de entrada e saída. Um periférico pode ser visto como qualquer dispositivo conectado a um computador de forma a possibilitar sua interação com o mundo externo.

Os periféricos são conectados ao computador através de um componente de hardware denominado interface. Isso significa que os periféricos não estão conectados diretamente aos barramentos do computador. Dessa forma, as interfaces constituem um elemento chave para coordenação da transferência de dados entre periférico e o processador, ou entre periférico e memória. As interfaces empregam no seu projeto um outro componente de hardware: o **controlador**. Um controlador nada mais é que um processador projetado especificamente para realizar uma função, como, por exemplo, controlar um disco rígido.

Os dispositivos de I/O, dependendo de sua interconexão física às interfaces, podem ser do tipo serial ou paralelo. Essa característica está relacionada à maneira pela qual os dados são transferidos entre os dispositivos de I/O e as interfaces. Esse mecanismo é baseado em um protocolo de transmissão constituído por um conjunto de linhas de controle e linhas específicas para o envio e recepção de dados. Uma interface serial é aquela que existem apenas uma linha para os dados. Nesse caso, podemos imaginar o protocolo de transferência como uma forma de transformar um byte em uma seqüência de bits e vice-versa. Os modems, alguns tipos de mouse e impressoras são exemplos de dispositivos seriais. Uma interface paralela possui várias linhas para os dados, permitindo assim que vários bits sejam transferidos simultaneamente (em paralelo) entre os dispositivos de I/O e a interface. O número de linhas corresponde normalmente ao número de bits que compõem um byte (8 bits), ou uma palavra ($n \times 8$ bits). As impressoras paralelas são exemplos típicos desse tipo de dispositivo.

Mapeamento em espaço de memória e em espaço de entrada e saída

O processador "enxerga" o controlador através de um conjunto de registradores internos ao controlador, os quais, entre outras tarefas, recebem ordens do processador, fornecem o estado de uma operação, ou permitem a leitura (escrita) de dados do periférico. Esses

registradores residem fisicamente no hardware do controlador e são acessados pelo processador através do barramento de dados e de endereços. Para que esse mecanismo seja possível os registradores internos do controlador devem estar associados a posições da memória. Então, acessar um desses controladores consiste basicamente em realizar uma operação de escrita em memória, fornecendo o endereço correspondente a esse registrador e a ordem a ser executada como valor a ser escrito.

Basicamente existem duas formas para a interpretação de um endereço, dependendo do tipo do processador e de como o sistema como um todo foi projetado à nível de hardware: entrada e saída **mapeada em memória** e entrada e saída **mapeada em espaço de entrada e saída**.

O mapeamento de entrada e saída em memória corresponde a definir, no momento do projeto do computador, uma zona de endereçamento de memória na qual não existirá fisicamente uma memória, mas sim dispositivos de I/O. Dessa forma, o espaço de endereçamento do computador está dividido em dois, um no qual cada endereço corresponde uma posição de uma memória, e outro no qual os endereços (posições de memória) corresponderão a registradores internos de controladores de diferentes dispositivos.

Já o mapeamento em espaço de entrada e saída ocorre em um grande número de processadores, que possuem instruções especiais para a manipulação de entrada e saída. Nesse tipo de processador o que ocorre, na realidade, é que, no projeto do próprio processador (não do computador), são definidos dois espaços de endereçamento completamente distintos: Um espaço de endereçamento "normal" e outro dedicado a entrada e saída (I/O mapped). A distinção entre os dois é feita pelo emprego de instruções, ou seja, existe um conjunto de instruções para acesso da memória normal e um conjunto de instruções para acesso a memória de entrada e saída.

I/O programada

Na técnica de acesso a dispositivos de entrada e saída programado, toda interação entre o processador e o controlador é de responsabilidade exclusiva do programador. O ciclo de funcionamento é baseado no envio de um comando ao controlador e na espera de uma realização. Por exemplo, o processador envia um comando de leitura ao controlador e, em seguida, pode ler continuamente o registrador de estado para verificar se o dado solicitado já está disponível; em caso afirmativo, o processador efetua a leitura.

O problema desse método é que as operações de I/O são muito lentas se comparando com operações de cálculo. Utilizar continuamente o processador para verificar o andamento de uma operação de entrada e saída representa um desperdício muito grande de tempo de cálculo. Uma solução paliativa é inserir operações de cálculo entre verificações sucessivas do estado da operação de entrada e saída. Esse procedimento de verificar periodicamente o estado de uma operação de entrada e saída é denominado **polling**.

O emprego do polling reduz o problema desperdício de tempo do processador, mas introduz um outro problema: determinar a frequência de sua realização. Se a frequência de polling for muito alta, pode significar desperdício de tempo, principalmente se o dispositivo for realmente lento. Por outro lado, efetuar o polling com uma frequência baixa pode acarretar esperas desnecessárias por parte do dispositivo de entrada e saída, ou ainda, perda de informações. O ideal seria que o dispositivo I/O "chamasse" a atenção do processador logo que o dado estivesse disponível. Isso é possível graças ao mecanismo de interrupções.

Mecanismo de Interrupções

É o mecanismo baseado na geração de um sinal ao processador sempre que ocorre um determinado evento externo ao processador. Ao receber esse sinal, o processador pára momentaneamente o que está fazendo para executar uma rotina interrompida sem que ela perceba essa interrupção.

O emprego de interrupções implica numa série de detalhes tanto de software quanto de hardware. Para isso temos o controlador de interrupções. Entre suas funções estão: (a) identificar a fonte de interrupção a fim de executar, para cada uma delas, um procedimento específico. (b) privilegiar o atendimento de uma interrupção em relação a outra (prioridades). Ou ainda, (c) selecionar quais interrupções serão atendidas (mascaramento).

As interrupções podem ser ativadas por hardware ou software. Como já dissemos, no processo de hardware um evento externo faz com que um sinal seja gerado ao processador. O processador para a

atividade em andamento para realizar uma rotina específica(o tratador) e depois retorna à sua posição original.

Já uma interrupção de software é provocada pela execução de uma instrução específica.

Há ainda uma terceira interrupção, que é a interrupção interna do processador, que é gerada quando ocorre um erro interno no processador.

Acesso direto à Memória

Para a transferência de dados, uma solução é transferir diretamente os dados da interface (controlador) para a memória. Esse mecanismo é conhecido como **acesso direto à memória** (DMA – Direct Memory Access).

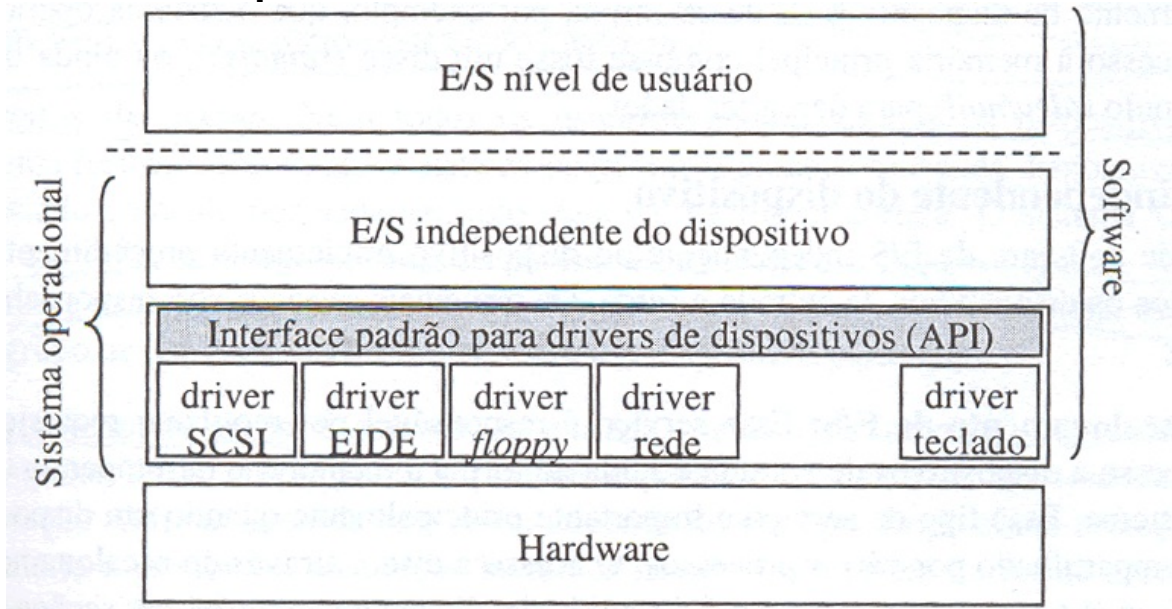
A técnica de DMA baseia-se no emprego de um hardware especial, o controlador de DMA, para realizar a transferência de dados entre um dispositivo de entrada e saída e a memória. O controlador de DMA possui internamente uma série de registradores utilizados pelo processador para programar a transferência de dados.

Para que ocorra uma transferência por DMA é necessário que, em primeiro lugar, o processador inicialize o controlador de DMA fornecendo informações como quantidade de dados a transferir (tamanho do bloco) a origem e o destino desses blocos, e ainda qual o sentido da transferência, se memória para dispositivo ou vice-versa. Após essa programação, o processador dispara a execução de DMA, iniciando a transferência. Enquanto o controlador de DMA efetua a transferência, o processador pode dedicar-se a outra tarefa. Ao término da transferência, o controlador de DMA sinaliza o processador através de uma interrupção de hardware.

Princípios Básicos de I/O

O objetivo primeiro do subsistema de entrada e saída é tentar padronizar ao máximo as rotinas de acesso aos periféricos de forma a reduzir o número de rotinas de entrada e saída. Tal característica permite, ao mesmo tempo, a inclusão de novos dispositivos sem necessidade de alterar a interface de programação com o usuário final. Para atingir esse objetivo, o subsistema de entrada e saída é normalmente organizado em uma estrutura de 4 camadas, onde a camada i fornece uma abstração de mais alto nível à camada $i+1$ das funções de entrada e saída. Essa abstração é obtida através de uma interface de programação padrão que engloba uma série de operações comuns e necessárias a todos os dispositivos.

Drivers de dispositivo



A camada drivers de dispositivo (device drivers) é composta por um conjunto de módulos de software cada um implementado para fornecer os mecanismos de acesso a um dispositivo de entrada e saída específico. O principal objetivo dos drivers de dispositivos é um dispositivo de entrada e saída específico. O objetivo principal dos drivers de dispositivos é "esconder" as diferenças entre os vários dispositivos de entrada e saída, fornecendo à camada superior uma "visão uniforme" desses dispositivos através de uma interface de programação única. Ela é responsável por implementar as rotinas necessárias ao acesso e à gerência de um dispositivo específico. É nesse nível que o software de I/O realiza a programação de registradores internos de controladores que compõem a interface física dos dispositivos e implementa os respectivos tratadores de interrupção. Assim, cada tipo de dispositivo requer um driver apropriado. Essa camada fornece uma abstração a mais genérica possível para a camada superior, a de I/O independente do dispositivo.

Para atender o requisito de fornecer uma "visão uniforme", os dispositivos de entrada e saída são classificados em: **orientados a bloco** e **orientados a caractere**.

Em um **dispositivo orientado a bloco** (*block devices*), o armazenamento de informações e sua transferência entre o periférico e o sistema são realizados através de blocos de dados de tamanho fixo. As unidades de disco são o exemplo mais comum de dispositivos orientados a bloco.

Os **dispositivos orientados a caractere** (*character devices*), realizam transferências byte a byte, a partir de um fluxo de caracteres

sem necessidade de considerar uma estrutura qualquer. As portas seriais são exemplos de character devices.

Existe ainda um outro tipo de dispositivo, **denominado pseudo-dispositivo** (*pseudo-devices*) que na realidade não corresponde a nenhum periférico físico. Ele é apenas uma abstração empregada para adicionar funcionalidades ao SO, explorando a interface padronizada já existente. É dessa forma que o Unix permite o acesso à memória principal (ramdisk) como se fosse um disco.

I/O independente do dispositivo

A camada de software de I/O independente do dispositivo implementa procedimentos e funções gerais a todos os dispositivos de entrada e saída. Os principais serviços sob responsabilidade dessa camada são:

- Escalonamento de I/O

- Denominação (inserção de nome lógico ao dispositivo)

- Buferização

- Cache de dados

- Alocação e liberação

- Direitos de acesso

- Tratamento de erros

Entrada e saída à nível de usuário

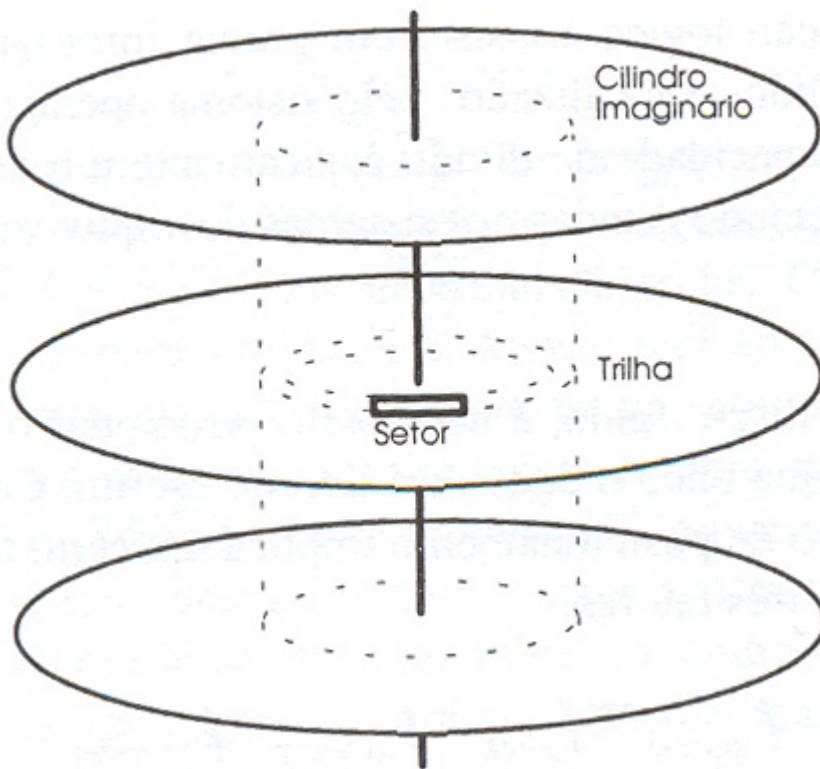
A visão que um usuário possui dos dispositivos de I/O de um sistema é fornecida por uma interface de programação associada as bibliotecas de entrada e saída, ou aos aplicativos de desenvolvimento.

Dispositivos periféricos típicos

Discos Rígidos

Os discos rígidos são dispositivos de armazenamento destinados a grandes quantidades de dados. Fisicamente, um disco rígido pode ser visto como composto por dois grandes blocos. O primeiro bloco é um conjunto de discos metálicos (aço ou alumínio) superpostos e dispostos em alturas diferentes com auxílio de um eixo central. As duas superfícies de cada um desses discos são recobertas por uma película magnética na qual os dados são gravados. No momento de acesso ao disco, essa estrutura é mantida em uma rotação constante. O segundo bloco é uma estrutura mecânica que suporta um conjunto de cabeçotes, um para cada superfície de disco. Essa estrutura é capaz de realizar movimentos de vai-vem de maneira que os cabeçotes possam ser deslocados da borda do disco até o centro.

Do ponto de vista da organização lógica, cada superfície de um disco é dividida em circunferências concêntricas denominadas **trilhas**. Cada trilha, por sua vez, é subdividida radialmente em unidades chamadas **setores**. Em geral todos os setores tem o mesmo tamanho, o qual varia entre 512 e 4096 bytes (o mais comum é 512). O setor constitui a unidade mínima de leitura e gravação em um disco. O conjunto de trilhas de todas as superfícies do disco que ficam exatamente à mesma distância do eixo central forma o **cilindro**. As abstrações cilindro, trilha e setor são utilizadas para designar a organização lógica da unidade de disco.



A definição de trilhas e setores em um disco chama-se **formatação física** e é um procedimento realizado no fabricante.

A controladora de disco também é outro pedaço do hardware. Devido a restrições físicas na construção de discos rígidos, ela ajuda em um problema: A tecnologia atualmente disponível nos permite apenas empilhar 8 discos, o que daria no máximo 16 cabeças de leitura e escrita. Entretanto, ao analisarmos as especificações de um disco real de, por exemplo, 4.1 Gb, encontramos 255 cabeças, 63 setores de 512 bytes e 525 cilindros. O que ocorre é que esses valores são divisões lógicas dispostas e gerenciadas pela controladora sobre 8 discos físicos, o que justifica em certa parte os números "quebrados". Nesse caso, as cabeças correspondem ao número de superfícies graváveis, e os cilindros, aos números de trilhas. Dessa forma, a capacidade total do disco é obtida multiplicando-se *cabeças x cilindros x setores x tamanho_do_setor* ($255 \times 525 \times 63 \times 512 = 4318272000$ bytes).

Outros termos bastante associados aos discos rígidos são **formatação lógica e partições**. A formatação lógica consiste em gravar informações no disco de forma que os arquivos possam ser escritos, lidos e localizados pelo SO. Por sua vez, o conceito de partição está associado à capacidade de dividir logicamente um disco em vários outros discos.

Tempo de acesso

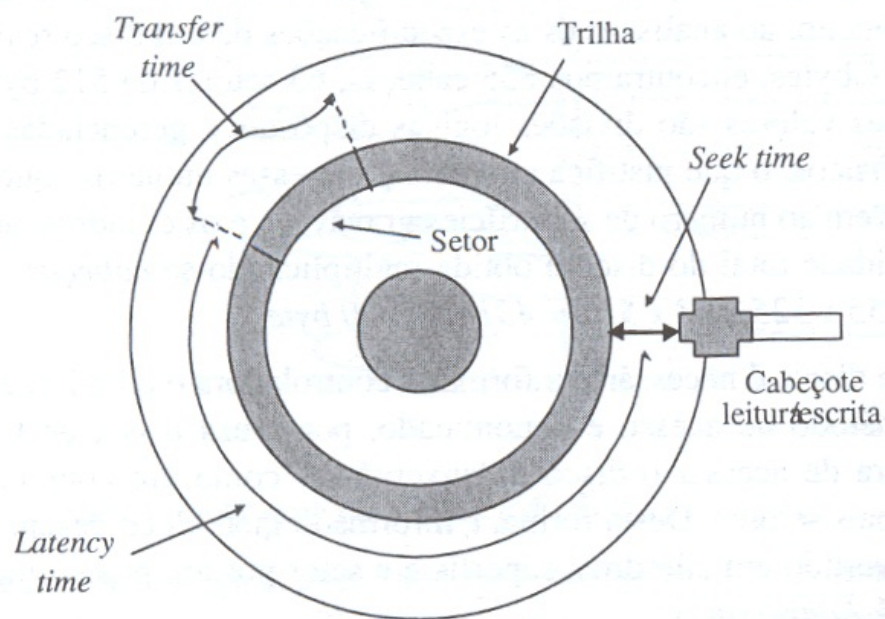
Para realizar um acesso ao HD, é necessário posicionar o cabeçote de leitura e escrita sob um determinado setor e trilha onde o dado será lido ou escrito. Isto é feito com um tempo de acesso. O tempo de acesso é assim dividido:

Seek time – tempo necessário para deslocar o cabeçote de leitura e escrita até o cilindro correspondente à trilha a ser acessada.

Latency time – tempo necessário, uma vez o cabeçote já na trilha correta para o setor a ser lido, ou escrito, se posicionar sob o cabeçote de leitura e escrita no início do setor a ser lido (ou escrito). Esse tempo também é denominado de atraso rotacional (rotational delay).

Transfer time – Corresponde ao tempo necessário à transferência dos dados, isso é, à leitura ou a escrita dos dados

$$t_{\text{access}} = t_{\text{seek}} + t_{\text{latency}} + t_{\text{transfer}}$$

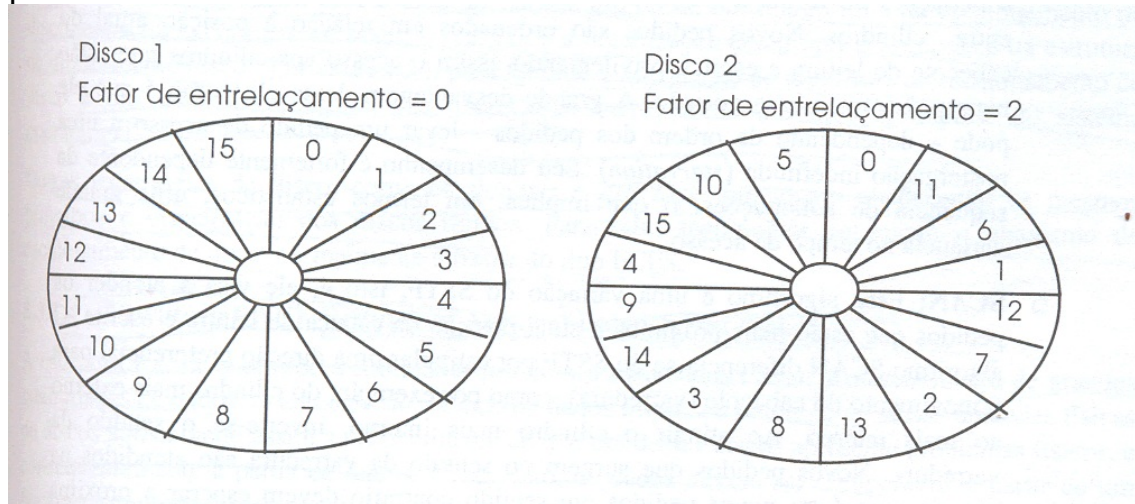


Entrelaçamento

Outro fator relacionado com a redução do tempo de acesso a um disco é o entrelaçamento (interleaving). É muito comum o acesso a vários setores contíguos em uma trilha de disco. Suponha que se deseje ler os setores 4 e 5 de uma determinada trilha. Devido à rotação do disco, o cabeçote de leitura e escrita provavelmente não se encontra mais no início do setor 5. Será necessário esperar que o disco dê uma volta completa (tempo de latência) para então efetuar a leitura do setor 5.

A forma usual para evitar esse problema é realizar um entrelaçamento dos setores. Essa técnica numera os setores não mais de forma contígua, mas sim com um espaço entre eles. O disco 2 da figura abaixo possui um fator de entrelaçamento igual a 2. Isso significa que entre o setor k e o setor $k+1$, existem dois outros setores. Podemos considerar que o disco 1 dessa mesma figura possui um fator de entrelaçamento igual a zero.

Com um entrelaçamento de 2, os setores 4 e 5 agora estão distantes, e após o cabeçote sair do setor 4 ele passa antes pelos setores 15 e 10. Dessa forma, ganha-se tempo e a transferência dos dados relativos ao setor 4 pode ser concluída, e o processador tem a chance de mandar o comando de leitura do setor 5 antes que o cabeçote passe sobre o início do mesmo.



Escalonamento de disco

Vejam agora as formas de escalonamento de um disco rígido:

FCFS(First come first served) – É o algoritmo de escalonamento mais simples. Nesse caso, as solicitações de acesso ao disco são realizadas na ordem em que os pedidos são feitos.

SSTF(Shortest seek time first) – A fila de pedidos é reordenada para atender as solicitações de forma a minimizar o movimento do cabeçote de leitura e escrita entre os cilindros. Novos pedidos são ordenados em relação à posição atual do cabeçote de leitura e escrita, privilegiando assim o acesso aos cilindros que estão mais próximos a essa posição. A grande desvantagem desse algoritmo é que ele pode levar um pedido de acesso à postergação indefinida.

SCAN – É uma variação do SSTF, isto em visa atender os pedidos que estão mais próximos à atual posição da cabeça de leitura e

escrita. O algoritmo SCAN diferencia-se do SSTF por estipular uma direção preferencial para o movimento do cabeçote (varredura). Novos pedidos que surgem no sentido da varredura são atendidos na própria varredura; novos pedidos em sentido contrário deverão esperar a próxima varredura para serem lidos. Por ser parecido com o mecanismo de um elevador, também chamamos esse algoritmo de algoritmo do elevador.

C-SCAN (circular SCAN) – O algoritmo SCAN, imediatamente após inverter a varredura, inicia o atendimento privilegiando os pedidos correspondentes aos cilindros recém-servidos por consequência os pedidos dos cilindros do outro extremo da varredura – feitos anteriormente – devem esperar. O algoritmo C-SCAN é uma variação do SCAN com o objetivo de acabar com esse privilégio. Nesse caso, por exemplo, quando a varredura atinge o cilindro mais externo, o cabeçote de leitura e escrita é reposicionado no cilindro mais externo onde ele reinicia a varredura.

Considerando a existência de todos esses algoritmos de escalonamento, podemos nos perguntar qual deles é o melhor. Na realidade, não existe resposta definitiva. Muitos fatores contribuem para o desempenho de um ou outro algoritmo de escalonamento de disco, entre eles o número de pedidos (carga), e a organização dos próprios arquivos e da estrutura de diretórios no disco. Por essa razão, para facilitar a adequação do algoritmo de escalonamento de disco a um sistema específico, este módulo é escrito como um módulo a parte do sistema operacional.

Outras tecnologias de disco, como floppys e zips, funcionam obedecendo os mesmos princípios e organização dos discos rígidos. Entretanto, o escalonamento de disco normalmente é fixo e do tipo FCFS.

Estrutura RAID (Redundant Array of Independent Disks)

A estrutura RAID se propõe a solucionar problemas associados com o armazenamento de grandes quantidades de dados. Ela é associada sempre à backups.

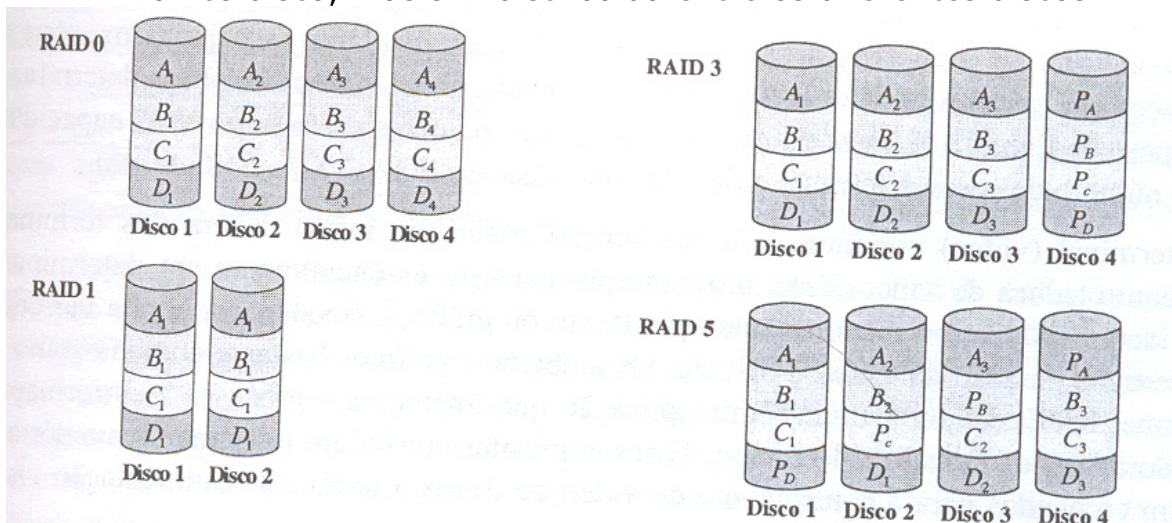
O princípio básico de uma estrutura RAID é combinar vários discos rígidos físicos em uma estrutura lógica de discos de forma a aumentar a confiabilidade e o desempenho dos discos. Como o próprio nome nos leva a imaginar, o conjunto de discos (array) armazena informações de forma redundante (redundant), permitindo assim a recuperação de dados em caso de falha física de um disco. Eis os níveis de RAID:

RAID 0 - Nesse nível, os dados a serem escritos são divididos entre os diferentes discos físicos que compõem o disco RAID sem considerar nenhum mecanismo para o controle ou correção de erros. Esse procedimento de escrever (e depois acessar) dados em paralelo em diferentes discos denomina-se stripping.

RAID 1 - Esse nível é conhecido como espelhamento (mirroring). Aqui um dado é escrito simultaneamente em um disco primário e em um disco secundário de cópia.

RAID 2/3/4 - Os dados são armazenados em diferentes discos, utilizando-se stripping, mas ao mesmo tempo, para cada strip, é calculada uma paridade. A paridade é então gravada em um disco físico à parte pertencente ao array. Em caso de falha em um dos discos, a informação contida nos demais discos, mais a informação da paridade, é suficiente para a reconstrução dos dados armazenados no disco que apresentou problemas. A diferença entre os níveis 2, 3 e 4 está na forma pela qual a paridade é calculada: paridade a bit (RAID 2), paridade a byte (RAID 3) e paridade de bloco (RAID 4).

RAID 5 - Esse nível é similar ao nível de RAID 3, isto é, os dados são divididos entre os diferentes discos e, para cada strip, é calculada a paridade. A diferença reside no fato de que essa informação de paridade não é mais concentrada em um único disco, mas sim distribuída entre os diferentes discos.



O controle necessário para implementação de uma configuração RAID pode ser realizado em hardware ou em software. Quando implementado por hardware a capacidade de stripping, cálculo de paridade, e da escrita simultânea em discos é provida pela própria controladora de disco. Nesse caso, nenhuma intervenção especial do SO é necessária.

Para o caso de RAID implementado por software, essas informações devem ser fornecidas pelo próprio SO, isto é, os dados a serem escritos são particionados pelo SO, e as diferentes fatias de dados são enviadas aos discos um por um.

Vídeo

No modo texto o vídeo é organizado em uma matriz de caracteres, as posições da memória de vídeo correspondem a caracteres e a seus atributos. No modo gráfico, a matriz é organizada em pixels; cada pixel tem associada uma série de posições de memória correspondentes as suas cores. A memória de vídeo da placa determina sua resolução (número de linhas e colunas da matriz) e o número de cores de cada pixel. A capacidade de resolução e número de cores determina o tipo da controladora (EGA, VGA, SVGA, etc).

Teclado

O princípio de operação do teclado é bastante simples: gerar um símbolo para cada tecla pressionada.

Quando uma tecla é pressionada, o teclado identifica a linha e a coluna associadas a essa tecla e gera um código que é denominado de scan code (código de varredura). Para cada tecla, existem, na realidade, dois códigos, um para identificar o pressionador da tecla e outro para sinalizar que a tecla foi liberada. Algumas teclas, ou seqüências de teclas (CTRL-ALT, etc) geram mais de um código. O pressionar de uma tecla gera ainda uma interrupção de hardware e, por consequência, a execução de um tratador de interrupções específico para o teclado.

O tratador de interrupção, nesse caso, lê registros internos específicos da interface do teclado para recuperar o scan code da tecla pressionada. Com base no scan code, o tratador de interrupções consulta uma tabela interna, substituindo o scan code pelo código ASCII correspondente à tecla pressionada. O código ASCII da tecla é, em seguida, armazenado em uma região especial da memória (buffer de teclado) de onde é recuperado por chamadas de SO. Nós podemos ver essa interação como uma relação do tipo produtor-consumidor, na qual o tratador de interrupções de teclado produz códigos ASCII e armazena-os em uma fila. Posteriormente, outro processo – o Sistema operacional – “consume” o conteúdo dessa fila, realizando assim a leitura dos caracteres pressionados. As primitivas para “consumo” de caracteres são disponibilizadas ao usuário final através de rotinas de bibliotecas, como, por exemplo, `getc()` na linguagem C.

Sob um certo ponto de vista, não se pode dissociar a entrada e saída de dados, já que nós usuários queremos visualizar os caracteres que digitamos na tela de nosso computador. Esse procedimento de ler dados do teclado e escreve-los na tela denomina-se **ecoamento**.

Rede

Uma placa de rede é constituída por um hardware que transforma os sinais digitais em sinais analógicos segundo sua tecnologia. Ela também possui internamente uma capacidade de memória (buffers) na qual dados a serem transmitidos, ou recebidos, são armazenados. A placa de rede trabalha sob um certo aspecto orientada a eventos. Um evento é o final da transmissão. A ocorrência desse evento é interpretada como disponibilidade para nova transmissão. Outro evento é a recepção de uma mensagem. Nesse caso, a mensagem deve ser lida. Essa descrição, entretanto, é extremamente simplista. O software de rede é bastante complexo e envolve muitas abstrações, e então ele é organizado em camadas.