Gerenciamento de Entrada e Saída

Ciclo 6 – AT1

Prof. Hermes Senger

Referência:

Tanenbaum – Cap. 5

Silberschatz – Cap. 13

Nota

O presente material foi elaborado com base no material didático dos livros:

Sistemas Operacionais com Java, 7ª edição, de A. Silberschatz, P.B. Galvin e G. Gagne, Editora Campus, 2008, disponibilizado pela editora.

Sistemas Operacionais Modernos, 2a edição, de A. Tanenbaum, Editora Pearson Prentice Hall, São Paulo, 2003, disponibilizado pela editora.

Subsistema de E/S

- Hardware de E/S
- Interface de aplicação para E/S
- Subsistema de E/S do Kernel
- Tratamento de E/S
- Tratamento de requisições de E/S

Objetivos

- Estudar a estrutura geral de um subsistema de E/S
- Compreender os princípios do tratamento de E/S e sua complexidade

Hardware de E/S

Varia muito de um sistema para outro.

O S.O. precisa "acomodar" essa variação de forma transparente ao usuário.

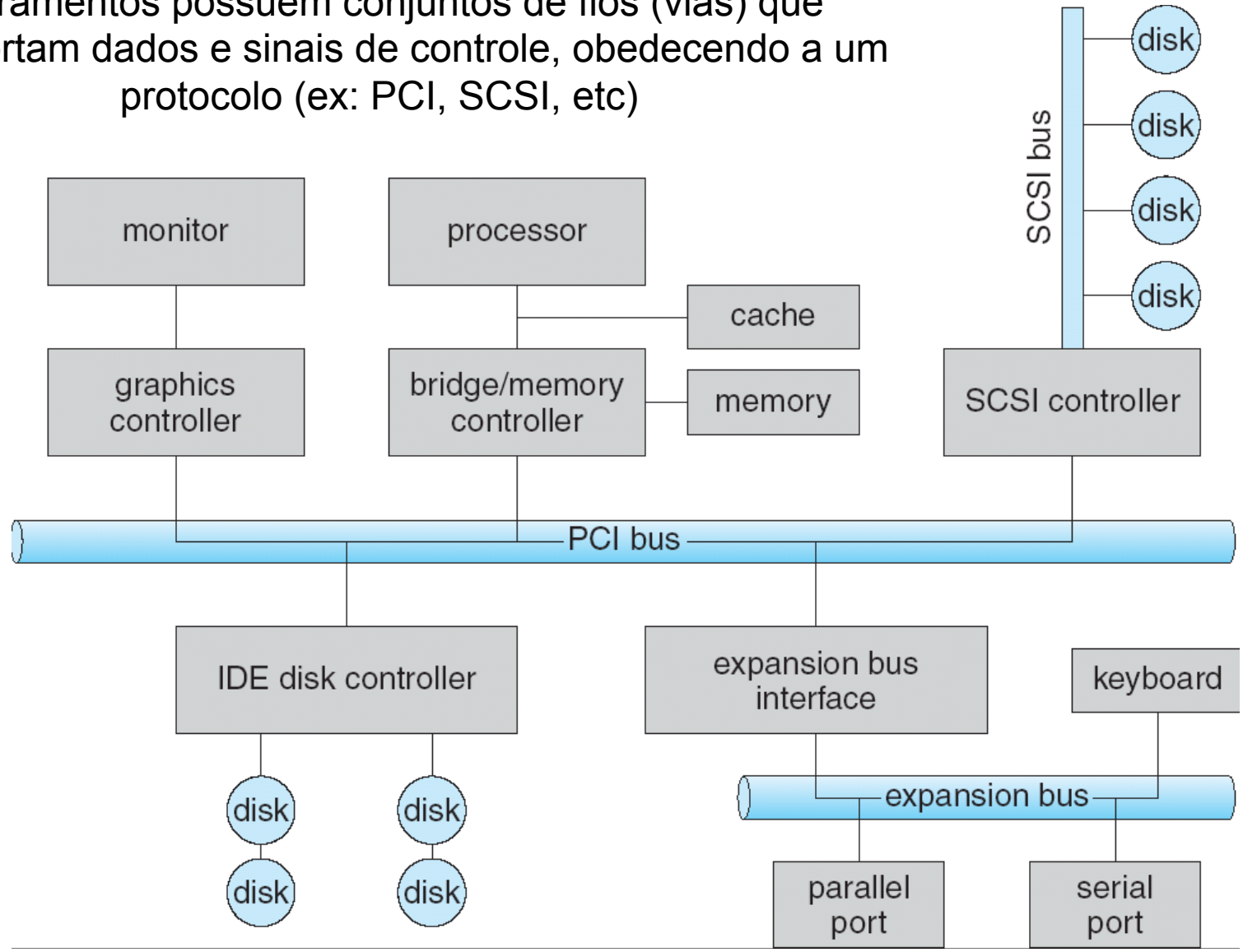
Dispositivo	Taxa de dados
Teclado	10 bytes/s
Mouse	100 bytes/s
Modem 56 K	7 KB/s
Canal telefônico	8 KB/s
Linhas ISDN dual	16 KB/s
Impressora a laser	100 KB/s
Scanner	400 KB/s
Ethernet clássica	1,25 MB/s
USB (*universal serial bus* — barramento serial universal)	1,5 MB/s
Câmara de vídeo digital	4 MB/s
Disco IDE	5 MB/s
CD-ROM 40x	6 MB/s
Ethernet rápida	12,5 MB/s
Barramento ISA	16,7 MB/s
Disco EIDE (ATA-2)	16,7 MB/s
FireWire (IEEE 1394)	50 MB/s
Monitor XGA	60 MB/s
Rede SONET OC-12	78 MB/s
Disco SCSI Ultra 2	80 MB/s
Ethernet Gigabit	125 MB/s
Dispositivo de Fita Ultrium	320 MB/s
Barramento PCI	528 MB/s
Barramento da Sun Gigaplane XB	20 GB/s

Controladores(as)

- Tarefas da controladora:
  - ❖ Enviar sinais de controle ao dispositivo
  - ❖ Converter fluxo serial de bits em bloco de bytes
  - ❖ Executar correção de erros
  - ❖ Tornar o bloco disponível para ser copiado para a memória principal
- Grande variedade de controladoras:
  - ❖ Algumas mais simples: controladora de porta serial – chip que atua sobre os sinais de uma porta serial
  - ❖ Ou mais complexas: por exemplo, controladora de discos SCSI
    - ◆ geralmente implementada como uma placa separada que é conectada ao computador (possui processador, microcódigo, e uma memória privada) para processar mensagens do protocolo SCSI que é bastante complexo
- Tipicamente possui **registradores de dados**, **de comando** e **de *status*** que permitem a interação do processador

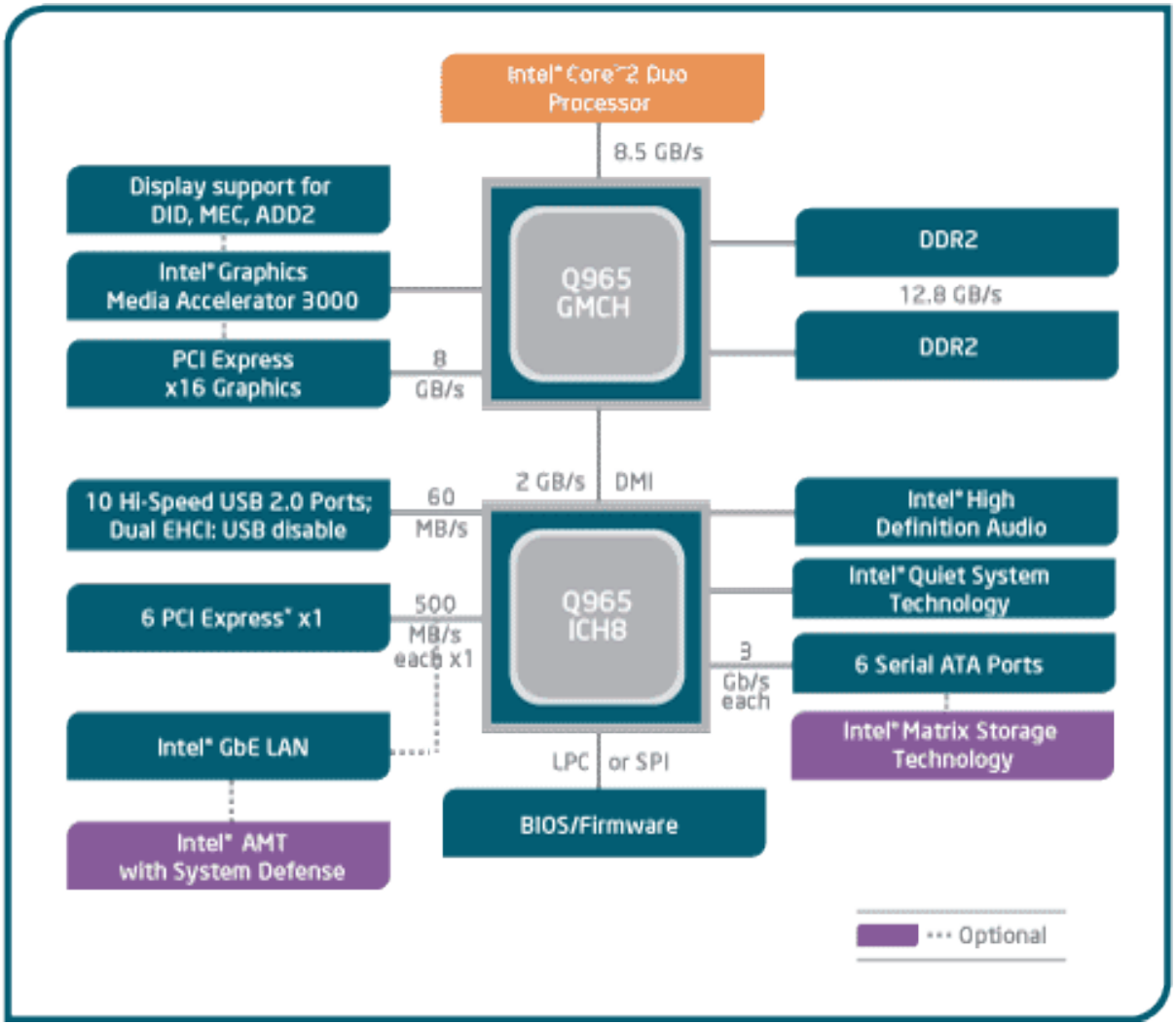
Estrutura típica do barramento do PC

Barramentos possuem conjuntos de fios (vias) que transportam dados e sinais de controle, obedecendo a um protocolo (ex: PCI, SCSI, etc)



Hardware de E/S do Pentium

(figura copiada de intel.com)



Comunicação entre CPU e controladoras

- A CPU se comunica com as controladoras através de três arquiteturas básicas
  - ***Memory-mapped I/O***: os registradores da controladora são mapeados no espaço de endereçamento de memória do processador
    - Ex: controladora de vídeo do PC (memória de vídeo)
  - **Portas de I/O com endereços específicos** são usadas para implementar os registradores (de comandos, registradores de dados, e de status)
  - Híbrido

Comunicação entre CPU e controladoras

- Exemplo de instrução de E/S para arquitetura *Memory-mapped I/O*

```
MOV     AX, [1000]
```

Obs.: 1000 seria um endereço de dispositivo mapeado em memória

- Exemplo de instrução de E/S para arquitetura baseada em portas de I/O

```
IN      AL, 3F8h
```

Obs.: 3F8h é o endereço da porta serial

Endereços de algumas portas de I/O do PC

Faixa de endereços de I/O (hexa)	Dispositivo
000-00F	Controladora de DMA
020-021	Controladora de interupção
200-20F	Controladora de jogos
2F8-2FF	Porta serial (secundária)
320-32F	Controladora de disco
378-37F	Porta paralela
3D0-3DF	Controladora de vídeo
3F0-3F7	Controladora de Floppy
3F8-3FF	Porta serial (primária)

Portas de I/O tipicamente possuem 4 registradores: **data-in**, **data-out**, **Status** e **control**

E/S programada - Polling

- Dispositivos de E/S precisam ser controlados. Ex.:
  - Ex: Determinar o estado (*status*) de um dispositivo
    - command-ready
    - busy
    - Error
- Há várias três maneiras básicas de fazer isso
- **Busy-wait** ciclo de espera até que o dispositivo esteja pronto
- Ex: o pseudo-código abaixo envia uma cadeia de caracteres para a impressora

```
copy_from_user(buffer, p, count);          /* p é um buffer do núcleo */
for(i=0; i < count; i ++) {                /* executa o laço para cada caractere */
        while(*printer_status_reg != READY);   /* fica no laço até que status indique pronto */
        *printer_data_register = p[i];         /* envia um caracter para a saída */
}
return_to_user( );
```

E/S por Interrupções

- Ex: pseudo-código abaixo envia uma cadeia de caracteres para a impressora usando E/S orientada à interrupção

```
copy_from_user(buffer, p, count);
enable_interrupts( );
while (*printer_status_reg != READY) ;
*printer_data_register = p[0];
scheduler( );
```

(a)

```
if (count == 0) {
    unblock_user( );
} else {
    *printer_data_register = p[i];
    count = count – 1;
    i = i + 1;
}
acknowledge_interrupt( );
return_from_interrupt( );
```

(b)

a) Código executado quando é feita a chamada ao sistema para impressão

b) Rotina de tratamento de interrupção

E/S por Interrupções

- O dispositivo avisa a CPU que está pronto
  - ❖ envia um sinal **Interrupt-request** em um pino específico
- A CPU salva o estado de execução corrente
- O **vetor de interrupção** despacha a interrupção ao **tratador** apropriado (***interrupt handler***)
  - ❖ Cada posição do vetor contém o endereço de um tratador específico para uma interrupção
  - ❖ Isso pode ser feito com base em prioridades
  - ❖ Certas interrupções podem ser adiadas ou até mesmo ignoradas
- O ***interrupt handler*** recebe e trata a interrupção
- Esse mesmo mecanismo também é usado para **exceções** (*exceptions*)

E/S por Interrupções

- Há 2 tipos básicos de interrupções:
  - **Mascarável** (*maskable*): o seu atendimento pode ser ignorado ou atrasado
    - Ex: O S.O. pode desabilitar interrupção antes de executar algum trecho de código crítico
    - É usada por controladores de dispositivos para requisitar serviço (ex: impressora avisa que está pronta)
  - **Não-mascarável** (*nonmaskable*): reservada para eventos tais como erros de memória irrecuperáveis
- Computadores modernos necessitam de um controlador de interrupções

Vetor de eventos do Intel Pentium

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

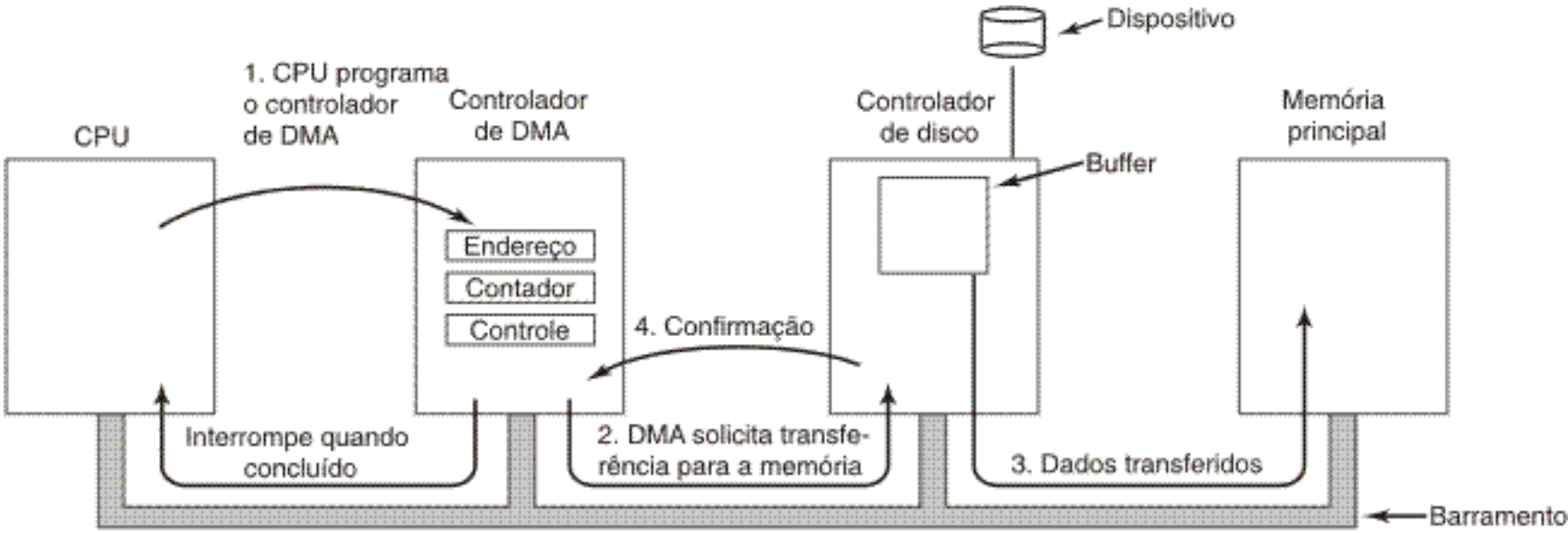
Não-mascaráveis

Dispositivos (mascaráveis)

Acesso Direto à Memória

- **DMA** – *Direct Memory Access*
  - ❖ É um processador de uso específico, encarregado de fazer grandes transferências de dados
  - ❖ Também chamado de controlador de DMA
- Utilizado para evitar a E/S programada no caso de grandes transferências
- Transfere dados diretamente entre um controlador de dispositivo e a memória
  - ❖ Libera o processador para outras tarefas (melhora a eficiência do sistema)

Operação de transferência por DMA



- Nesse caso, a CPU recebe uma única interrupção gerada pelo DMA, avisando que a transferência foi concluída

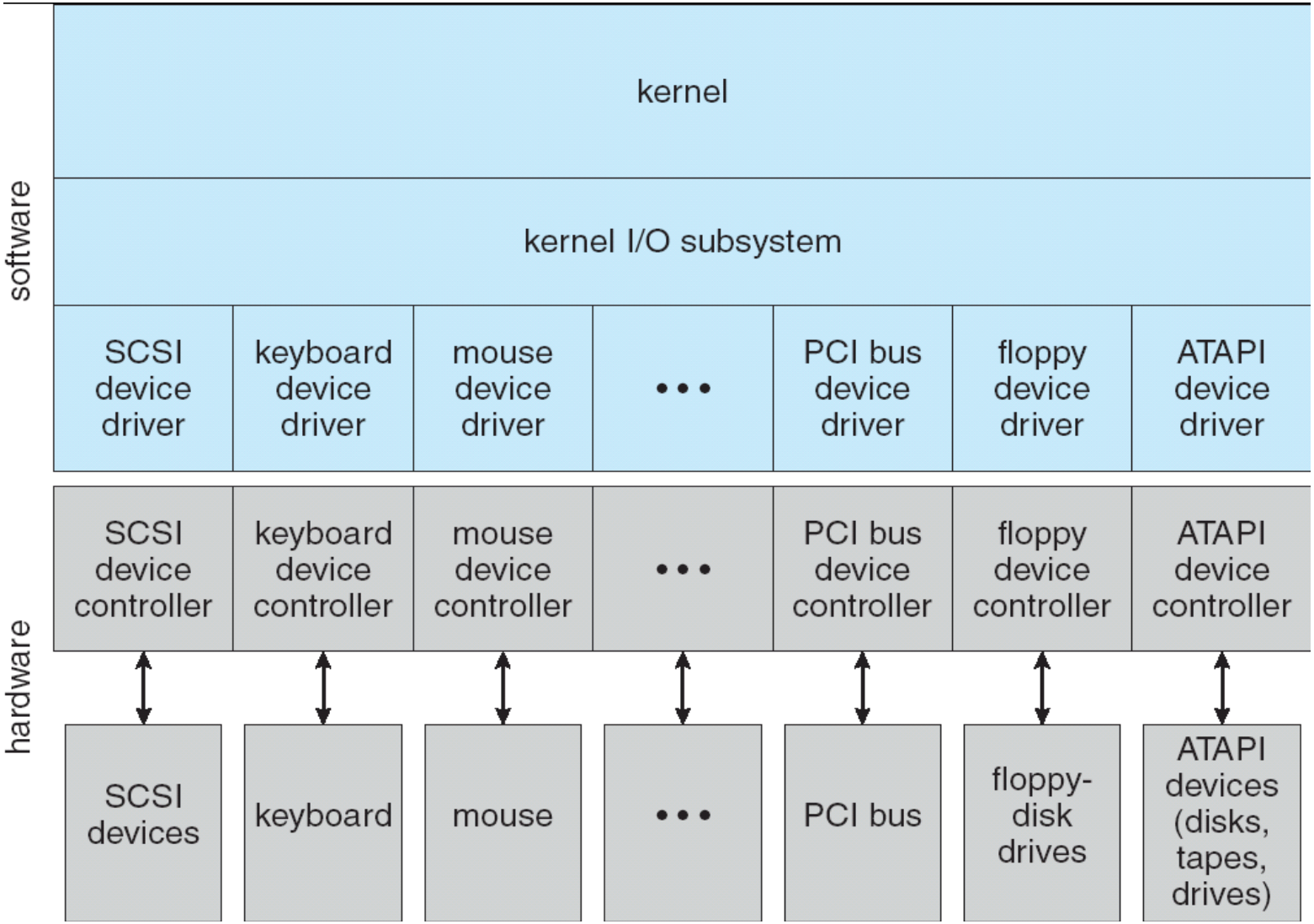
Questão de projeto:

- Como fazer para não ter de reescrever o SO inteiro cada vez que um novo dispositivo é acrescentado?
- Como tratar tantos dispositivos tão diferentes de uma forma (quase) uniforme?
  - ❖ Use os princípios da Eng. de SW ...
    - ◆ Abstração
    - ◆ Encapsulamento
    - ◆ E algumas **camadas** ...

API de E/S

- (Obviamente) As aplicações não interagem diretamente com os dispositivos, cujas características podem variar em diversas dimensões:
  - Orientados a *stream* de caractere ou bloco
  - Acesso sequencial ou randômico
  - Compartilhável ou dedicado
  - Velocidade de operação
  - Somente leitura, somente escrita ou leitura/escrita
- Chamadas ao sistema de E/S encapsulam comportamentos dos dispositivos
- A camada dos ***device-drivers*** esconde do *kernel* as diferenças entre controladoras de E/S

Estrutura de I/O típica do kernel



Características dos dispositivos de I/O

Atributo	Variação	Exemplo
Modo de transferência	caracter bloco	terminal disco
Método de acesso	síncrono assíncrono	fita teclado
Compartilhamento	dedicado compartilhado	fita teclado
Velocidade	latência *seek time* taxa de transferÊncia *delay* entre operações	
Direção de I/O	somente leitura somente escrita leitura-escrita	CD-ROM gráfico disco

Dispositivos orientados a bloco vs. orientados a caracter

- *Drives* de **disco** são orientados a bloco
  - ❖Comandos típicos: ***read***, ***write***, ***seek***
  - ❖E/S bruta: exibe o disco como uma sequência linear de blocos
  - ❖Podem oferecer arquivos mapeados em memória
- Disp. Orientados a caracter incluem o teclado, mouse, porta serial
  - ❖Comandos típicos: `get, put`
  - ❖Bibliotecas podem ser montadas logo acima para permitir edição de linhas

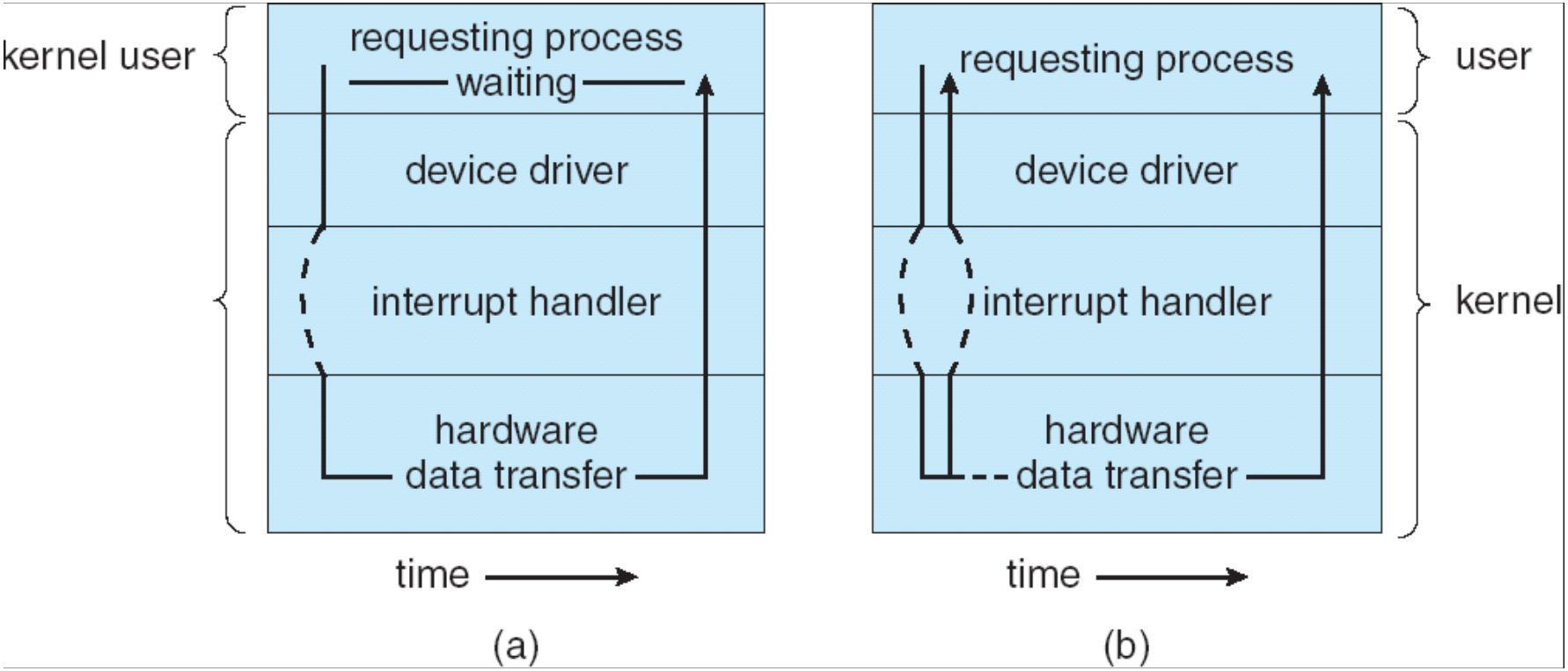
Dispositivos de Rede

- São muito diferentes dos dispositivos de bloco e de caracter, e por isso têm sua própria interface
- Unix e Windows NT/9*x*/2000 implementam a interface *socket*
  - ❖Separação entre o protocolo de rede e a operação da rede
  - ❖Inclui a funcionalidade **`select`**:
    - `select` permite testar (fazer ***polling***) se chegou um pacote p/qq *socket* ou se algum *socket* possui pacote disponível p/envio
- Abordagens variam bastante (*pipes*, *FIFOs*, *streams*, *queues*, *mailboxes*)

E/S Bloqueante vs Não-bloqueante

- **Bloqueante**: o processo é suspenso até que a E/S seja completada
  - ❖ Fácil de usar e de entender seu funcionamento
  - ❖ Inadequada em muitos casos …
- **Não-bloqueante**: chamada não bloqueia a execução da aplicação
  - ❖ retorna imediatamente, com qq quantidade de dados que estejam disponíveis – retorna um contador de bytes lidos
  - ❖ Ex: uma aplicação que lê comandos do usuário enquanto exibe outras informações na tela (*multi-threading*)
- **E/S Assíncrona** – processo não é bloqueado, executa enquanto a E/S é executada
  - ❖ Subsistema de E/S avisa o processo quando a operação completou
  - ❖ Difícil de usar

E/S Bloqueante vs Não-bloqueante

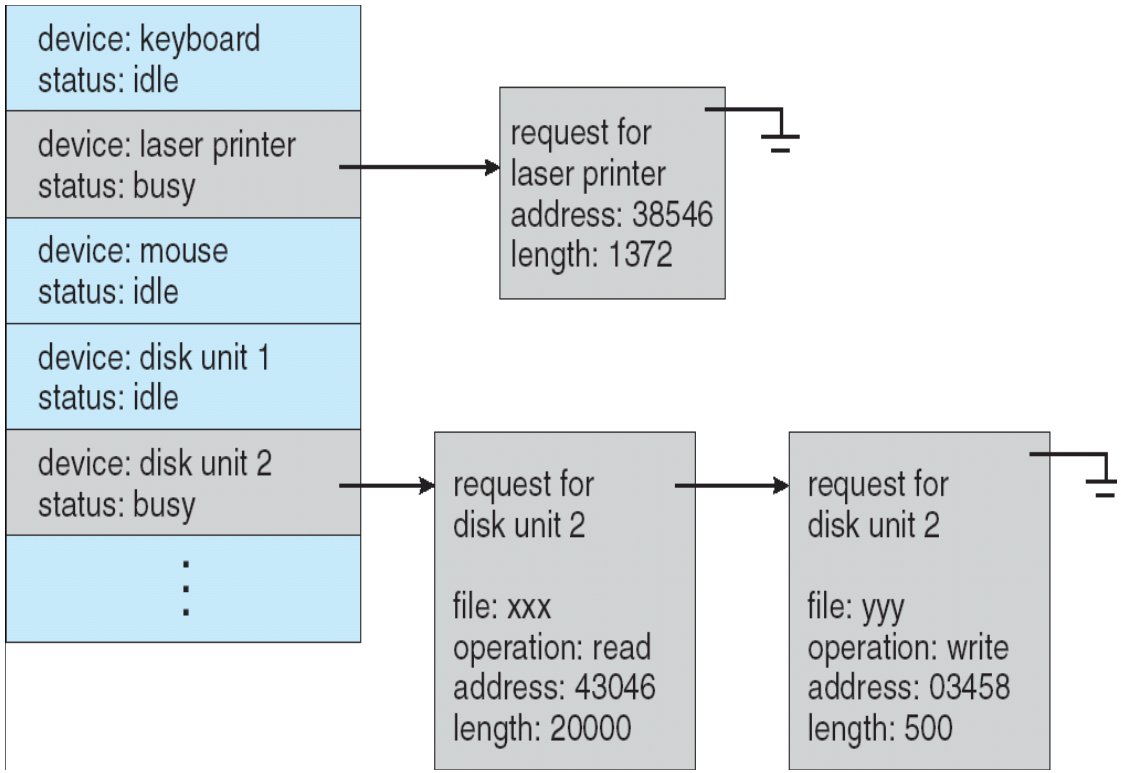


Síncrona

Assíncrona

Operações do subsistema de E/S

- **Escalonamento**: frequentemente, o subsistema de E/S escolhe a ordem em que as requisições devem ser executadas por cada dispositivo.
  - ❖ Critérios: Justiça, melhor desempenho, etc
  - ❖ Ex: escalonamento da cabeça de leitura/escrita do disco



Operações do subsistema de E/S

- **Buferização**: armazenamento temporário de dados em memória enquanto estes são transferidos de entre dois dispositivos
  - ❖Ou entre aplicação e dispositivo
  - ❖Permite acomodar melhor as diferenças de velocidade
    - ◆Ex: Programa que lê registros do disco e envia dados para impressora
  - ❖Permite acomodar melhor as diferenças de tamanho de transferências.
    - ◆Ex: Redes com protocolos diferentes (MTUs)
  - ❖Permite manter a "semântica de cópia"
    - ◆Ex: Gravação de uma cadeia de caracteres que foi modificada em seguida

Operações do subsistema de E/S

- ***Caching***: consiste em guardar uma cópia dos dados em uma memória rápida
  - ❖Melhora o desempenho
  - ❖Cache é **somente uma cópia** … o item sempre reside em outro lugar
- ***Spooling***: armazena temporariamente a saída de um dispositivo
  - ❖Útil nos casos em que o dispositivo só pode servir a uma requisição por vez
  - ❖ Ex: ***spool*** de impressão

Operações do subsistema de E/S

- **Reserva de dispositivo**: há casos em que não é possível multiplexar o acesso a um dispositivo
  - ❖Ex: fita
  - ❖Nesse caso, o subsistema de E/S fornece primitivas para alocação e liberação de dispositivos
  - ❖Certifique-se de que foram liberados no final da operação ...

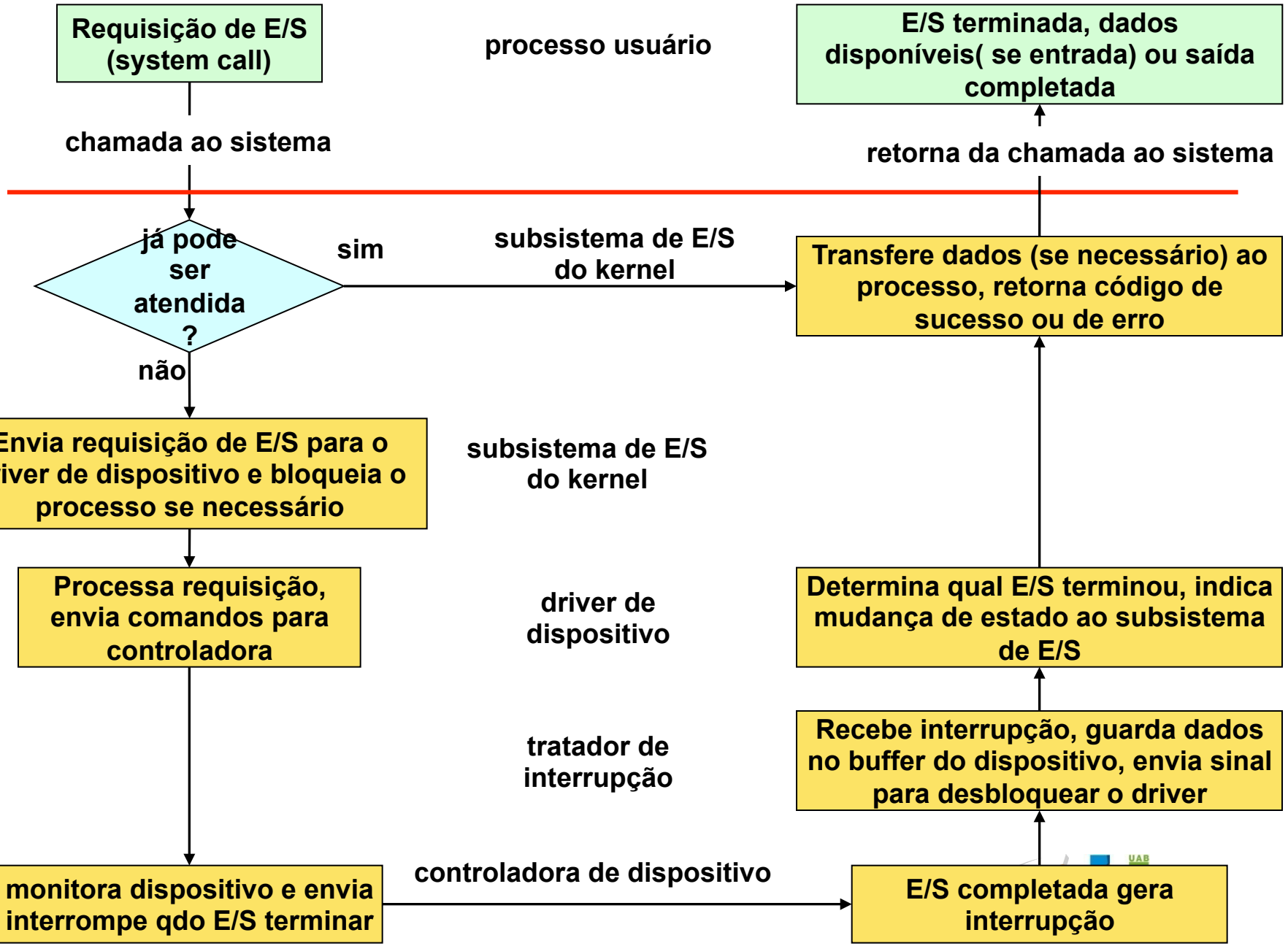
Tratamento de erros

- Operações podem estar sujeitas a falhas.
  - ❖ Falhas podem ser permanentes ou transientes (isto é, passageiras)
  - ❖ Exemplos: Erro de leitura de disco, dispositivo indisponível, falhas de escrita
- Se possível, o subsistema deve tentar mascarar a falha – isto é, tomar alguma providência
  - ❖ Ex: tentar uma nova leitura
- No pior caso, o sistema deve retornar um código de erro para avisar ao usuário/aplicação
  - ❖ Ex: variável **`errno`** do UNIX
- Logs de erros de sistema também podem ser gerados para esclarecer sobre erros ocorridos

Proteção de E/S

- Processos de usuário podem (de forma proposital ou acidental) prejudicar a execução normal de um sistema, emitindo instruções ilegais de E/S
  - Assim, todas as instruções de E/S devem ser definidas como privilegiadas
  - Toda E/S deve ser feita através de chamadas ao sistema (*system calls*)
    - Tanto a E/S mapeada em memória quanto o acesso a portas de E/S devem ser protegidos

O ciclo de vida de uma requisição de E/S



Fim