Sistemas de E/S

MCTA026-13 - Sistemas Operacionais

Emilio Francesquini
e.francesquini@ufabc.edu.br

2019.Q1

Centro de Matemática, Computação e Cognição
Universidade Federal do ABC

UFABC

Disclaimer

UFABC

- Estes slides foram preparados para o curso de **Sistemas Operacionais na UFABC**.
- Este material pode ser usado livremente desde que sejam mantidos, além deste aviso, os créditos aos autores e instituições.
- Este material foi baseado nas ilustrações e texto preparados por Silberschatz, Galvin e Gagne [SGG] e Tanenbaum e Bos [TB] detentores do copyright.
- Algumas figuras foram obtidas em: `http://pngimg.com`

Introdução

Roteiro

UFABC

- Visão geral
- Hardware de E/S
- Interface Aplicação e E/S
- Subsistema de E/S do Kernel
- Transformação de requisições de E/S em operações de hardware
- STREAMS
- Desempenho

Objetivos

UFABC

- Explorar a estrutura do subsistema de E/S de um SO
- Discutir os princípios de hardware de E/S e sua complexidade
- Fornecer detalhes do desempenho tanto do hardware quanto do software que lida com E/S

Visão geral

UFABC

- Gerenciamento de E/S é um componente primordial em um sistema operacional
  - É um aspecto importante do funcionamento de um sistema computacional
  - A variedade de dispositivos de E/S é enorme
    - A variedade de modos de controle também
  - Gerenciamento de desempenho
  - Novos tipos de dispositivos aparecem o tempo todo
- Portas, barramentos, controladores de dispositivos podem ser conectados a diversos dispositivos
- Drivers de dispositivos (*device drivers*) encapsulam os detalhes de cada dispositivo
  - São responsáveis por fornecer uma interface uniforme ao subsistema de E/S do SO

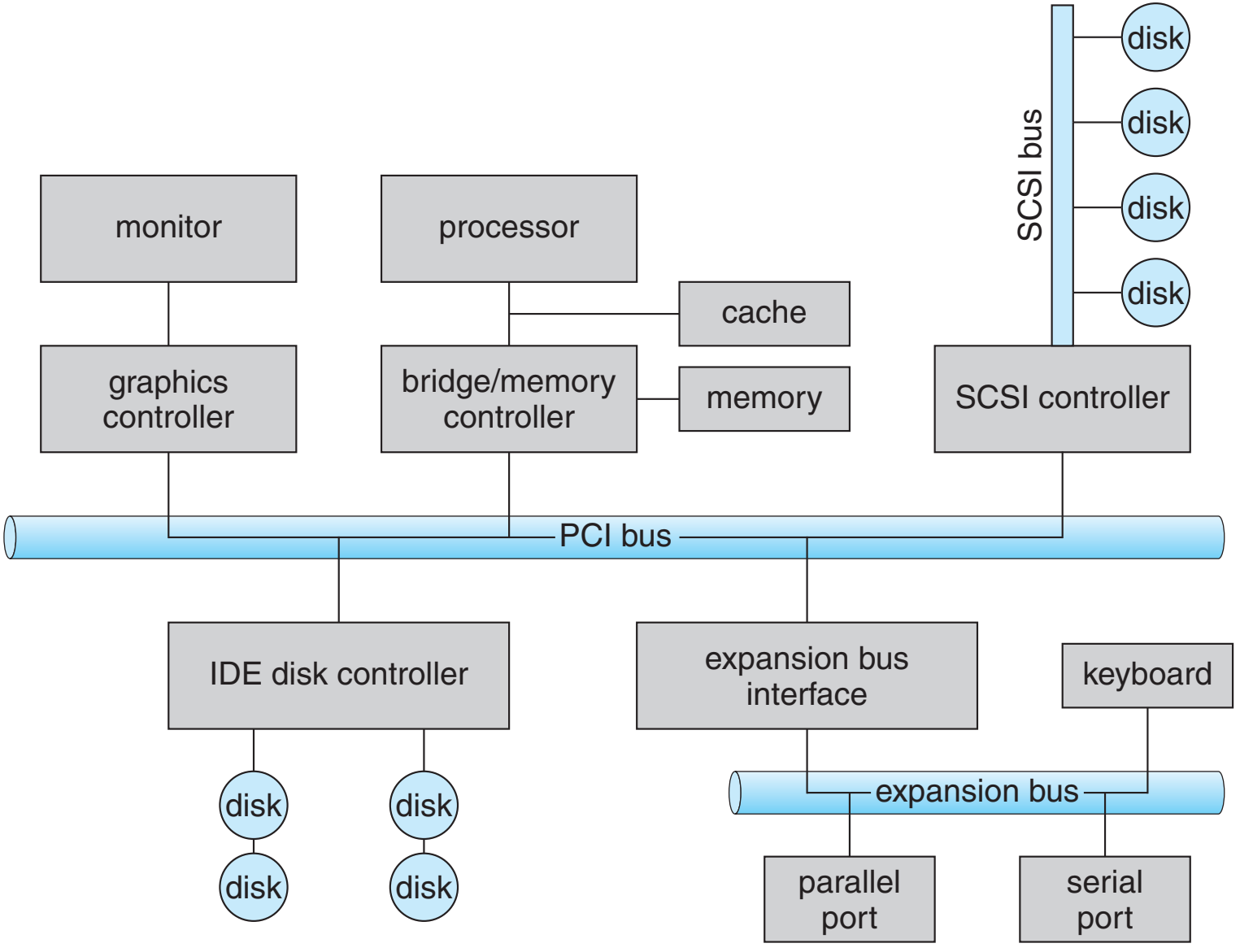
Hardware de E/S

Hardware de E/S

UFABC

- Há uma variedade enorme de dispositivos de E/S: armazenamento, transmissão, interface com humanos
- Todos tem, contudo, algo em comum: sinais entre os dispositivos de E/S e o computador
  - Porta (*port*) - ponto de conexão para um dispositivo
  - Barramento - bus - daisy chain ou acesso direto compartilhado
    - Barramento PCI comum em PCs e servidores, PCI Express (PCIe)
    - Barramento de expansão conectam dispositivos não tão rápidos
  - Controlador (host adapter) - dispositivos eletrônicos que controlam portas, barramentos, dispositivos
    - Algumas vezes integrados
    - Outras vezes são circuitos separados (host adapter)
    - Contêm processador, microcódigo, memória, controlador de acesso ao barramento, etc

Um barramento típico



Hardware de E/S

UFABC

- Instruções de E/S controlam os dispositivos
- Dispositivos têm, tipicamente, registradores onde os drivers de dispositivos colocam comandos, endereços, dados a serem escritos, leem os dados após a execução do comando, ...
  - Registradores de E/S de dados, status, controle
  - Tipicamente de 1 a 4 bytes ou um buffer FIFO
- Os dispositivos tem endereços utilizados por
  - Instruções diretas à memória
  - E/S mapeado em memória
    - Registradores do dispositivo são mapeados ao espaço de endereçamento do processador
    - Especialmente útil para grandes espaços de armazenamento (vídeo, por exemplo)

Localização das entradas e saídas em PCs

UFABC

I/O address range (hexadecimal)	device
000–00F	DMA controller
020–021	interrupt controller
040–043	timer
200–20F	game controller
2F8–2FF	serial port (secondary)
320–32F	hard-disk controller
378–37F	parallel port
3D0–3DF	graphics controller
3F0–3F7	diskette-drive controller
3F8–3FF	serial port (primary)

Lista parcial.

Polling

UFABC

- Para cada byte de E/S
  1. Leia o bit que indica ”ocupado”do registrador de status enquanto diferente de 0
  2. Host ajusta o bit de ”leitura ou escrita”e caso seja escrita escreve os bytes no registro apropriado
  3. Host ajusta o bit ”comando pronto”
  4. Controlador ajusta o bit ”ocupado”e executa a transferência
  5. Controlador limpa o bit ”ocupado”, ajusta o bit ”erro”e ”comando executado”quando a a transferência estiver completa

Polling

UFABC

- O passo um é chamado de laço de espera ocupada (busy wait) para aguardar a E/S do dispositivo
  - É razoavelmente rápido se o dispositivo é rápido
  - Ineficiente se dispositivo é lento
  - CPU pode se focar em outras tarefas?
    - Mas se um ciclo for perdido o dado pode ser sobrescrito/perdido

UFABC

- Polling, no melhor caso, leva 3 instruções
  - Leia status, ”E”lógico para extrair o bit de status, branch se não zero
  - Como ser mais eficiente se não-zero é o caso mais frequente?
- Interrupção da CPU disparada pelo dispositivo de E/S
  - É verificada pelo processador automaticamente após cada instrução

UFABC

- Manejador de interrupção (*Interrupt handler*) recebe as interrupções
  - Maskable - Filtrável
    - Pode ser ajustado para filtrar/ignorar algumas interrupções
- Vetor de interrupções para despachar cada interrupção para o handler apropropriado
  - Há uma troca de contexto no início e no fim
  - Baseado em prioridades
  - Algumas interrupções são nonmaskable
  - Encadeamento de interrupções caso mais de um dispositivo utilize a mesma interrupção

Ciclo de E/S dirigido a interrupções

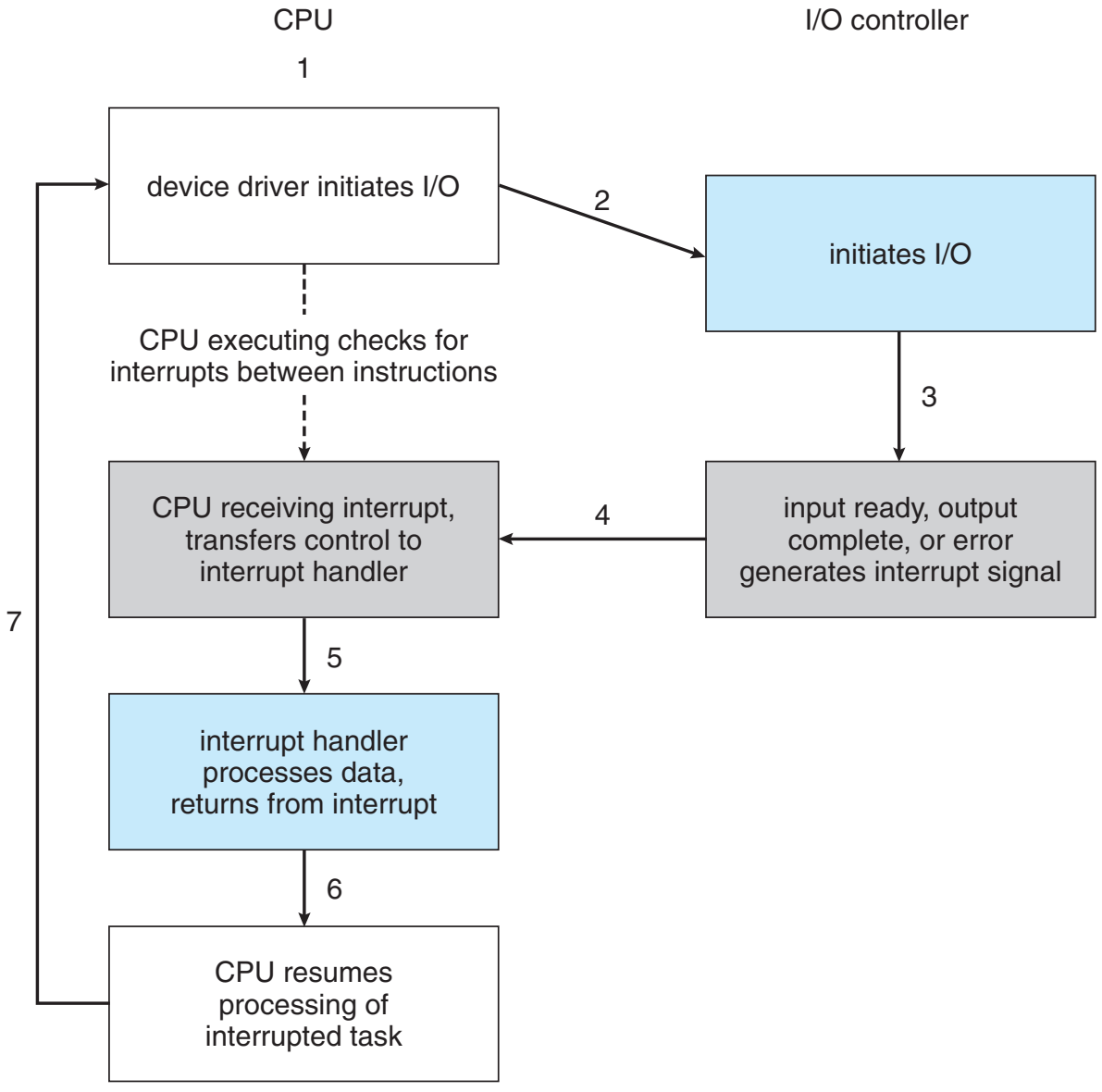


Tabela de eventos do Intel Pentium

UFABC

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

Interrupções

UFABC

- Mecanismo de interrupções também pode ser usado para exceções
  - Pode terminar o processo, *capotar* (*crash*) o sistema em caso de erros de hardware
- Falha de página executa quando há um erro de acesso à memória
- A chamada de sistema executa via um trap para disparar o kernel para executar uma requisição
- Sistemas com múltiplas CPUs podem trabalhar com as interrupções concorrentemente
  - Contanto que o SO tenha sido preparado para isso
- Como interrupções são usadas frequentemente, precisam ser muito rápidas

Direct Memory Access (DMA)

UFABC

- Usada para evitar E/S programada (*programmed I/O*) - um byte por vez para movimentações grandes de dados
- Precisa de um controlador de DMA
- Contorna o uso da CPU para controlar a transferência dos dados **diretamente** entre a memória e o dispositivo de E/S

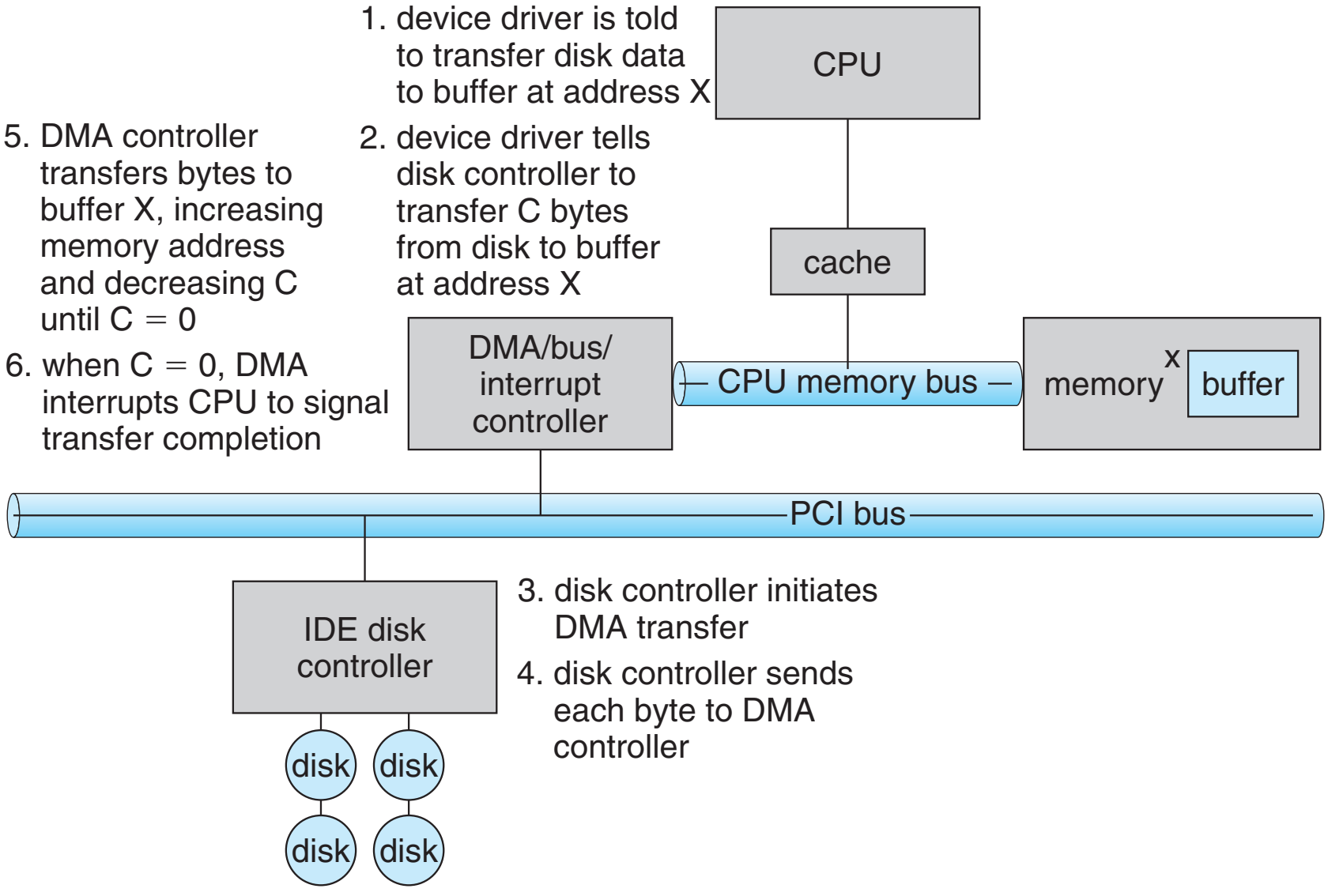
Direct Memory Access (DMA)

UFABC

1. SO escreve o comando diretamente na memória
   - Comando é composto de: endereço origem e destino, modo leitura ou escrita, número de bytes
2. Escreve no controlador o endereço do comando na memória
3. Controlador de DMA assume o comando do barramento
   - Roubo de ciclos da CPU, mas ainda assim é muito mais eficiente
   - Quando finalizado interrompe a CPU para que esta processe os dados

- Há uma variação que leva em conta endereços virtuais que pode ser ainda mais eficiente DVMA

Passos para uma operação de DMA



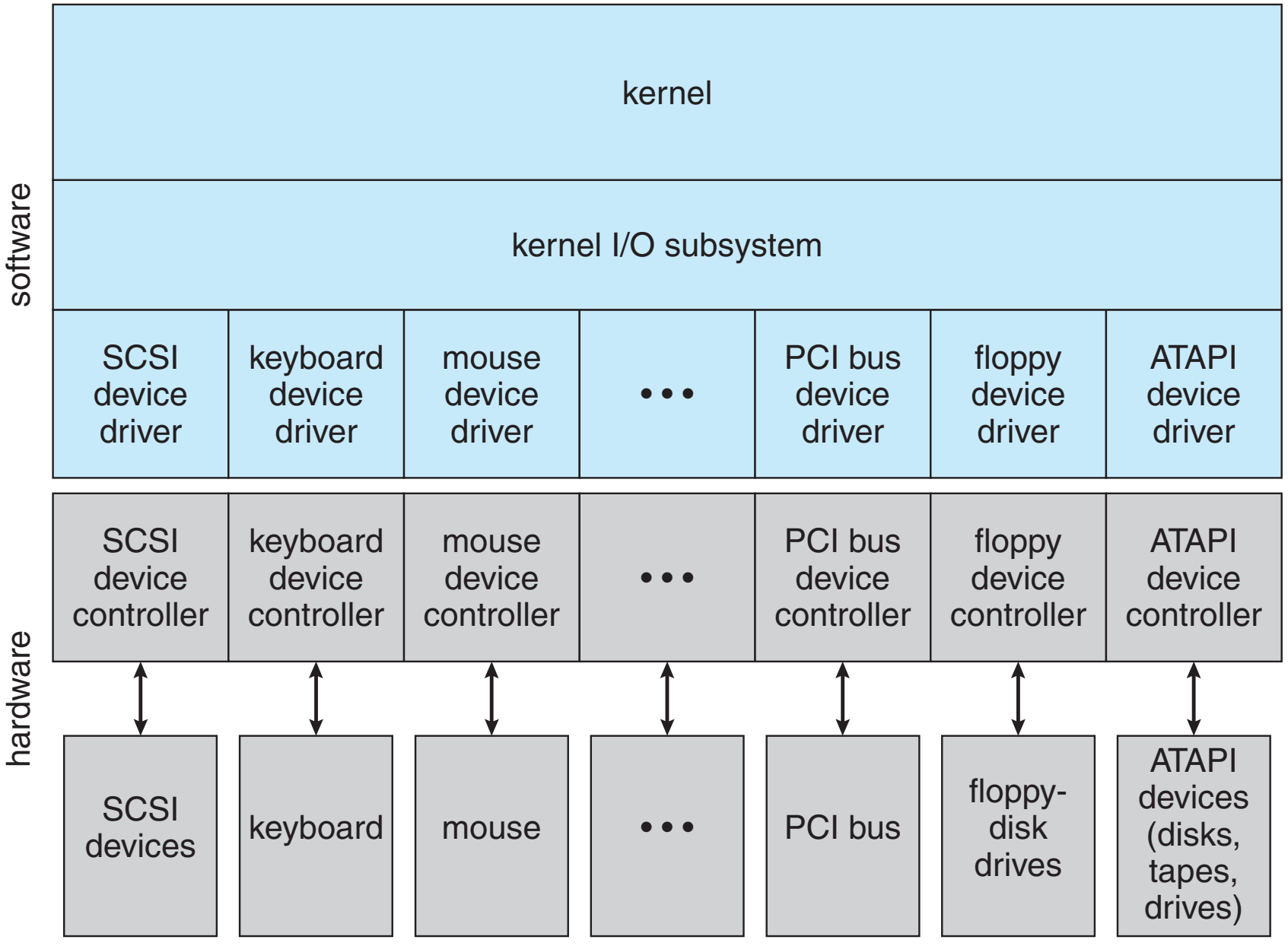
Interface de E/S para aplicações

Interface de E/S para aplicações

UFABC

- Chamadas de sistema de E/S encapsulam os diferentes comportamentos dos dispositivos em classes genéricas
- Drivers de dispositivo escondem as diferenças entre os controladores de E/S do kernel
- Novos dispositivos que utilizem protocolos de comunicação já estabelecidos não precisam de nenhum trabalho extra
- Cada SO tem o seu próprio sistema de E/S e arcabouços de drivers
- Dispositivos podem variar em diversas dimensões
  - Caracteres ou bloco
  - Acesso sequencial ou aleatório
  - Acesso síncrono ou assíncrono (ou ambos)
  - Compartilhável ou dedicado
  - Velocidade de operação
  - Apenas leitura, apenas escrita, ou ambos

A estrutura de E/S do kernel



Características dos dispositivos de E/S

UFABC

aspect	variation	example
data-transfer mode	character block	terminal disk
access method	sequential random	modem CD-ROM
transfer schedule	synchronous asynchronous	tape keyboard
sharing	dedicated sharable	tape keyboard
device speed	latency seek time transfer rate delay between operations	
I/O direction	read only write only read–write	CD-ROM graphics controller disk

Características dos dispositivos de E/S

UFABC

- Sutilezas dos dispositivos tratadas pelos drivers
- De maneira geral, podemos agrupar os dispositivos nas seguintes categorias:
  - E/S em blocos
  - E/S em caracteres (*stream*)
  - Acesso mapeado em memória
  - Soquetes de rede
- Para manipulação direta das características dos dispositivos de E/S utiliza-se, normalmente, um *escape* ou um *backdoor*
  - Nos UNIX `ioctl()` envia bits arbitrários para o registrador de controle do dispositivo e dados para o registrador de dados

Dispositivos de bloco e de caracteres

UFABC

- Discos são dispositivos de bloco
  - Comandos incluem leitura, escrita e busca (*seek*)
  - E/S crua (*raw*), E/S direta (*direct I/O*), ou acesso ao sistema de arquivos
  - Acesso de arquivos mapeados em memória são possíveis
    - Páginas são trazidas sob demanda para a memória
  - DMA
- Dispositivos de caracteres incluem teclados, mouses, portas seriais
  - Commandos incluem `get()` e `put()`
  - Bibliotecas possibilitam edições de linhas

Dispositivos de rede

UFABC

- Suficientemente diferentes de blocos e caracteres para merecerem a sua própria interface
- Linux, UNIX, Windows e muitos outros SOs têm uma interface de soquetes (*socket*)
  - Separam o protocolo de rede da operação da rede
  - Incluem a funcionalidade `select()`
- As abordagens variam bastante (pipes, FIFOs, streams, queues, mailboxes, ...)

UFABC

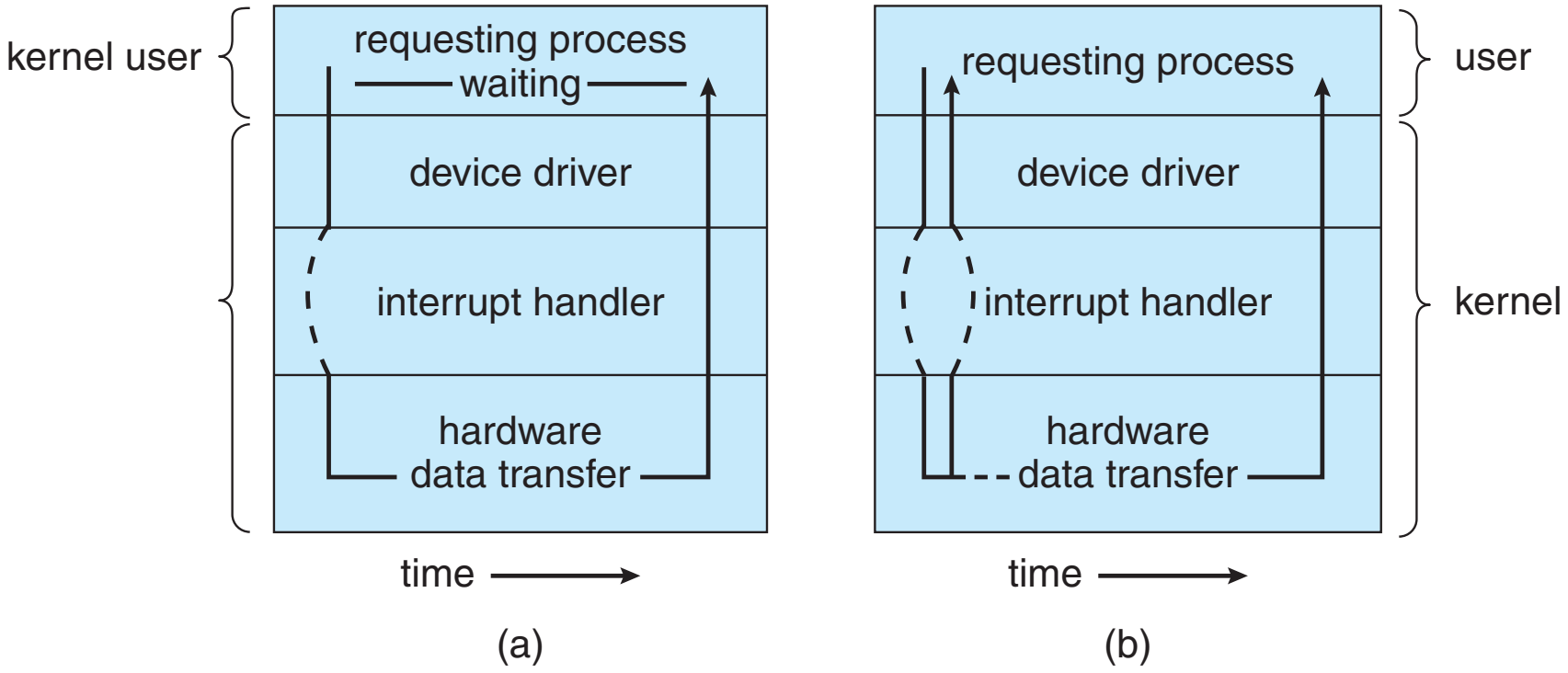
- Fornecem tempo atual, tempo decorrido, timers, ...
- Resolução normal é de 1/60 segundo
- Alguns sistemas fornecem timers de alta-resolução
- Timers de intervalos programáveis (*programmable interval timers*) são usadas para contar o tempo, interrupções periódicas, ...
- `ioctl()` no UNIX é usado para lidar com os aspectos menos regulares dos relógios e timers

E/S não bloqueante e E/S assíncronas

UFABC

- Bloqueante - processo é suspenso até a operação de E/S ser completada
  - Fácil de usar e de entender
  - Insuficiente em alguns casos
- Não bloqueante - chamada E/S devolve o controle assim que possível
  - Interface com usuário, cópia de dados (E/S com buffers)
  - Implementada com multi-threading
  - Devolve o controle o mais rápido possível com o número de bytes já lidos ou escritos
  - `select()` para localizar os dados prontos e `read()` e `write()` para transferir
- Assíncrono - processo executa enquanto a operação de E/S executa
  - Difícil de executar
  - Sistema de E/S avisa o processo quando a operação de E/S tiver completado

2 métodos de E/S



E/S Vetorizada

UFABC

- E/S vetorizada (*vectored I/O*) permite que uma chamada de sistema para diversas operações de E/S
- Por exemplo, a chamada `readve()` aceita um vetor de múltiplos buffers a serem lidos ou escritos
- Este método de *scatter/gather* é melhor que múltiplas chamadas individuais
  - Diminui o número de trocas de contexto e de overhead do SO
  - Algumas versões proveem atomicidade
    - Diminui a preocupação de diversos threads escrevendo ou lendo ao mesmo tempo

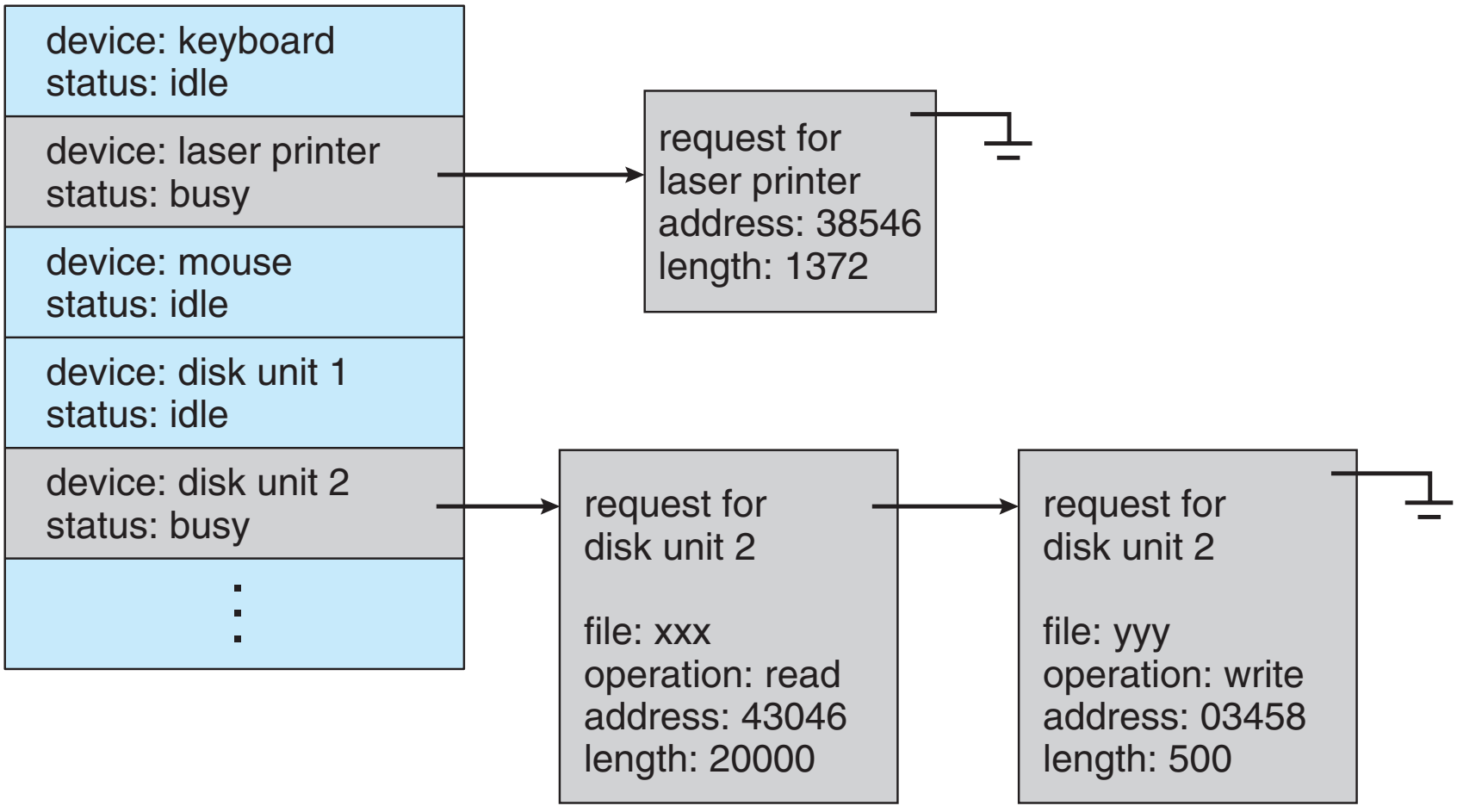
Subsistema de E/S do Kernel

Subsistema de E/S do kernel

UFABC

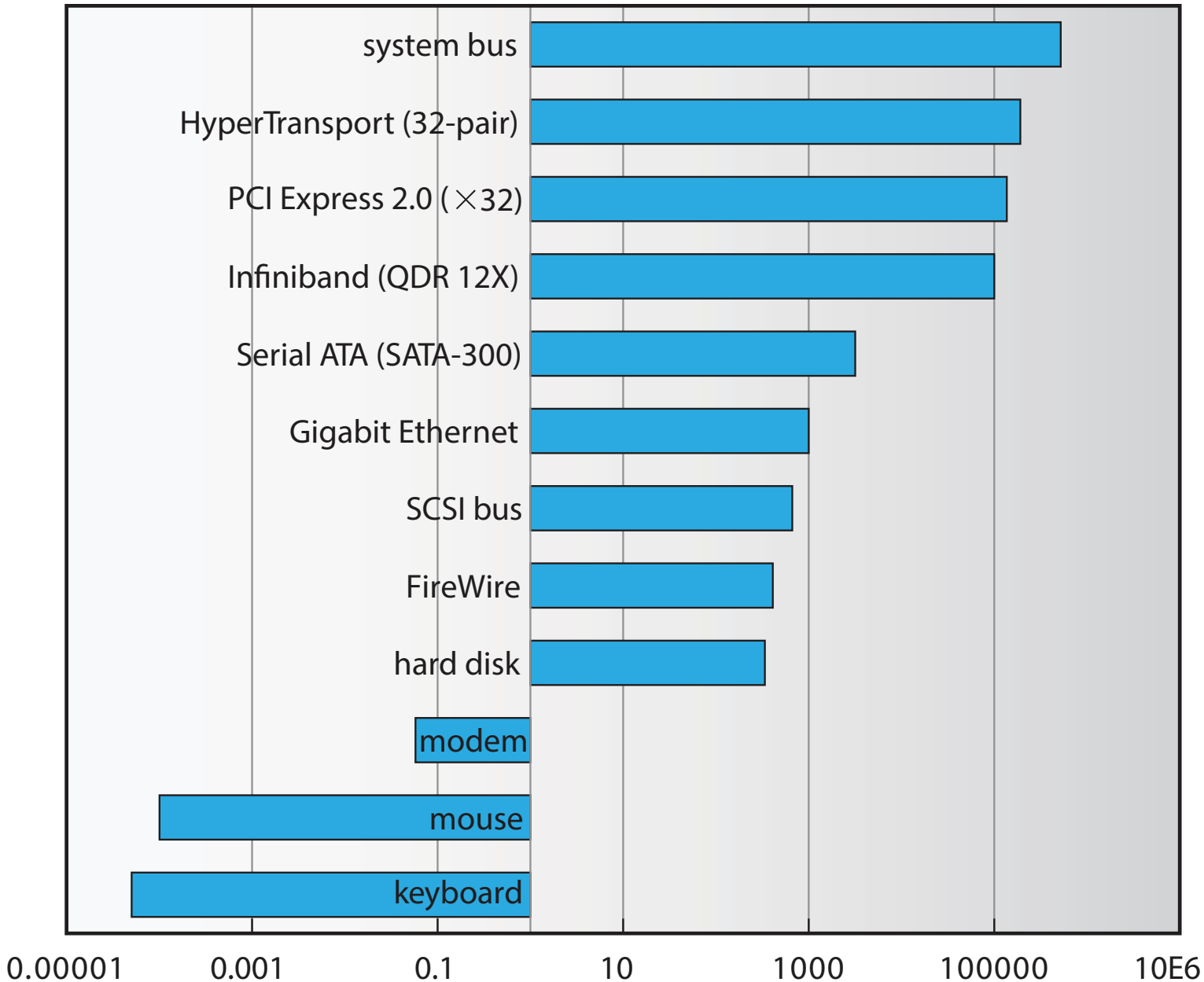
- Escalonamento (*scheduling*)
  - Alguns pedidos de E/S são ordenados através de uma fila por dispositivo
  - Alguns SOs tentam ser justos
  - Alguns SOs implementam QoS
- Buffering - Armazenamento de dados em memória enquanto a transferência entre dispositivos ocorre
  - Ajuda a lidar com a diferença
    - de desempenho entre dispositivos
    - de tamanhos (de blocos, por exemplo)
  - Mantém a semântica de cópia
  - Double buffering - duas cópias do dado
    - Kernel e usuário
    - Tamanhos distintos
    - Completo/Incompleto e Incompleto/Em uso
    - *Copy-on-write* pode ser utilizado para aumentara a eficiência em alguns casos

Tabela de status de dispositivos



Sun Enterprise 6000 - Taxas de transferência

UFABC



UFABC

- Caching - Dispositivo mais rápido mantém uma cópia dos dados
  - Sempre uma única cópia
  - Chave para desempenho
  - Pode ser combinado com buffering
- Spooling - Represa a saída para um dispositivo
  - Se o dispositivo só pode lidar com uma requisição por vez. Ex. impressão
- Reserva de dispositivos - Fornece acesso exclusivo a um dispositivo
  - Chamadas de sistema para alocação e desalocação
  - Cuidado com impasses!

Tratamento de erros

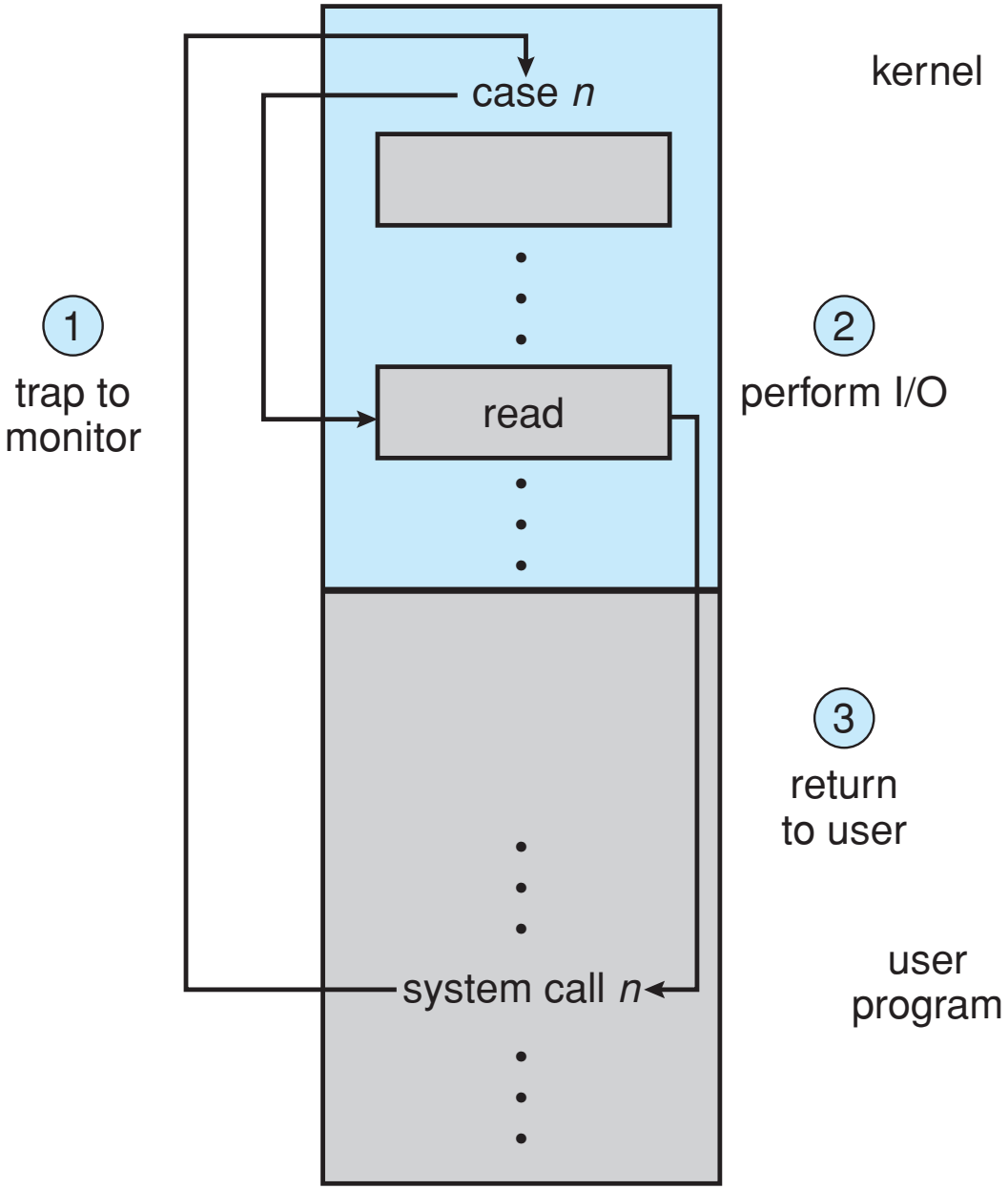
UFABC

- SO pode se recuperar de erros de leituras em disco, indisponibilidade de dispositivos, erros transientes de E/S, ...
  - Pode simplesmente tentar a operação novamente
  - Alguns sistemas tem tratamentos mais eficientes - Solaris DMA, AIX
    - Mantém um histórico das falhas e tendem a evitar dispositivos com altas taxas de erros (recuperáveis)
- A maior parte dos dispositivos informa um código de erro quando uma requisição falha
- SOs mantém os erros em um log do sistema

Proteção de E/S

- Usuários podem, inadvertidamente ou não, fazer algo que possa atrapalhar o funcionamento do sistema através de chamadas de E/S inválidas
  - Todas as instruções de E/S são portanto privilegiadas
  - E/S precisa ser desempenhado por chamadas de sistema
    - Endereços de E/S mapeados em memória também precisam ser protegidos

Uso de syscalls para E/S

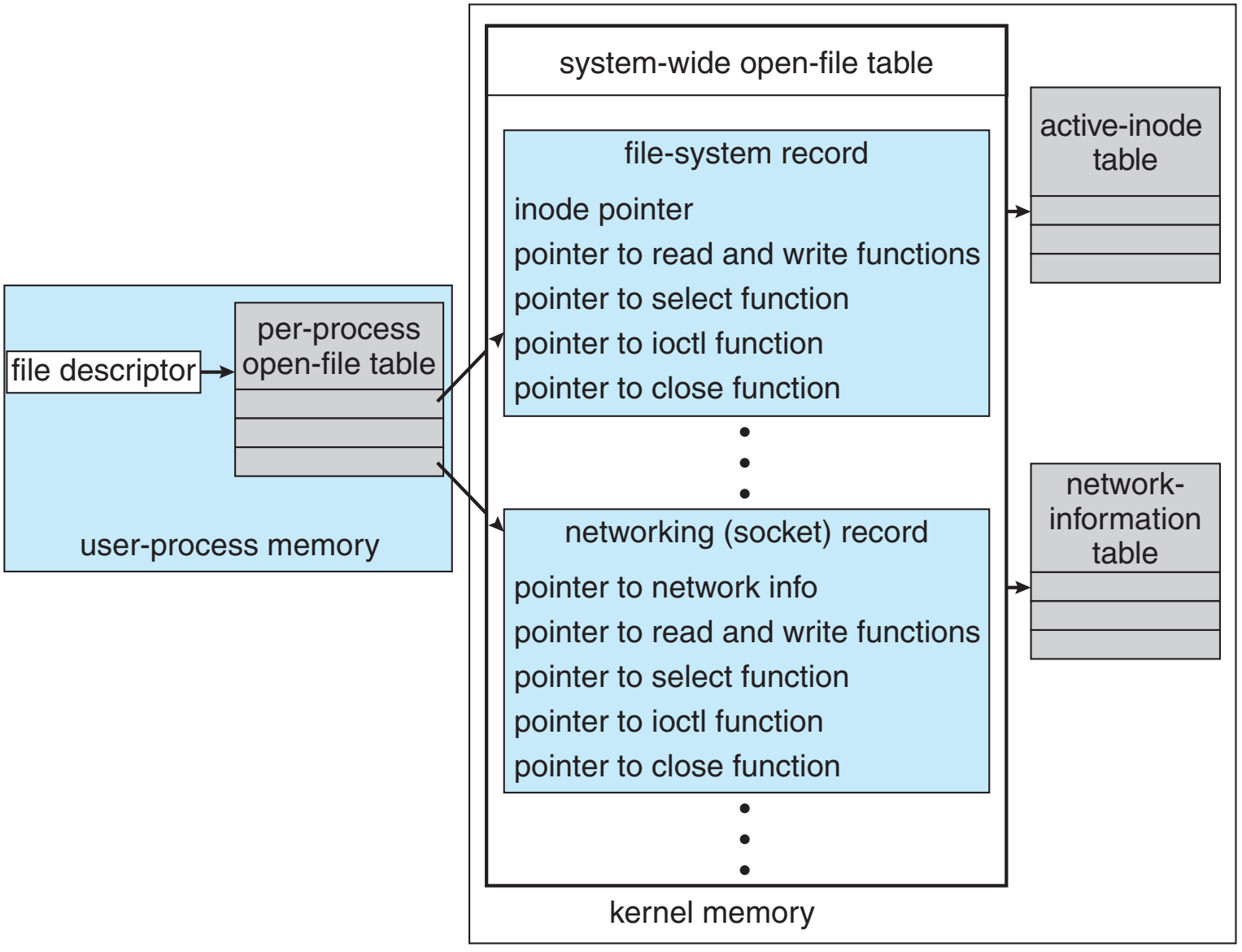


Estruturas de dados do kernel

UFABC

- O kernel precisa manter informações de estado para os componentes de E/S, incluindo tabelas de arquivos abertos, conexões de rede, estado dos dispositivos, etc
- São utilizas muitas estruturas de dados complexas para manter o estado de buffers, alocação de memória, blocos ”sujos”, ...
- Alguns SOs são orientados a objetos e usam trocas de mensagens para implementar E/S
  - Windows é um exemplo
    - Mensagens com informações de E/S são passadas entre o kernel e o modo de usuário
    - Mensagens modificadas conforme passam pelo driver (indo e vindo)
    - Quais as vantagens e desvantagens?

Estrutura de E/S do UNIX



Gerenciamento de energia

UFABC

- Não é exatamente E/S, mas tem alta relação
- Computadores e dispositivos eletrônicos em geral, geram calor e não raramente precisam de mecanismos de refrigeração
- SOs podem ajudar a gerenciar o seu uso
  - SOs para clouds podem migrar VMs entre máquinas físicas
    - Podem assim deligar máquinas físicas completas
- Computação móvel tem como o consumo de energia um dos pontos principais da sua execução

UFABC

- Gerenciamento de energia no nível de componentes
  - Entende o relacionamento entre os componentes
  - Constrói uma árvore de dispositivos que representa a topologia física dos dispositivos
  - Barramento do sistema $\rightarrow$ subsistema de E/S $\rightarrow$ {flash, USB, ...}
  - Driver controla o estado do dispositivo e se em uso
  - Componentes não utilizados $\rightarrow$ desligamento
  - Todos os dispositivos em um ramo da árvore inativos $\rightarrow$ desliga ramo

UFABC

- Wake locks - Previnem o desligamento de um dispositivo quando estiver em uso
- Power collapse - colocam um dispositivo em um estado de suspensão profundo
  - Uso de energia mínimo
  - Mínimo necessário para que o dispositivo possa responder a um evento externo (pressionamento de um botão, chamada telefônica, ...)

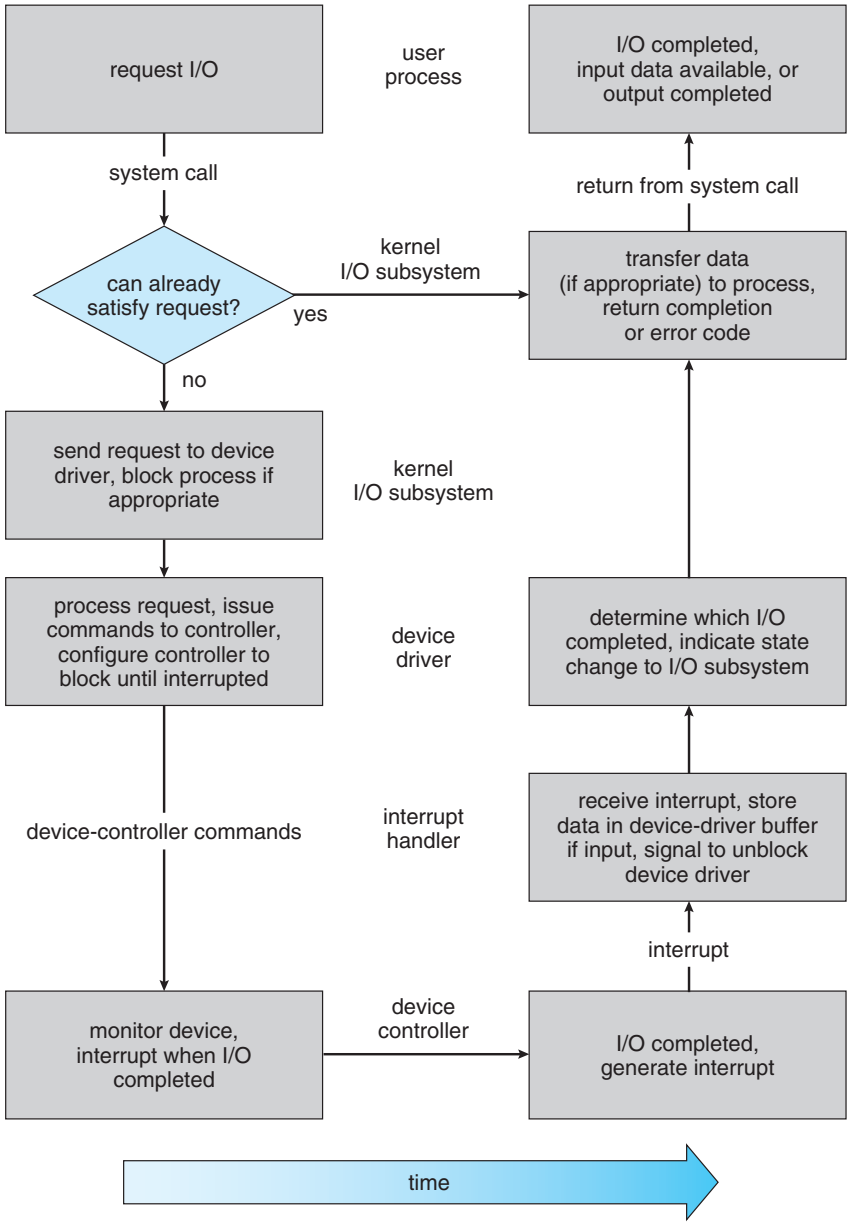
Requisições de E/S $\rightarrow$ Operações em hardware

Requisições de E/S $\rightarrow$ Operações em hardware

UFABC

- Considere a leitura de um arquivo do disco para um processo
  - Determinar qual dispositivo contém o arquivo
  - Traduzir de nomes entre o sistema de arquivos e o dispositivo
  - Ler dados do disco em um buffer
  - Disponibilizar os dados ao processo requisitante
  - Devolver o controle ao processo

Ciclo de vida de uma requisição de E/S

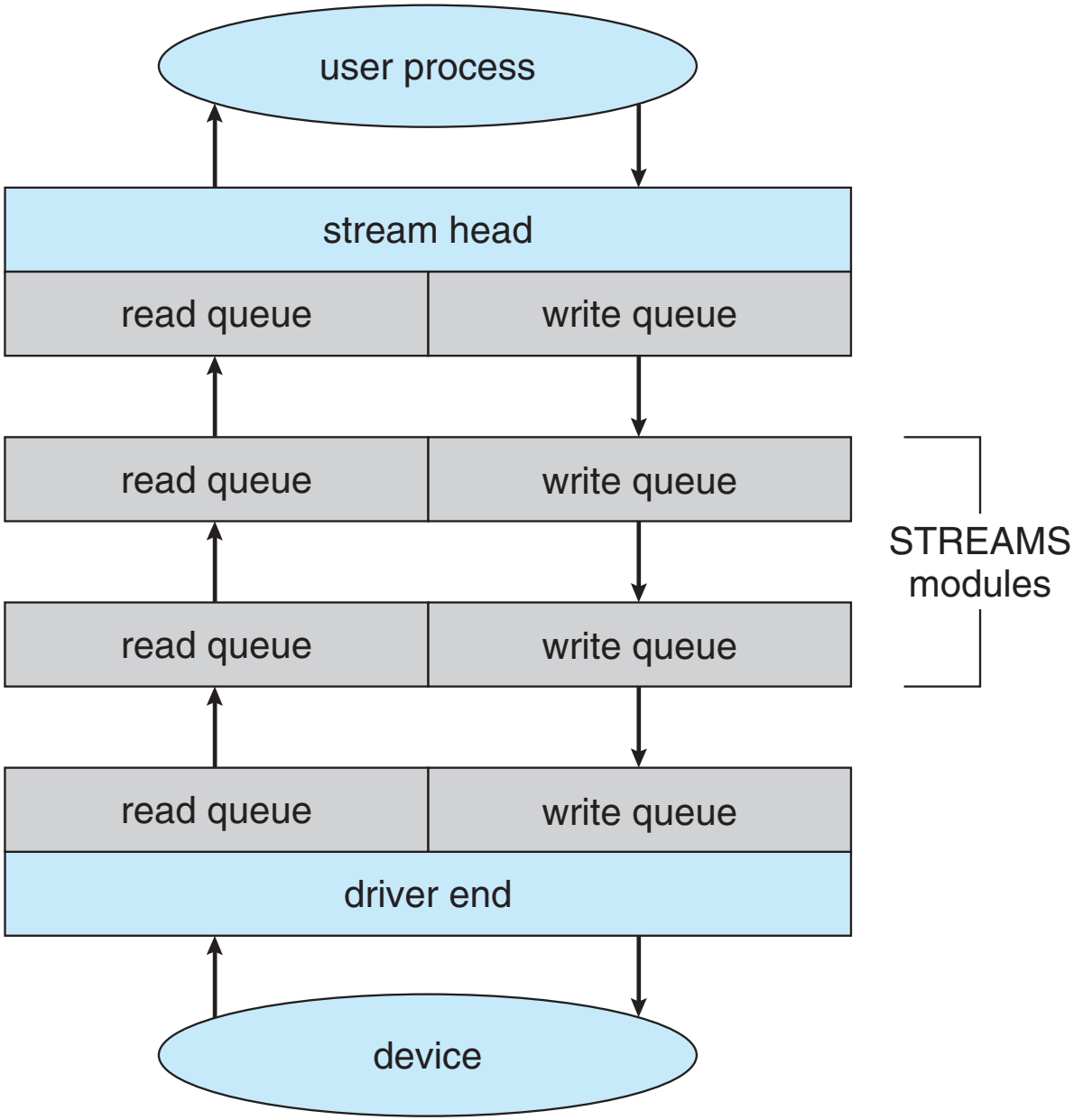


STREAMS

UFABC

- Um STREAM é um canal de comunicação full-duplex entre um processo executando no nível do usuário e um dispositivo no UNIX System V em diante
- Consiste de
  - Interfaces com o processo do usuário
  - Interfaces com o dispositivo de E/S
  - Zero ou mais módulos entre o processo e o dispositivo
- Cada módulo contém uma fila de leitura e outra de escrita
- Trocas de mensagens são utilizadas para comunicar entre as filas
  - Controle de fluxo é utilizado para indicar a disponibilidade ou ocupação de um dispositivo
- Internamente assíncrono, síncrono na visão do usuário quando se comunica com a interface

Estrutura do STREAMS



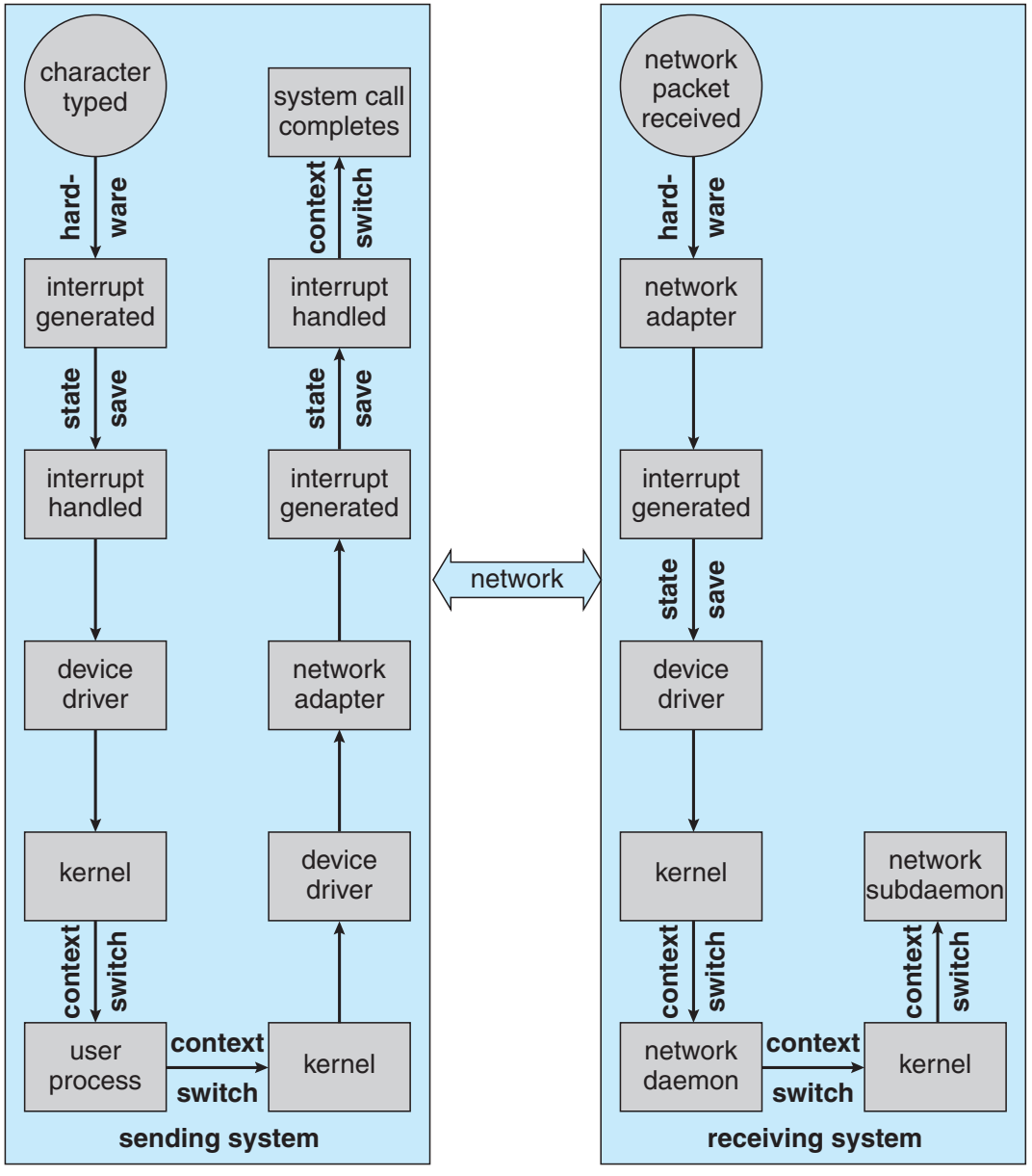
Desempenho

Desempenho

UFABC

- E/S é um dos fatores mais importantes no desempenho
  - Demanda a execução pela CPU do driver, código E/S do kernel
- Trocas de contexto devido às interrupções
- Cópia de dados
- Tráfego de rede é particularmente sensível

Comunicações intercomputadores



Melhorando o desempenho

UFABC

- Diminuição no número de trocas de contexto
- Redução na cópia de dados
- Redução nas interrupções através de transferências maiores, controladores inteligentes, polling
- DMA
- Dispositivos de hardware mais ”inteligentes”
- Equilíbrio no desempenho entre CPU, memória, barramento e E/S para aumento da vazão
- Transferência de processos em espaço de usuário para daemons e threads do kernel

Progressão da funcionalidade dos dispositivos