Cruzeiro do Sul
Educacional

Gerenciador de Entrada e Saída

Conteudista
Prof. Me. Claudney Sanches Júnior

Revisão Textual
Aline de Fátima Camargo da Silva

Sumário

Objetivos da Unidade

- Estudar os conceitos do gerenciador de entrada e saída dos SO;
- Compreender o funcionamento do SO e os controladores;
- Entender o gerenciamento de E/S dos SOs modernos.

Atenção, estudante! Aqui, reforçamos o acesso ao conteúdo *on-line* para que você assista à videoaula. Será muito importante para o entendimento do conteúdo.

Este arquivo PDF contém o mesmo conteúdo visto *on-line*. Sua disponibilização é para consulta *off-line* e possibilidade de impressão. No entanto, recomendamos que acesse o conteúdo *on-line* para melhor aproveitamento.

VOCÊ SABE RESPONDER?

Você saberia determinar corretamente quais são os dispositivos de Entrada e quais são os de Saída? Há diferença em seus gerenciamentos?

Contextualização

Figura 1 – Programador
Fonte: Vida de Programador

#ParaTodosVerem: a imagem mostra uma história em quadrinho em quatro partes. No primeiro quadrinho, há o título "Vida de programador – dia do programador, com fundo preto. No segundo, há três homens, dois sentados em frente a computadores e um em pé; o homem em pé pergunta: "Quem lava a cafeteira hoje?", e, um dos homens sentados responde: "Hoje é dia do programador". No terceiro, os dois homens que estavam sentados, conversam com um quarto personagem, um dos rapazes pergunta: "Quem paga o almoço hoje", e o quarto personagem responde: "Hoje é dia do programador". No quarto quadrinho, ocorre a mesma cena do segundo quadrinho, o homem de pé pergunta: "Preciso que alguém fique até mais tarde hoje, porque o cliente está fazendo o fechamento do mês", então, o programador, sentado, responde: "Quem disser que hoje é dia do programador vai entender de forma dolorosa porque o teclado é um dispositivo de entrada". Fim da descrição.

A empresa comprou uma nova impressora e gostaria de compartilhar com outros usuários. Na máquina virtual com o servidor *Linux*, monte um servidor de impressão com o Samba para garantir a segurança e a autenticação.

Introdução

O gerenciamento de dispositivos de Entrada e Saída (E/S) é uma das principais funções de um SO e consiste em controlar o acesso a todos os dispositivos de E/S, tais como: teclado, vídeo, impressoras, disco, fita magnética etc.

Nesta unidade, você vai estudar como o SO organiza, controla e acessa os dispositivos de E/S. Também, serão apresentadas as rotinas de E/S e você conhecerá como funciona os controladores e os *device drivers* presentes em um SO.

Gerenciamento de Entrada e Saída

Uma das principais funções do SO é controlar todos os dispositivos de E/S do computador. Ele deve enviar comandos para os dispositivos, tratar erros, atender as interrupções e fornecer uma interface simples e fácil de usar entre os dispositivos e o resto do sistema. De maneira geral, gerenciar E/S é uma parte significativa do código do SO.

O principal objetivo do Gerenciamento de E/S é facilitar ao usuário o acesso aos dispositivos, sem que ele tenha que se preocupar com detalhes de funcionamento do *hardware* dos diversos dispositivos. Essa tarefa é bastante complexa, devido ao grande número de dispositivos e as diferenças eletromecânicas entre eles. Além disso, muitos dispositivos como os discos podem ser usados por vários usuários simultaneamente e o SO é responsável pela integridade dos dados compartilhados.

O *software* de E/S pode ser estruturado em uma das camadas do SO, sendo que cada uma delas exibe uma tarefa bem definida para executar. Veja, na Figura 2, a estrutura de um sistema computacional em camadas.

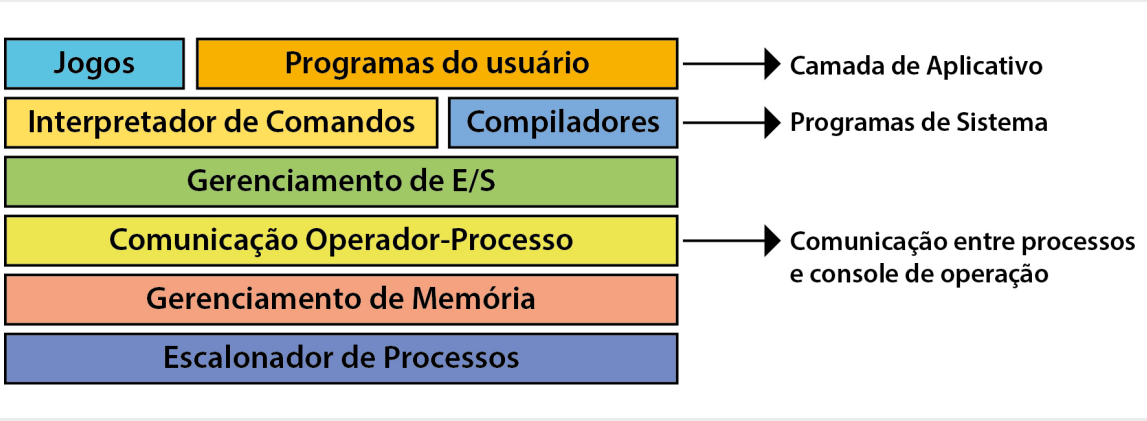

Figura 2 – Apresentação das camadas de SO
Fonte: Elaborada pelo conteudista

#ParaTodosVerem: a imagem mostra um quadro formado por duas linhas de cabeçalho: Jogos e Programas do Usuário, na primeira linha; e Interpretador de Comandos e Compiladores. Embaixo, há quatro linhas: Gerenciamento de E/S, Comunicação Operador-Processo, Gerenciamento de Memória, Escalonador de Processos. Fim da descrição.

Cada profissional da área computacional encara os dispositivos de E/S de uma forma distinta. Os engenheiros elétricos veem como chips, ligações elétricas, motores e componentes; enquanto os programadores, veem os comandos, as interfaces, as funções e os erros que podem ser repassados ao *software*. Porém, a programação dos dispositivos de E/S, muitas vezes, está interligada com o *hardware* e como ele se relaciona com o programa.

De modo genérico, podem-se classificar os dispositivos de E/S como sendo: dispositivos de bloco e dispositivo de caractere. O dispositivo de bloco é aquele que armazena a informação em blocos de tamanho fixo, normalmente de 512 *bytes* a 32.768 *bytes*. A principal característica dos dispositivos de bloco é que cada um deles pode ser lido e escrito independentemente. Os discos são os dispositivos de bloco mais comuns. O dispositivo de caractere envia e recebe um fluxo sem considerar quaisquer estruturas de bloco, ele não é endereçável e não dispõe de qualquer operação de posicionamento ou operação de acesso aleatório — *seek operation*. As impressoras, interfaces de redes, terminais e mouse são dispositivos de caractere.

Contudo, tal sistema de classificação não é perfeito. Os relógios ou timer que causam as interrupções não são classificados como dispositivos de bloco ou caractere. Os vídeos mapeados na memória também não se enquadram na classificação. Os sistemas de arquivos tratam os blocos como abstratos, mas, ainda assim, o modelo calcado em blocos ou caractere é amplamente utilizado para o desenvolvimento de *softwares* de E/S.

Os dispositivos de E/S apresentam uma enorme variação de velocidades, a qual impõem a necessidade de aceitar diferentes ordens de magnitudes. A seguir, veja algumas variações na Tabela 1.

Tabela 1 – Taxa de dados típicos

Dispositivos	Taxa de Dados
Teclado	10 bytes/s
Mouse	100 bytes/s
Modem	7 Kb/s
Impressora a *Laser*	100 Kb/s
Scanner	400 Kb/s
Ethernet	1,25 Mb/s
USB	1,5 Mb/s
Câmera de vídeo digital	4 Mb/s
Disco IDE	5 Mb/s
CD-ROM 40x	6 Mb/s
Disco ATA-2	16,7 Mb/s
FireWire	50 Mb/s
Disco SCSI Ultra - 2	80 Mb/s

Dispositivos	Taxa de Dados
Ethernet Gigabit	125 Mb/s
Barramento PCI	528 Mb/s

Fonte: Elaborada pelo conteudista

O SO fornece ao usuário uma interface de acesso aos dispositivos independente das características do *hardware*. Por exemplo, quando um usuário, ao desenvolver um programa em C, quer apresentar uma mensagem no monitor ele simplesmente utiliza o seguinte comando:

```
println("Oi")
```

Esse simples comando não é capaz de executar todos os procedimentos necessários para instruir o *hardware* do monitor a mostrar a mensagem. Na realidade, o compilador C traduz esse comando para uma chamada a uma rotina de E/S apropriada do SO, a qual envia os comandos para o driver do dispositivo que, por sua vez, instrui o controlador do dispositivo ou placa controladora. O controlador é quem envia os sinais ao monitor de vídeo para que este mostre a mensagem.

A seguir, veja a Figura 3, a qual demonstra o caminho que um comando escrito em um processo percorre até a sua execução.

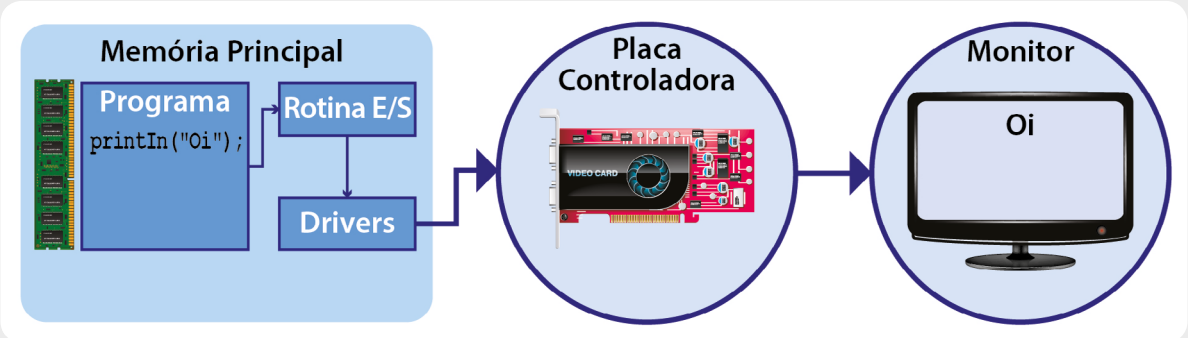

Figura 3 – Relacionamento entre os componentes de *software* e *hardware* em uma operação de escrita no monitor de vídeo

Fonte: Elaborada pelo conteudista

#ParaTodosVerem: a imagem mostra um esquema formado por uma forma retangular, seguida de duas formas circulares. Dentro da forma retangular há um esquema representando a Memória Principal; no primeiro círculo há uma placa controladora, em tons rosa e preto; no segundo círculo há um monitor. Fim da descrição.

Outro exemplo seria ler um registro de um arquivo gravado em um disco. Para isso, o programador somente necessitaria fornecer o nome do arquivo e o registro que desejaria ler. Entretanto, para que os dados sejam lidos do disco, seriam necessários vários comandos, por exemplo:

- Dizer a localização física do registro no disco, isto é, em qual disco, qual trilha, qual setor;
- Enviar o comando para rotacionar o disco e posicionar a cabeça de leitura e gravação sobre a trilha;
- Enviar o comando para ler os dados quando o setor passar pela cabeça;
- Enviar os dados lidos para a localização de memória especificada.

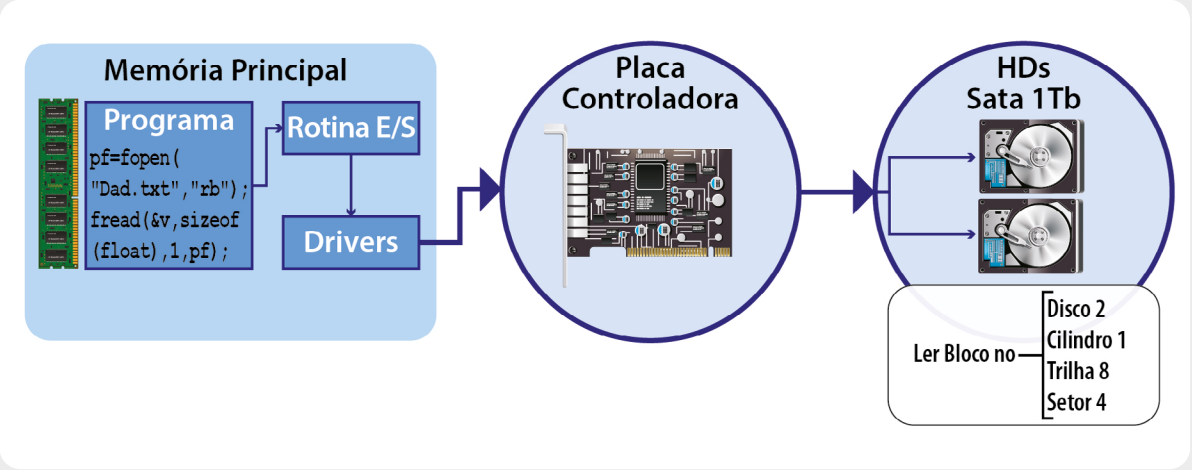

Figura 4 – Relacionamento entre os componentes de *software* e *hardware* em uma operação de leitura do disco
Fonte: Elaborada pelo conteudista
#ParaTodosVerem: a imagem mostra um esquema formado por uma forma retangular, seguida de duas formas circulares. Dentro da forma retangular há um esquema representando a Memória Principal; no primeiro círculo há uma placa controladora, em tons de preto e cinza; no segundo círculo há dois HDs. Fim da descrição.

O sistema de gerenciamento de dispositivos é feito: dividindo-se a tarefa de acesso aos dispositivos em várias camadas, cada qual, desempenhando uma função, conforme ilustrado na figura a seguir:

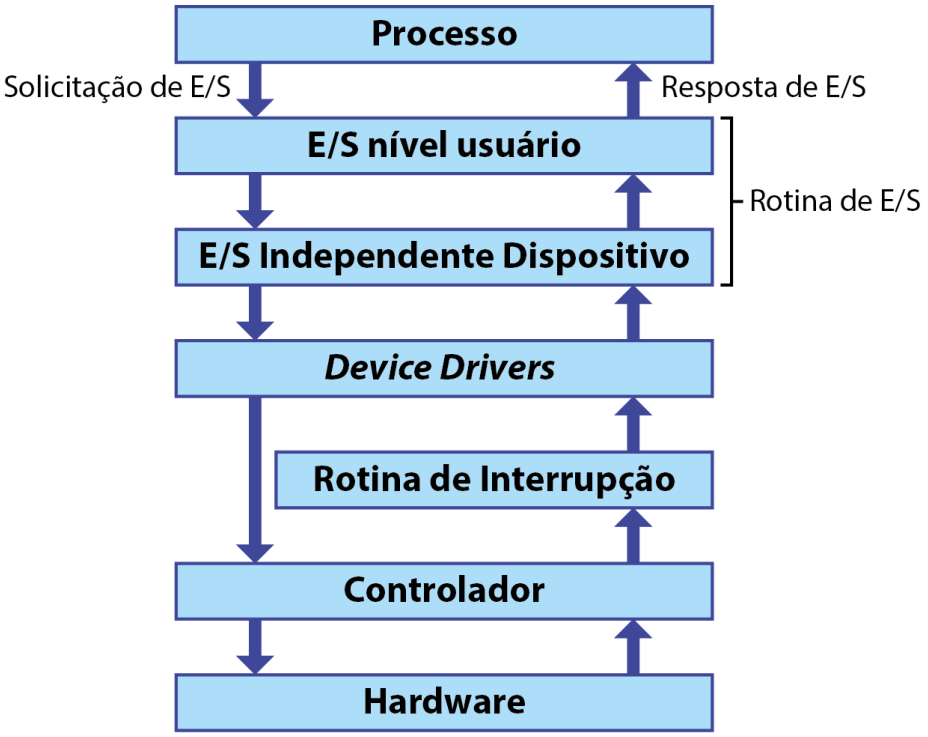

Figura 5 – Componentes do sistema de gerenciamento de dispositivos de E/S
Fonte: Elaborada pelo conteudista

#ParaTodosVerem: a imagem mostra um diagrama formado por três linhas azuis, uma embaixo da outra, e interligadas por meio de flechas azul escuro. Nas linhas, de cima para baixo, há os termos "Processo", "E/S nível usuário", "E/S Independente dispositivo", "*Device Drivers*", "Rotina de interrupção", "Controlador", "*Hardware*". Fim da descrição.

Rotina de E/S

Apesar de a maior parte do *software* de E/S estar dentro do SO, uma pequena parte dele pode ser acessada pelo usuário desenvolvedor. Esta parte está na camada de E/S em nível de usuário, no qual se encontram as bibliotecas que podem ser ligadas a programas de usuário. Por exemplo, os comandos *printf, read, write, wireteln* e o *spooling*.

Os objetivos da camada E/S independentemente do dispositivo são abrangentes, e pode ser citado como principais objetivos: criar uma interfaceamento uniforme para os *device drivers*, fornecer um mecanismo de nomeação do dispositivo, criar um tamanho de bloco independente do dispositivo, criar um espaço de bufferização para dispositivos de bloco e de caractere, cuidar da alocação de blocos livres em dispositivos de bloco, alocar e liberar dispositivos e manipular erros.

A rotina de interrupção ou manipuladores de interrupção deve ser escondida ou transparente ao usuário. A forma de implementação mais comum dos SO é a de que a rotina de solicitação de um processo, quando solicita uma E/S, o coloca no estado bloqueado e bloqueia o *device drivers*. Quando a interrupção de E/S ocorrer, a rotina de interrupção deve desbloquear o device drivers que, por sua vez, vai desbloquear o processo, colocando-o na fila dos prontos.

Controladores

O dispositivo de E/S se conecta ao sistema por intermédio de um controlador, o qual corresponde à parte eletrônica do dispositivo responsável por enviar os comandos para o dispositivo externo (a parte eletromecânica).

Geralmente, as unidades de E/S são compostas de dois componentes principais: componentes eletrônicos e os componentes mecânicos. Os componentes eletrônicos são conhecidos como o controlador de dispositivos ou adaptador. Nos computadores pessoais é comum encontrá-los na forma de uma placa controladora ou placa de circuito impressa, que pode ser inserida em algum conector de expansão ou slots do barramento. Por exemplo, existe a placa controladora de vídeo, a placa controladora de disco, a placa de rede etc.

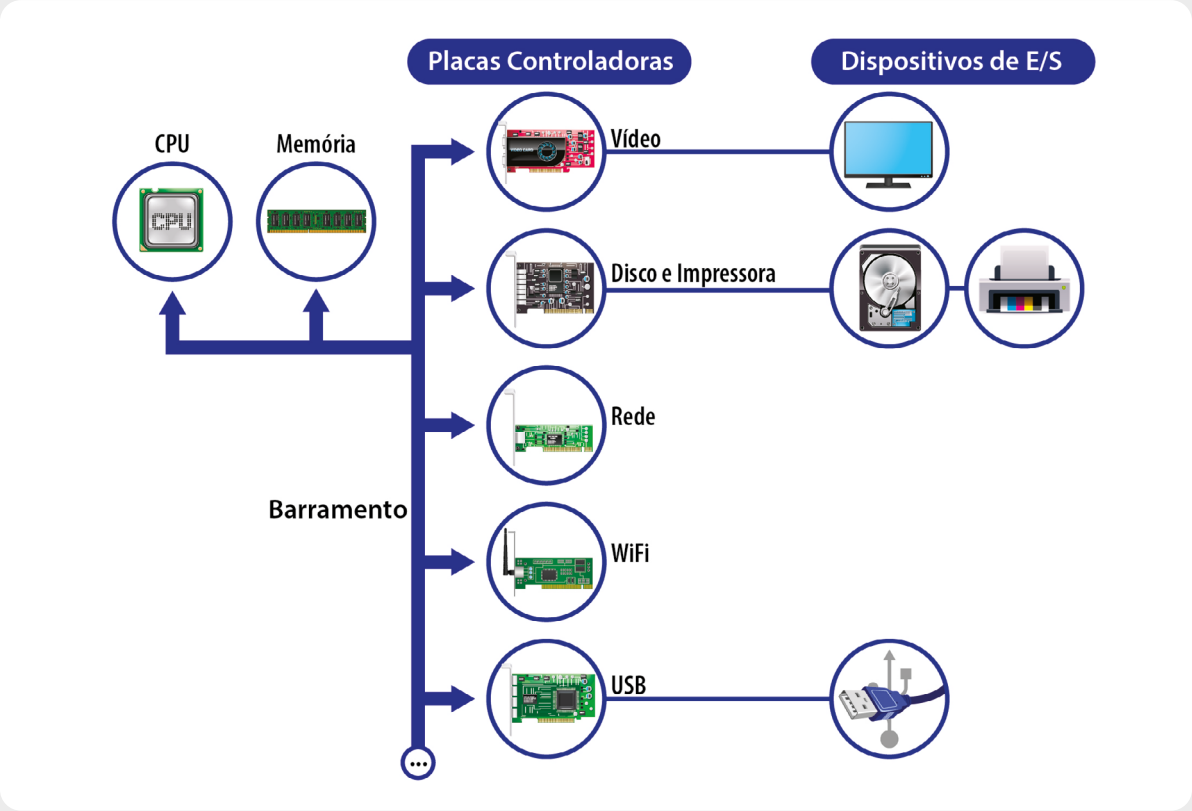

Figura 6 – CPU, Memória, Barramento e Placas Controladoras
Fonte: Elaborada pelo conteudista

#ParaTodosVerem: a imagem mostra um diagrama formado por formas circulares interligadas por setas. No lado esquerdo, em dois círculos lado a lado há o CPU e a Memória. Ao lado, já em forma de coluna, há 5 círculos, referentes a Vídeo, Disco e Impressora, Rede, WiFi, USB. Aos círculos "Vídeo" e "Disco e Impressora" ligam-se, ao lado, o ícone de um monitor e de uma impressora. Do mesmo modo, ao círculo USB, liga-se um ícone de um cabo USB. Fim da descrição.

Muitos controladores foram criados para controlar mais de um dispositivo mecânico, e normalmente são padronizados (IEEE, ISO, ANSI etc.). Com eles, o SO não necessita baixar o nível ou enviar instruções ao dispositivo explicitando cada ação. Por exemplo, o controlador de vídeo controla o disposto monitor explicitando como o sinal será gerado, a intensidade, o feixe e o *rendering*. Enquanto o SO pode enviar ao controlador instruções como: desenhe algo, escreva, utilize a resolução x etc. Se não existisse o controlador, o SO deveria conhecer o dispositivo e enviar instruções detalhadas para a execução dos mesmos comandos.

Cada controlador apresenta registradores para a comunicação com a CPU. Por meio da escrita nesses registradores, o SO comanda o controlador e com a leitura de alguns registradores ele sabe o estado do dispositivo. Além disso, alguns controladores exibem um *buffer* de dados, no qual o SO pode ler e escrever.

A CPU acessa os registradores e *buffers* do controlador, basicamente, de duas maneiras. A primeira é associar o registrador de controle ao número da porta de E/S. Assim, a instrução em *Assembly* ficaria IN REG, PORT, ou seja, a CPU deve ler o registrador do controlador PORT e armazenar o resultado no registrador REG. Os primeiros computadores utilizavam esta solução.

A segunda maneira é mapear todos os registradores no espaço de endereçamento da memória. Assim, cada um deles tem associado um endereço de memória único. Esse sistema é chamado de E/S mapeada na memória. Em geral, os endereços associados estão no topo da memória principal. Um esquema híbrido, no qual você encontrará as duas soluções juntas, foi elaborado, sendo o *buffer* associado ao E/S mapeado na memória e portas de E/S associada aos registradores.

Importante

O SO **não envia** os comandos diretamente para o dispositivo de E/S, e, sim, para o **controlador do dispositivo**. Este é quem comanda diretamente o dispositivo. Por exemplo, quando o SO identifica um comando para mostrar alguma mensagem no monitor de vídeo (*writeln* ou *println*), ele envia a mensagem para a placa controladora de vídeo, a qual enviará os sinais para comandar o monitor de vídeo para formar a mensagem.

Os controladores, em geral, possuem registradores internos que são usados para armazenar dados e comandos. O SO envia comandos para os controladores, carregando-os nos registradores do controlador. Os parâmetros do comando também são armazenados em outros registradores.

Após o comando ter sido aceito pelo controlador, a CPU fica livre para realizar outra tarefa, enquanto o controlador executa o comando de E/S.

Cite-se, quando um programa quer ler um bloco de dados do disco:

- O SO, por intermédio da CPU, envia os comandos de quais dados ler, onde colocar para o controlador, cabendo ao controlador comandar a leitura dos dados;

- Enquanto a leitura está sendo feita, a CPU pode ser direcionada para executar outra tarefa;

- Completada a leitura, o controlador gera uma interrupção para permitir que o SO ganhe o controle da CPU, verifique os resultados da operação e passe o controle para o programa que solicitou a leitura dos dados.

Direct Memory Access (DMA – Acesso Direto à Memória)

Os dados que entram no sistema mediante um dispositivo de entrada devem ser armazenados em uma área de memória, quer seja uma variável, ou um buffer de dados, para que possam ser utilizados por um programa. Nos primeiros sistemas de computadores, essa entrada de dados era feita em três etapas:

1. O controlador lê os dados do dispositivo e armazena-os em um *buffer* dentro do próprio controlador;
2. O controlador gera uma interrupção para avisar o SO de que os dados já estão disponíveis;
3. O SO é acionado, lê os dados do *buffer* do controlador e coloca-os em um *buffer* na memória principal.

Desse modo, na etapa 3, o SO usa a CPU a fim de fazer a transferência dos dados do controlador para a memória, sendo que a CPU poderia executar outro processo se o próprio controlador se encarregasse de colocar os dados na memória.

A maioria dos controladores de hoje, já conseguem fazer DMA. Nessa técnica, o próprio controlador já transfere os dados para a memória principal, liberando o SO desse trabalho e, consequentemente, liberando a CPU para realizar outro trabalho enquanto a transferência é efetuada. Por exemplo, imagine que um programa queira ler dados do disco:

- O SO envia um comando para o controlador de disco o informando sobre quais dados devem ser lidos e em que lugar devem ser armazenados na memória principal;
- O controlador lê os dados do disco, colocando-os em seu *buffer* interno;

- Após ter lido os dados do disco, o controlador transfere os dados para a localização da memória principal indicada pelo SO;

- Após a transferência, o controlador gera uma interrupção para avisar ao SO que os dados já se encontram na memória.

Driver de Dispositivo ou *Device Driver*

Um *device driver* é a parte do SO que é dependente do *hardware*. Ou seja, seu código é específico para manipular um dispositivo de E/S. A função de um *device driver* é receber os comandos da Rotina de E/S, reconhecê-los e enviar os comandos correspondentes para o controlador para que este possa comandar o dispositivo de E/S.

Vale ressaltar que o *device drivers*, cujos também podem ser descritos como direcionadores de dispositivos ou ser conhecido popularmente apenas por *driver*, tem como objetivo geral, aceitar os pedidos abstratos de serviços independentemente do dispositivo e ver como este pode ser executado. Normalmente, um sistema possui diversos *drivers*, tais como *drivers* para discos - *disk drivers*, fitas magnéticas - *tape drivers*, terminais – terminal *drivers*, impressoras etc.

Para entender melhor o funcionamento de um *driver*, considere o que acontece quando um processador de textos, por exemplo o *Word*, manda um documento ser impresso em negrito:

- Cada impressora tem seu conjunto de comandos específicos, que são diferentes entre elas. Um desses comandos será o de impressão em negrito;

- Mesmo que o processador de texto conheça o conjunto de comandos de várias impressoras, quando uma nova destas são lançadas, o processador não saberá manipulá-la;

- Para resolver esse problema, o processador de texto não envia os comandos diretamente para a impressora. Ele envia um comando abstrato (por exemplo, o comando negrito) para o *driver* dela;

- O *driver* da impressora converte esse comando abstrato para o comando correspondente da impressora. Esse comando é passado para o controlador desta e, posteriormente, para a impressora.

Conforme o exposto, há um conjunto de comandos abstratos que o *driver* usa, independentes de dispositivo que o programa utiliza e um conjunto de comandos correspondentes, comandos dependentes de dispositivo.

No caso de *driver* de impressoras, quando um fabricante lança uma nova desta no mercado, ele também disponibiliza o *driver* para utilização dela. Portanto, a interface entre os comandos abstratos e o *driver* deve ser bem definida. Ou seja, este precisa conhecer todos os comandos abstratos para saber identificá-los e enviar os comandos correspondentes para o controlador do dispositivo.

Os *drivers* atendem às requisições da seguinte forma: se estiver ocioso no momento de uma requisição ou pedido, ele o atende de imediato; mas se estiver ocupado, ele cria uma fila de pedidos pendentes.

Para atender ao pedido de E/S, os *drivers* necessitam traduzir os termos abstratos para uma forma mais concreta, ou seja, deve definir quais operações precisam ser programadas no controlador. Deve, ainda, escrever os comandos dos registradores do controlador, bloqueando-se até ser acordado por uma interrupção.

Após a operação ser concluída, o *device drivers* necessita verificar a presença de erros e despertar o processo solicitador passando os dados solicitados, bem como o código do *status* da operação. Por fim, irá verificar na fila de pedidos se existe algum pendente e, caso não encontre nenhum, irá se bloquear e ficar assim até um novo pedido.

Material Complementar

Leituras

Como Instalar o Samba
https://bit.ly/3MheedR

Servidor de Arquivos para Rede Local
https://bit.ly/3tQdMgj

Atividades de Fixação

1 – Qual é a função principal do gerenciador de entrada e saída (E/S) em sistemas operacionais?

a. Gerenciar o processamento de dados em aplicativos de escritório, como aceleração de texto e planilhas.
b. Controlar a conexão de dispositivos de *hardware*, como teclados, mouses e monitores, ao computador.
c. Garantir que todos os processos do sistema operacional tenham acesso total à memória RAM.
d. Realizar a alocação de recursos de CPU entre os processos em execução.
e. Gerenciar a comunicação entre o sistema operacional e dispositivos de armazenamento, como discos rígidos e unidades USB.

2 – Qual é a função principal dos *drivers* (ou *drivers*) em sistemas operacionais?

a. Gerenciar a segurança e proteção de dados em um sistema.
b. Controlar o acesso à *internet* e redes de computadores.
c. Garantir que todos os programas em execução tenham recursos suficientes de memória RAM.
d. Permitir a comunicação eficiente entre o sistema operacional e dispositivos de *hardware* específicos, como impressoras, placas de vídeo e dispositivos de armazenamento.
e. Monitore a temperatura interna do computador e evite superaquecimento.

Atenção, estudante! Veja o gabarito desta atividade de fixação no fim deste conteúdo.

Referências

DEITEL, H. M. **Sistemas Operacionais**. 3. ed. São Paulo: Pearson Prentice Hall, 2005.

TANENBAUM, A. S. **Sistemas Operacionais Modernos**. 3. ed. São Paulo: Pearson Prentice Hall, 2009.

Gabarito

Questão 1

e) Gerenciar a comunicação entre o sistema operacional e dispositivos de armazenamento, como discos rígidos e unidades USB.

Justificativa: o gerenciador de entrada e saída (E/S) em sistemas operacionais é responsável por gerenciar a comunicação entre o sistema operacional e dispositivos de armazenamento, como discos rígidos, unidades USB, CD/DVD, entre outros. Ele controla a leitura e gravação de dados em dispositivos de armazenamento e garante que os processos tenham acesso eficiente a esses dispositivos.

Questão 2

d) Permitir a comunicação eficiente entre o sistema operacional e dispositivos de *hardware* específicos, como impressoras, placas de vídeo e dispositivos de armazenamento.

Justificativa: os *drivers*, também conhecidos como *drivers*, são programas de *software* específicos para permitir que o sistema operacional seja comunicado de maneira eficaz com dispositivos de *hardware* específicos. Eles atuam como intermediários, traduzindo comandos do sistema operacional para uma linguagem que o *hardware* do dispositivo possa entender. Isso garante que os dispositivos funcionem corretamente quando conectados ao computador.