Gerenciamento de Entrada e Saída

Hélio Crestana Guardia e Hermes Senger

O controle da **entrada e saída** (E/S – ou I/O, *input/output*) de dados dos dispositivos é uma das funções principais de um sistema operacional. Para promover o **compartilhamento seguro** do uso dos recursos, contudo, não é permitido aos processos o acesso direto aos dispositivos de entrada e saída. Assim, cabe ao SO oferecer serviços (chamadas de sistema) que permitam ler e escrever dados.

A interação dos programas com o SO para o acesso aos dispositivos pode ocorrer enviando e recebendo **bytes** de/para dispositivos de caractere, ou realizando operações de **arquivos** em dispositivos de bloco.

Atuações do SO nas operações de E/S:

- Emitir comandos para os dispositivos
- Atender interrupções geradas pelos dispositivos
- Tratar erros nas operações desses dispositivos
- Prover uma interface para utilização dos dispositivos; se possível a mesma para todos eles, de forma que a sintaxe das operações seja independente dos dispositivos.

Para entender como o SO trata do gerenciamento dos dispositivos, é importante conhecer alguns aspectos de suas operações.

Aspectos de *hardware*

Dispositivos de E/S

Dispositivos de E/E podem ser agrupados basicamente em duas categorias: dispositivos de **caractere** e dispositivos de **bloco**. Dispositivos de **caractere** permitem a transferência de sequências de caracteres nas operações de leitura e escrita. Exemplos desse tipo de dispositivo incluem terminais, impressoras de linha, etc., em que o acesso é puramente sequencial. Dispositivos de **bloco**, como os discos rígidos e as unidades de armazenamento em memória externa, manipulam informação em blocos de tamanho fixo, normalmente com tamanho múltiplo de 512 bytes. Diferentemente dos dispositivos de caractere, contudo, dispositivos de bloco permitem acessar cada bloco de armazenamento de maneira independente.

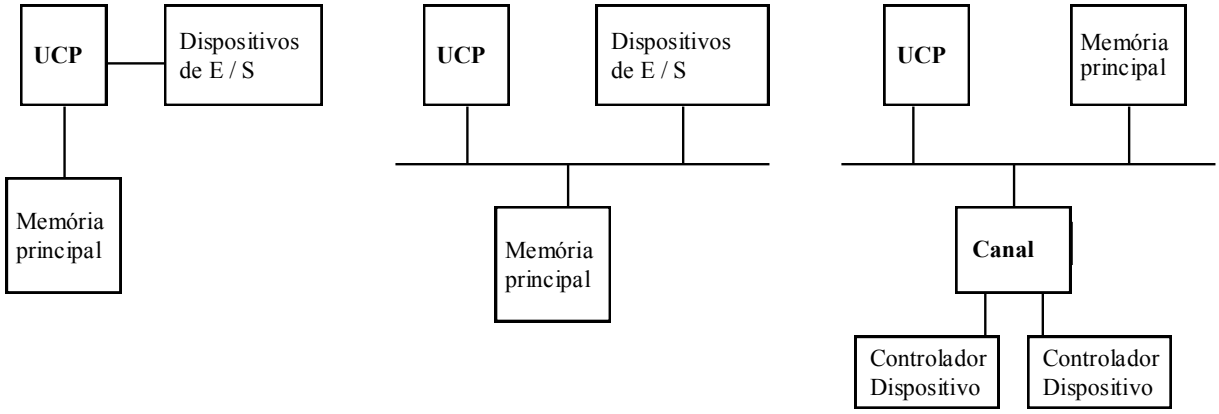
Controladores de dispositivos

Unidades de E/S apresentam dois componentes, um **componente mecânico**, que corresponde ao **dispositivo** que realiza as operações, e um **componente eletrônico**, chamado de controlador de dispositivo, que é conectado ao computador e atua como interface para o dispositivo. Alguns controladores podem intermediar a interação com mais de um dispositivos de E/S. Um controlador de discos SATA, por exemplo, pode tratar da interação com vários discos SATA. Do mesmo modo, um controlador USB pode lidar com vários dispositivos USB.

A conexão entre o processador e a memória ocorre através de um **barramento**, ao qual também pode estar interligado um controlador de vídeo de alto desempenho. Ligado a esse barramento podem estar ligados outros barramentos, como o PCI ou USB, aos quais outros controladores de dispositivos são interligados.

Uma característica dessa forma de interconexão é que, para o SO, sendo executado na CPU, realizar qualquer transferência de/para algum controlador de dispositivo, é preciso usar instruções específicas para a leitura e escrita no barramento. O mecanismo de DMA, citado posteriormente nesse texto, pode, contudo, alterar essa forma de operação.

Computadores de grande porte usam múltiplos barramentos com controladores especializados para E/S (canais) que aliviam a carga da UCP principal.



Sistemas operacionais, normalmente, realizam E/S interagindo com os controladores, ao invés de atuar diretamente com os dispositivos. Considerando a leitura de dados do disco, por exemplo, podemos ver a atuação do controlador:

- a informação é lida do disco na forma de uma sequência de bits, começando com um preâmbulo (nº do setor e do cilindro, tamanho do setor, etc), seguindo-se os 4096 bits de um setor e um código de correção de erro.
- o controlador converte a fileira de bits em um bloco de bytes, montado num *buffer* interno.
- depois de verificado contra erros, o bloco pode ser copiado para a memória principal.

A interação entre cada controlador e a UCP ocorre através de registradores especiais, que podem fazer parte da área de endereçamento da memória (mapeados na memória), ou podem referir-se a endereços no barramento. Desta forma, cada controlador é acessado, via barramento, através de um conjunto de endereços definidos. Leitura e escrita via barramento são feitas com instruções IN e OUT, e equivalentes, do repertório de instruções do processador.

Ex: IBM-PC - endereços especiais para E/S

***Control*ador de E/S**		**Endereços**	**Interrupções**
Teclado		60H - 63H	09 H
RS 232 primária	COM1	3F8H - 3FFH	0C H
	COM2	2F8H - 2FFH	0B H
Impressora	LPT1	378H - 37FH	07 H

O sistema operacional executa E/S escrevendo valores nos registradores dos controladores. O controlador de disquetes do IBM-PC, por exemplo, aceita**va** (!) 15 comandos diferentes, como: READ, WRITE, FORMAT, SEEK, etc.

Quando um comando é aceito pelo controlador, a UCP pode ser direcionada para outras atividades. **Interrupções** de fim de operação chamam a atenção da UCP, que é transferida para a execução de uma rotina do SO que lê os resultados da operação nos registradores.

Acesso Direto à Memória (DMA)

Para evitar que o processador tenha que intermediar cada transferência de valores de/para um controlador de dispositivo, usando o barramento, é possível usar um dispositivo de controle que permite o acesso direto à memória pelos controladores, sem intermediação do processador. Esse dispositivo é o **controlador de acesso direto à memória** (DMA). Assim, é possível ao SO iniciar a operação do controlador DMA, indicando qual a operação deve ser realizada (leitura ou escrita), qual endereço de controlador deve ser acessado, e em que posições da memória e quantos dados devem ser copiados.

As etapas nas operações de E/S sem e com o acesso direto à memória são descritas a seguir.

a) Acesso direto sem DMA

- SO envia comando de transferência para controlador do dispositivo
- o controlador lê o bloco (pensando em dispositivo de bloco) bit a bit e o armazena num *buffer* interno
- a verificação de erro é efetuada
- se não houve erros, o controlador gera uma interrupção
- a UCP (executando código do SO) lê os dados, bytes ou palavras, via barramento, e os armazena na memória

Problema: gasto de tempos da UCP para ler informações byte a byte pelo barramento

b) Acesso a disco com DMA

- SO envia comando de transferência para o controlador do dispositivo
- SO passa endereço da memória e tamanho como parâmetros extras ao controlador de DMA
- dados já verificados, armazenados no *buffer* pelo controlador, são transferidos para a memória diretamente, ou por intermédio do controlador de DMA
- interrupção é gerada à UCP quando transferência foi concluída

Aspectos de *software*

Definido que o SO deverá implementar e oferecer serviços de E/S para os processos, resta pensar como organizar esse código de maneira eficiente. Uma estratégia comum usada pelos SOs para a implementação do *software* de E/S consiste em organizar as suas funções como uma série de níveis. Enquanto os níveis mais baixos procuram esconder as particularidades do *hardware* das

camadas superiores, estas procuram implementar uma interface bem definida e com boa apresentação.

Objetivos do *software* de E/S

- **Independência de dispositivo**: programas devem poder usar o mesmo código para manipulação dos diversos dispositivos (discos rígidos e flexíveis, unidades de memória USB, impressoras e terminais, etc). Deve ser responsabilidade do SO cuidar das diferenças de operação entre os dispositivos.
- **Nomeação uniforme**: arquivos e dispositivos devem poder ser endereçados simplesmente por nomes independentes de dispositivos. Também a transferência entre eles não deve requerer alterações nos nomes.
- **Manipulação de erros**: erros devem ser tratados o mais próximo do *hardware* possível, sendo notificados aos níveis superiores somente no caso de não ser possível tratá-los no nível em que são encontrados.
- **Transferência síncrona (bloqueante) e assíncrona (orientada a interrupções)**: é responsabilidade do SO fazer com que as operações de E/S sejam bloqueantes para os programas de usuário, de forma que elas só retornam quando concluídas, ou não, caso essa forma de operação seja solicitada explicitamente pelo processo.
- **Dispositivos compartilhados e dedicados**: o SO deve propiciar o uso de recursos dedicados e gerenciar o compartilhamento de recursos de uso comum.

Visando atender esses objetivos, uma forma de estruturação do *software* de E/S é apresentada:

1. Tratadores de interrupção
2. Controladores de dispositivos
3. *Software* de E/S independente de dispositivo
4. *Software* de E/S ao nível do usuário

Tratadores de interrupção

Para esconder as interrupções dos níveis superiores do *software* do sistema, suas ocorrências, normalmente, são tratadas da seguinte forma:

- controladores de dispositivos são implementados na forma de processos, ou outras entidades do SO que podem ser bloqueadas
- à medida que um comando de E/S é emitido, o processo controlador apropriado é bloqueado, aguardando o término da operação iniciada
- na ocorrência de uma interrupção, a rotina de tratamento é encarregada de acordar o processo controlador apropriado do SO
- o processo de tratamento do SO acorda o processo cliente da operação de E/S solicitada

*Control*adores de dispositivos (*device drivers*)

Device drivers contêm a parte de código de E/S específica (dependente) para cada dispositivo, servindo, no máximo, para uma classe de dispositivos bem semelhantes. Têm a função de emitir os comandos aos controladores físicos (através dos registradores) e verificar que eles sejam desempenhados de maneira apropriada.

Considerando um controlador de disco, por exemplo, detalhes sobre setores, trilhas, cilindros, cabeças, movimento da cabeça de leitura, fator de *interleaving*, e outros aspectos físicos

e lógicos só são conhecidos no âmbito de sua atuação. É sua responsabilidade receber os pedidos de solicitação de serviço e determinar a sequência em que as operações serão realizadas. Ex: determinar se o motor está funcionando, posicionar a cabeça de leitura, etc.

Requisições recebidas pelos controladores envolvem a ativação do dispositivo, podendo o processo ficar bloqueado até que a operação seja completada, ou não, dependendo da operação solicitada. Operações concluídas podem ainda envolver o retorno de dados, ou parâmetros de status da operação, ao *software* independente de dispositivo.

De maneira geral, *device drivers* são específicos para um Sistema Operacional, e são comumente implementados pelo fabricante do controlador físico de dispositivo. Para tanto, o fabricante deve conhecer quais **serviços** deve implementar para o SO específico, com quais interfaces e parâmetros, e traduzir esses serviços padronizados do SO para comandos específicos reconhecidos pelo controlador físico de dispositivo. *Device drivers* também podem incluir parte da rotina de tratamento de interrupção do dispositivo ao qual se referem.

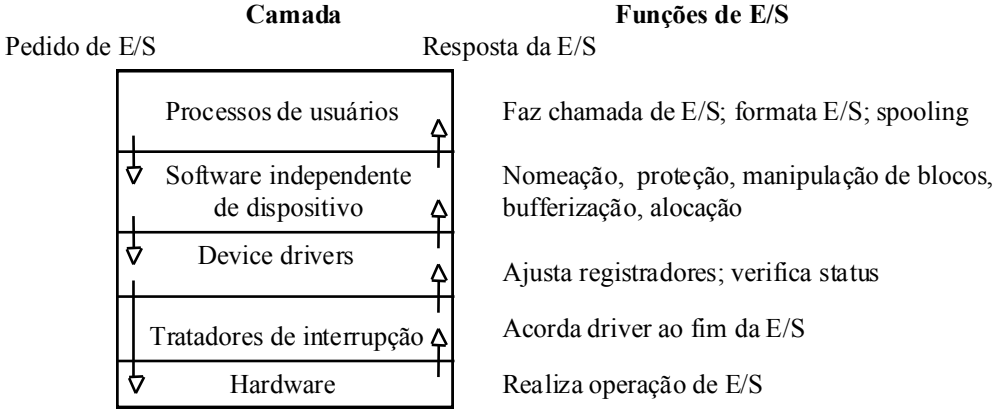
Software de E/S independente de dispositivo

Embora a distinção entre a parte do *software* de E/S que é dependente de dispositivo, ou não, seja particular a cada SO, as funções que seguem são tipicamente independentes de dispositivos:

- interfaceamento uniforme para os controladores de dispositivos
- nomeação dos dispositivos: mapeamento dos nomes simbólicos aos dispositivos
- proteção dos dispositivos
- fornecimento de um tamanho de bloco independente de dispositivo
- bufferização: compatibilizar diferenças entre o tamanho da informação solicitada pelos processos de usuários e as unidades usadas pelos dispositivos
- alocação de espaço em dispositivos de bloco: algoritmos independentes para determinação de espaços disponíveis
- alocação e liberação de dispositivos dedicados

Interação entre as camadas

A interação entre as camadas do software de E/S de um SO ocorre via chamadas de funções / serviços e envolve também a ocorrência de interrupções. A figura a seguir ilustra essas interações.



Entrada e saída padrão

Processos comumente solicitam dados de entrada que devem ser fornecidos pelos usuários. Também é comum a exibição de dados (mensagens) para os usuários durante a execução de um programa.

Para isso, no Unix, há o conceito de **arquivos padrão** para **entrada**, **saída** e **saída de erros**. Chamados, respectivamente, de STDIN, STDOUT e STDERR, esses 3 arquivos estão comumente associados ao **terminal**, ou à janela em que o programa foi iniciado (caso o usuário esteja interagindo com o sistema através de um ambiente gráfico).

Terminais:

- mapeados em memória ou conectados através de linha serial
- conectados ao teclado ou não
- *raw* (simples encaminhadores de dados) e *cooked* (permitem edição dos dados digitados)
- eco (repetição dos dados digitados na tela)

Leitura complementar

Para complementar a aprendizagem sobre processos recomendamos a leitura do capítulo 13 do Silberschatz ou o capítulo 5 do Tanenbaum (este último, disponível online para a UAB através do link http://ufscar.bvirtual.com.br). Ambos os capítulos possuem exercícios que vale a pena ver. Isto não é obrigatório, mas é bastante desejável para uma melhor aprendizagem.