

SISTEMAS DE I/O

UC: Sistemas para Internet

Profª

Alberto Felipe Friderichs Barros



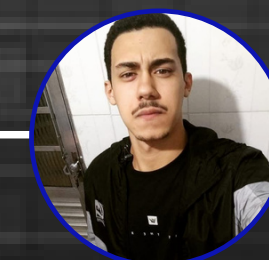
Cecília Corrêa Klein

Aluna :



Dener Ferreira Machado de Oliveira

Aluno:



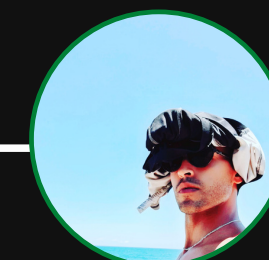
Pedro Francisco Espindola

Aluno:



Alan Silva Gonçalves da Costa

Aluno:



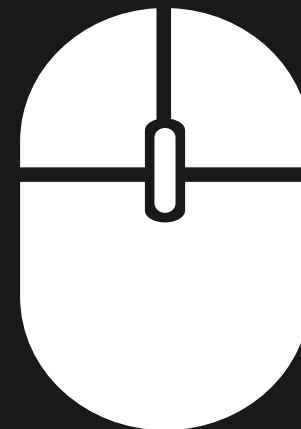
OBJETIVO DO CAPÍTULO 13

- Explorar a estrutura do **subsistema de I/O** de um **sistema operacional**
- Discutir os **princípios** e as **complexidades** do hardware de I/O
- Explicar os **aspectos** de desempenho do **hardware** e do **software de I/O**

13.1 VISÃO GERAL

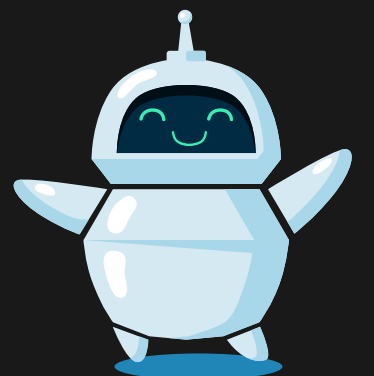
- A **Preocupação** dos projectista de S.O é manter o **controle** de dispositivos **conectados no PC**.
- **Porque?** os dispositivos de I/O variam desde a **função** e **velocidade**

considere:



mouse

Disco rígido



Robô de fita

- **É necessário métodos variados** para controlá-los.
- **Esses métodos formam** o subsistema de I/O do **Kernel** que **separa o resto do kernel** das **complexidades** de **gerenciamento** dos dispositivos de I/O.

- A **tecnologia** de dispositivos de I/O **exibe duas tendências conflitantes**.

- **Crescente padronização** de interfaces de **software e hardware**

- Essa **tendência** incorpora gerações de **dispositivos aperfeiçoados** em **PC e SO** existentes.

- **Vemos variedades** maiores de dispositivos de I/O.

- **Alguns dispositivos novos** é **tão diferentes dos anteriores** sendo **desafiador incorporar** aos **PC e S.O.**

- Esse desafio é superado por **combinação técnicas** de hardware e software.

- Os **elementos** de hardware de I/O como **portas, bases e controladores** acomodam **variedades** de dispositivos.

- O **kernel** é estruturado para usar **módulos de drivers** de dispositivos.

- Os **drivers** apresentam uma **interface - padrão** de acesso aos dispositivos para o **subsistema** de I/O.

13.2 Hardware de I/O

- Os **aspectos** do hardware de I/O é **complexos** no nível de detalhes do projeto de **componentes eletrônicos**.
- Os **conceitos** são suficiente para nos habilitar a entender os recursos de I/O do S.O.
- Os **conceitos** principais:
 - Um **bus**;
 - Um **controlador**;
 - Uma porta de I/O e seus **registradores**;
 - O **relacionamento** de aperto de mãos entre o **hospedeiro** e um **controlador** de dispositivos;

13.2 Hardware de I/O

A **execução** desse aperto de mãos em um **loop** de **sondagem** ou por meio de **interrupções**;

A **delegação** desse trabalho a um **controlador** de **DMA** para grandes transferência.

- Cada **tipo** de **dispositivo** tem seu próprio **conjunto** de **recursos**.
- **Definições** de **bits** de **controle** e **protocolos** para **interação** com o **hospedeiro** - sendo todos diferentes.

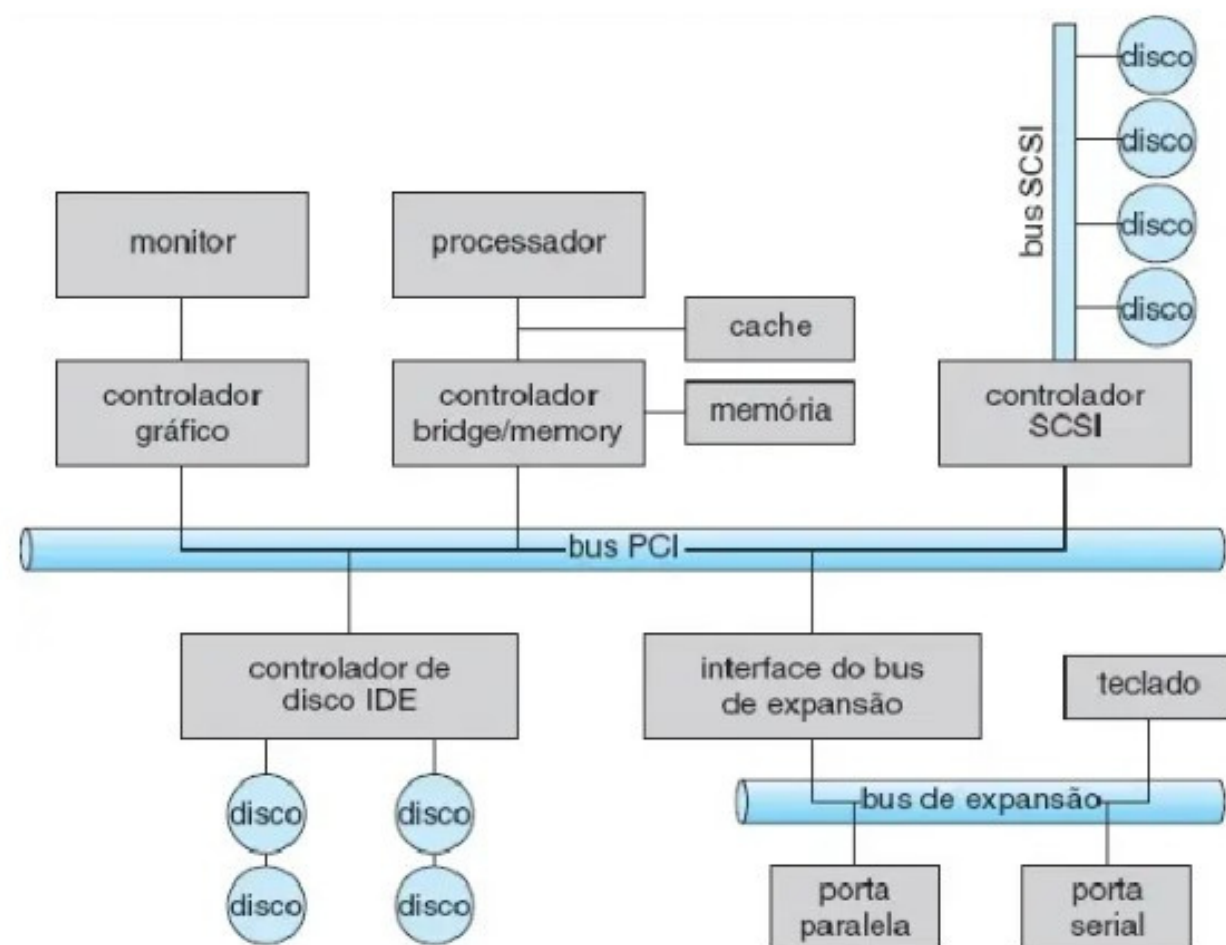


Figura 13.1 Uma estrutura típica de bus de PC.

- Um **bus de expansão** (o bus comum em um sistema Pc) conecta o **subsistema processador-memória** aos dispositivos **rápidos**, e um **bus de expansão** conecta dispositivos relativamente lentos, como o **teclado** e as **portas serial** e **USB**. Na parte **superior direita** da figura, **quatro discos** estão conectados, juntos, em um **bus Small Computer System Interface (SCSI)** conectado a um **controlador SCSI**. Outros **buses** comuns usados para interconectar as partes **principais** de um **PC** incluem o **PCI Express (PCIe)**, com **throughput** de até **16GB** por segundo, e o **Hyper-transport**, com **throughput** de até **25 GB** por segundo.

intervalo de endereços de I/O (hexadecimal)	dispositivo
000–00F	controlador de DMA
020–021	controlador de interrupções
040–043	timer
200–20F	controlador de jogos
2F8–2FF	porta serial (secundária)
320–32F	controlador de disco rígido
378–37F	porta paralela
3D0–3DF	controlador gráfico
3F0–3F7	controlador de drive de disquete
3F8–3FF	porta serial (primária)

Figura 13.2 Locações de portas de I/O de dispositivos nos PCs (parcial).

- Mostra os endereços usuais de **porta de I/O** para **PCs**.
- portal de I/O consiste, em **quatro registradores** de **statuts**, **controle**, **dados de entrada** e **dados de saída**.

13.2.1 Sondagem (Polling)

Aperto de mãos.

- Suponha que **2 bits** sejam usados para coordenar o relacionamento **produtor-consumidor** entre o **controlador** e o **hospedeiro**.
- O **controlador** indica seu estado através do **bit busy** no registrador **status**.(Lembre que **ligar** um **bit** significa gravar um **1 no bit**, e **desligar** um **bit** significa gravar um **0 no bit**).
- O **controlador** **liga** o **bit busy** quando está ocupado trabalhando e **desliga** o **bit busy** quando está pronto para aceitar o próximo comando.
- O **hospedeiro** sinaliza o que deseja através do **bit command-ready** no registrador **command**.
- O **hospedeiro** **liga** o **bit command-ready** quando um comando está disponível para a **execução** pelo **controlador**.
- Nesse exemplo, o **hospedeiro** grava a **saída** por meio de uma **porta**, coordenando-se com o **controlador** por meio do aperto de mãos.

13.2.2 Interrupções

- O **hardware da CPU** tem um **fio** chamado **linha de solicitação de interrupção** que a **CPU** examina após **executar** cada **instrução**.
- Quando a **CPU** detecta que um **controlador** confirma um sinal na linha de **solicitação de interrupção**, ela **executa** um **salvamento de estado** e salta para a **rotina de manipulação de interrupções** em um endereço **fixo** na **memória**.
- O **manipulador de interrupções** determina a **causa da interrupção**, **executa** o **processamento** necessário, realiza uma **restauração de estado** e **executa** uma **instrução return from interrupt** para retornar a **CPU** ao estado de **execução** anterior à **interrupção**.

13.2.3 Acesso Direto à Memória

- Para um dispositivo que faz grandes transferências, como, por exemplo, um drive de disco, parece desperdício usar um caro processador de uso geral na verificação de bits de status e na alimentação de dados, byte a byte, em um registrador de controlador — um processo chamado I/O programado (PIO — programmed I/O).
- Muitos computadores evitam sobrecarregar a CPU principal com PIO, descarregando parte desse trabalho para um processador de uso específico chamado controlador de acesso direto à memória (DMA — direct-memory-access). Para iniciar uma transferência DMA, o hospedeiro grava um bloco de comando DMA na memória.

13.2.4 Resumo do Hardware de I/O

- Embora os aspectos do hardware de I/O sejam complexos quando considerados no nível de detalhe do projeto de componentes eletrônicos, os conceitos que acabamos de descrever são suficientes para nos habilitar a entender muitos recursos de I/O dos sistemas operacionais. Vamos revisar os conceitos principais:
- Um controlador
- Uma porta de I/O e seus registradores
- O relacionamento de aperto de mãos entre o hospedeiro e um controlador de dispositivos
- A execução desse aperto de mãos em um loop de sondagem ou por meio de interrupções
- A delegação desse trabalho a um controlador de DMA para grandes transferências

13.3 Interface de I/O da Aplicação

- Nesta seção, discutimos técnicas de estruturação e interfaces para o sistema operacional que habilitam os dispositivos de I/O a serem tratados de modo-padrão e uniforme.
- Explicamos, por exemplo, como uma aplicação pode abrir um arquivo em um disco sem saber o tipo de disco e como novos discos e outros dispositivos podem ser adicionados a um computador sem comprometer o sistema operacional.

13.3.1 Dispositivos de Blocos e de Caracteres

- A interface de dispositivo de bloco captura todos os aspectos necessários para o acesso a drives de disco e outros dispositivos orientados a blocos. O esperado é que o dispositivo entenda comandos como `read ()` e `write ()`.
- Se for um dispositivo de acesso randômico, também é esperado que ele tenha um comando `seek ()` para especificar que bloco deve ser transferido em seguida.
- Normalmente, as aplicações acessam tal dispositivo por uma interface do sistema de arquivos. Observe que `read ()`, `write ()` e `seek ()` capturam os comportamentos essenciais dos dispositivos de armazenamento de blocos para que as aplicações sejam isoladas das diferenças de baixo nível entre esses dispositivos.

13.3.2 Dispositivos de Redes

- Como as características de desempenho e endereçamento do I/O de rede diferem significativamente das características do I/O de disco, a maioria dos sistemas operacionais fornece uma interface de I/O de rede que é diferente da interface `read()`—`write()`—`seek()` usada para discos.
- Uma interface disponível em muitos sistemas operacionais, incluindo o UNIX e o Windows, é a interface de socket de rede.

13.3.3 Relógios e Timers

- A maioria dos computadores tem **relógios e timers** em hardware que fornecem **três funções básicas**:
- Informam a **hora corrente**;
- Informam o **tempo decorrido**;
- **Configuram** um **timer** para disparar a operação X no tempo T.
- Essas **funções** são muito usadas pelo sistema operacional, assim como por **aplicações** de **tempo** crítico. Infelizmente, as chamadas de sistema que as **implementam** não são **padronizadas** entre os sistemas operacionais.

13.3.4 I/O sem Bloqueio e Assíncrono

- Outro **aspecto** da **interface** de chamadas de sistema está **relacionado** com a escolha entre I/O com e sem **bloqueio**.
- Quando uma **aplicação** emite uma chamada de sistema com **bloqueio**, a **execução** da **aplicação** é **suspensa**.
- A **aplicação** é **transferida** da fila de **execução** do sistema operacional para uma fila de espera.
- Após a chamada de sistema ser concluída, a **aplicação** é **transferida** de volta para a fila de **execução**, onde se torna **elegível** para **retomar a execução**.

13.3.5 I/O Vetorizado

- Alguns sistemas operacionais fornecem outra **variação** importante de I/O, por meio de suas **interfaces de aplicações**.
- O I/O **vetorizado** (também conhecido como **scatter-gather**, ou **espalhar-reunir**) permite que uma única chamada de sistema execute **múltiplas** operações de I/O envolvendo **múltiplas** localidades.

13.4 Subsistema de I/O do Kernel

- Os **kernels** fornecem muitos **serviços** relacionados com o **I/O**.
- **Vários serviços** — **scheduling**, armazenamento em **buffer**, **armazenamento em cache**, **spooling**, **reserva** de dispositivos e **manipulação de erros** — são fornecidos pelo **subsistema** de I/O do **kernel** e embutidos na **infraestrutura** de **hardware** e de **drivers** de dispositivos.
- O **subsistema** de I/O também é responsável pela sua própria **proteção** contra processos **incorretos** e usuários **maliciosos**.

13.4.1 Scheduling de I/O

- O **escalonamento de processos** ou **agendados de tarefas** (em inglês **scheduling**) é uma atividade organizacional feita pelo **escalonador (scheduler)** da **CPU** ou de um sistema distribuído, possibilitando **executar os processos mais viáveis e concorrentes**, priorizando determinados tipos de processos, como os de **I/O Bound** e os **CPU Bound**.
- **I/O bound** é um termo utilizado para designar os **sistemas** que fazem uso **intensivo de entrada/saída**. (como por exemplo algum usuário querendo copiar um arquivo para o **Pen Drive**).
- **CPU Bound** é quando o tempo de **processamento** depende mais do **processador** do que das **entradas e saídas**, fazendo assim com que atrapalhe o tempo total de **processamento**.

13.4.2 Armazenamento em Buffer

- Buffer é uma região física da memória de um computador utilizada para armazenar dados temporariamente, tendo como finalidade manter as informações salvas antes de serem efetivamente usadas.
- Como sabemos, um buffer é uma área da memória que armazena os dados que estão sendo transferidos entre dois dispositivos ou entre um dispositivo e uma aplicação.
- Uma das utilidades do armazenamento em buffer é fornecer adaptações para dispositivos que tenham diferentes tamanhos de transferências de dados (Tais disparidades são particularmente comuns em redes de computadores, em que os buffers são amplamente usados na fragmentação e remontagem de mensagens.)

13.4.3 Armazenamento em Cache

- Na área da computação, um cache é uma camada de armazenamento físico de dados de alta velocidade que guarda um subconjunto de dados, geralmente temporário por natureza, para que futuras solicitações referentes a esses dados sejam atendidas de modo mais rápido do que é possível fazer comparado ao acesso ao local de armazenamento principal dos dados.
- O armazenamento em cache permite reutilizar com eficiência dados recuperados ou computados anteriormente.
- Basicamente o acesso à cópia armazenada em cache é muito mais eficiente do que o acesso original.

13.4.4 Spooling

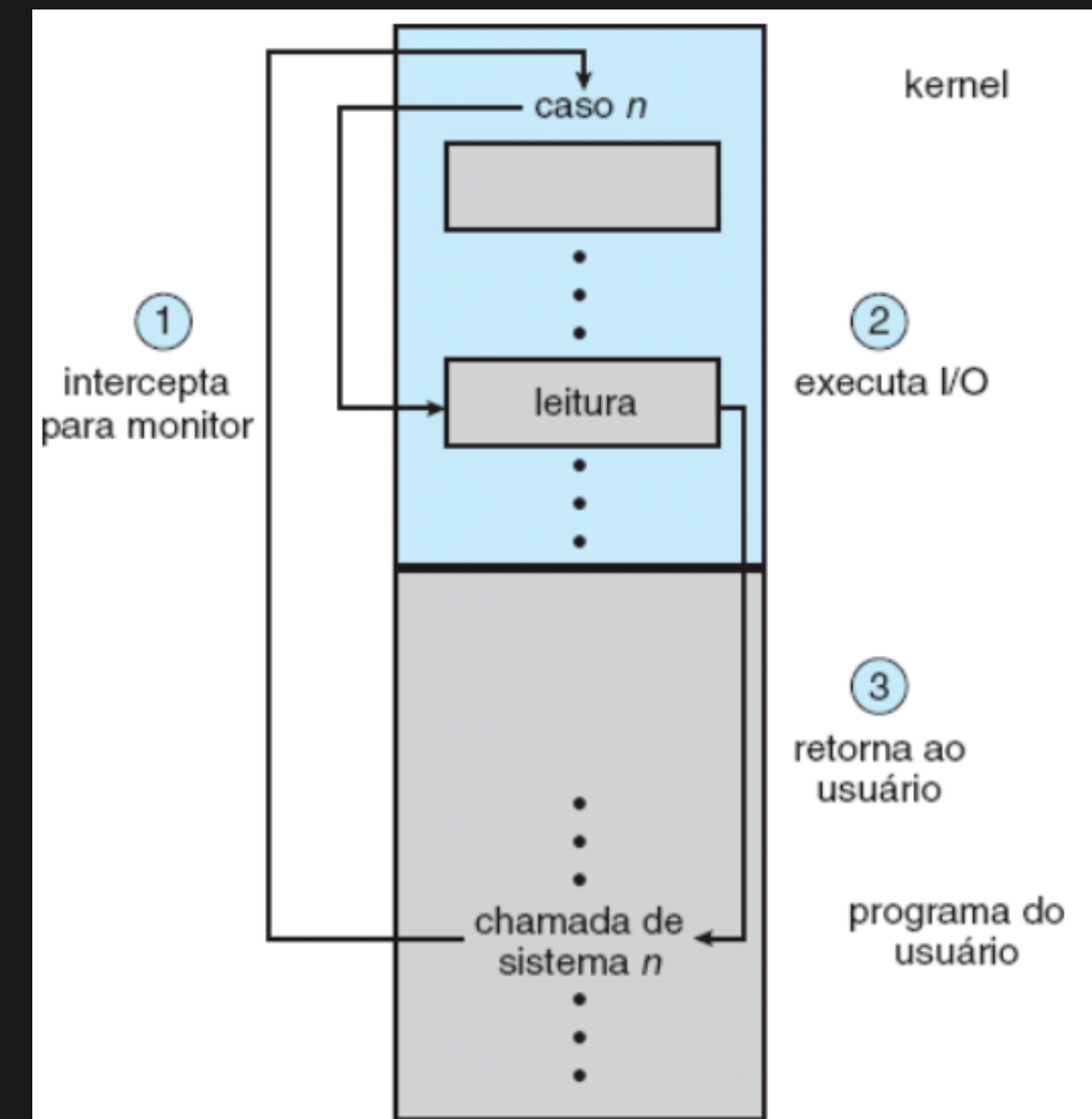
- Um spool é um buffer que mantém a saída para um dispositivo, tal como uma impressora que não pode aceitar fluxos de dados intercalados.
- Embora uma impressora só possa servir um job de cada vez, várias aplicações podem querer imprimir suas saídas concorrentemente, sem deixar que sejam misturadas.
- O sistema operacional resolve esse problema interceptando todas as saídas enviadas para a impressora.
- A saída de cada aplicação é armazenada no spool de um arquivo de disco separado.
- Quando uma aplicação termina a impressão, o sistema de spooling enfileira o arquivo de spool correspondente para ser emitido pela impressora.
- Enfim, o sistema de spooling copia um a um os arquivos de spool enfileirados, para a impressora. Permitindo sistema exibir a fila, remover jobs antes que imprimam, a suspender impressões enquanto a impressora está sendo usada, e assim por diante.

13.4.5 Manipulação de Erros

- Um sistema operacional que usa memória protegida pode se resguardar de muitos tipos de erros de hardware e de aplicações, de modo que uma falha total no sistema não seja o resultado usual de cada defeito mecânico menor. Dispositivos e transferências de I/O podem falhar de muitas maneiras, por questões temporárias; por exemplo, quando uma rede fica sobrecarregada, ou por questões “permanentes”, e quando um controlador de disco apresenta defeito.
- Os sistemas operacionais conseguem, com frequência, resolver efetivamente falhas temporárias.
- Por exemplo, uma falha de um read () em disco resulta em nova tentativa do read (), e um erro de um send () na rede resulta em um resend () se o protocolo assim o especificar.
- Infelizmente, se um componente importante experimentar uma falha permanente, o sistema operacional não conseguirá fazer a recuperação.

13.4.6 Proteção de I/O

- Os **erros** estão **intimamente** relacionados com a **questão** da proteção.
- Um **processo** de usuário pode tentar, **acidental** ou **intencionalmente**, corromper a operação normal de um sistema procurando emitir **instruções** de I/O **ilegais**.
- Podemos usar **vários mecanismos** para **assegurar** que tais **corrupções** não possam ocorrer no sistema.



13.4.7 Estruturas de Dados de Kernel

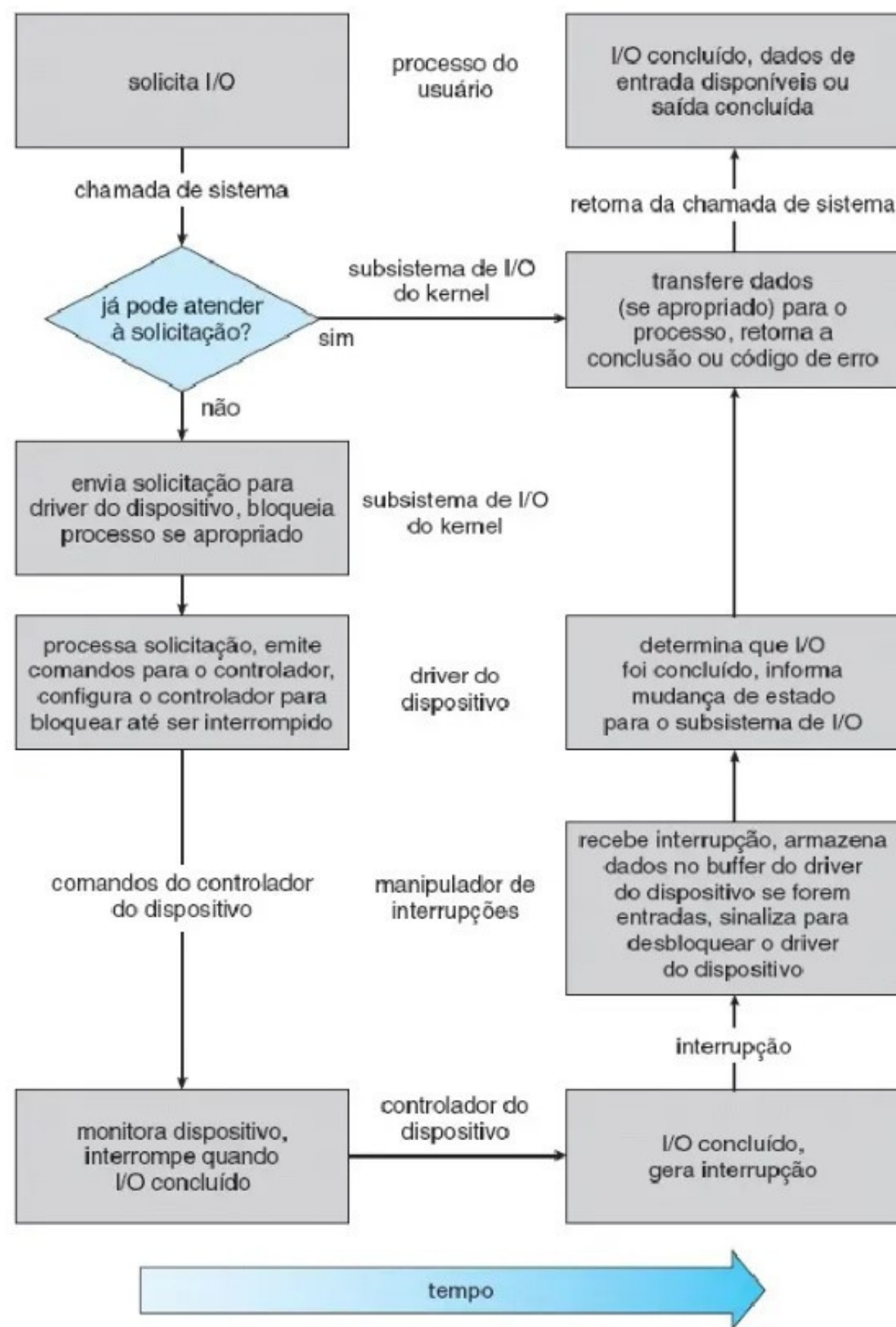
- O **kernel** precisa manter **informações** de estado sobre o uso dos **componentes** de I/O.
- Ele faz isso por meio de uma **variedade** de suas **estruturas** de dados internas.
- O **kernel** usa muitas **estruturas** semelhantes para rastrear **conexões** de rede, **comunicações** de dispositivos de **caracteres** e outras atividades de I/O.

13.4.8 Resumo do Subsistema de I/O do Kernel

- Gerenciamento do espaço de nomes para arquivos e dispositivos.
- Controle de acesso a arquivos e dispositivos.
- Controle de operações [por exemplo, um modem não pode executar um seek ()].
- Alocação de espaço para o sistema de arquivos.
- Alocação de dispositivos.
- Armazenamento em buffer, armazenamento em cachê, e spooling.
- Scheduling de I/O.
- Monitoramento do status dos dispositivos, manipulação de erros e recuperação de falhas.
- Configuração e inicialização de drivers de dispositivos.

The background features a dark gray field with a pattern of white binary digits (0s and 1s) and stylized circuit traces. The traces are composed of horizontal and vertical lines with small circular nodes at the intersections, resembling a printed circuit board layout. The binary code is scattered throughout, with some sequences appearing more prominent than others.

13.5 Transformando Solicitações de I/O em Operações de hardware



- Um processo emite uma chamada de sistema read () com bloqueio para o descritor de um arquivo que foi aberto previamente.
- O código da chamada de sistema no kernel verifica a precisão dos parâmetros. No caso de entrada, se os dados já estão disponíveis no cache de buffer, eles são retornados ao processo, e a solicitação de I/O é concluída.
- Caso contrário, um I/O físico deve ser executado. O processo é removido da fila de execução e inserido na fila de espera para o dispositivo, e a solicitação de I/O é incluída no schedule. Eventualmente, o subsistema de I/O envia a solicitação ao driver do dispositivo. Dependendo do sistema operacional, a solicitação é enviada por uma chamada de sub-rotina ou de uma mensagem interna do kernel.
- O driver do dispositivo aloca espaço no buffer do kernel para receber os dados e inclui o I/O no schedule. Eventualmente, o driver envia comandos ao controlador do dispositivo gravando nos registradores de controle do dispositivo.
- O controlador do dispositivo opera o hardware do dispositivo para executar a transferência de dados.
- O driver pode sondar o status e os dados, ou pode ter estabelecido uma transferência DMA para a memória do kernel. Vamos supor que a transferência seja gerenciada por um controlador de DMA que gera uma interrupção quando a transferência é concluída.
- O manipulador de interrupções correto recebe a interrupção por meio da tabela de vetores de interrupções, armazena quaisquer dados necessários, sinaliza para o driver do dispositivo e retorna a interrupção.
- O driver do dispositivo recebe o sinal, determina que a solicitação de I/O foi concluída, determina o status da solicitação e sinaliza para o subsistema de I/O do kernel que a solicitação foi concluída.
- O kernel transfere os dados ou retorna códigos para o espaço de endereçamento do processo solicitante e transfere o processo da fila de espera de volta para a fila de prontos.
- A transferência do processo para a fila de prontos o desbloqueia. Quando o scheduler atribui o processo à CPU, ele retoma a sua execução quando se completa a chamada de sistema.

13.6 Streams

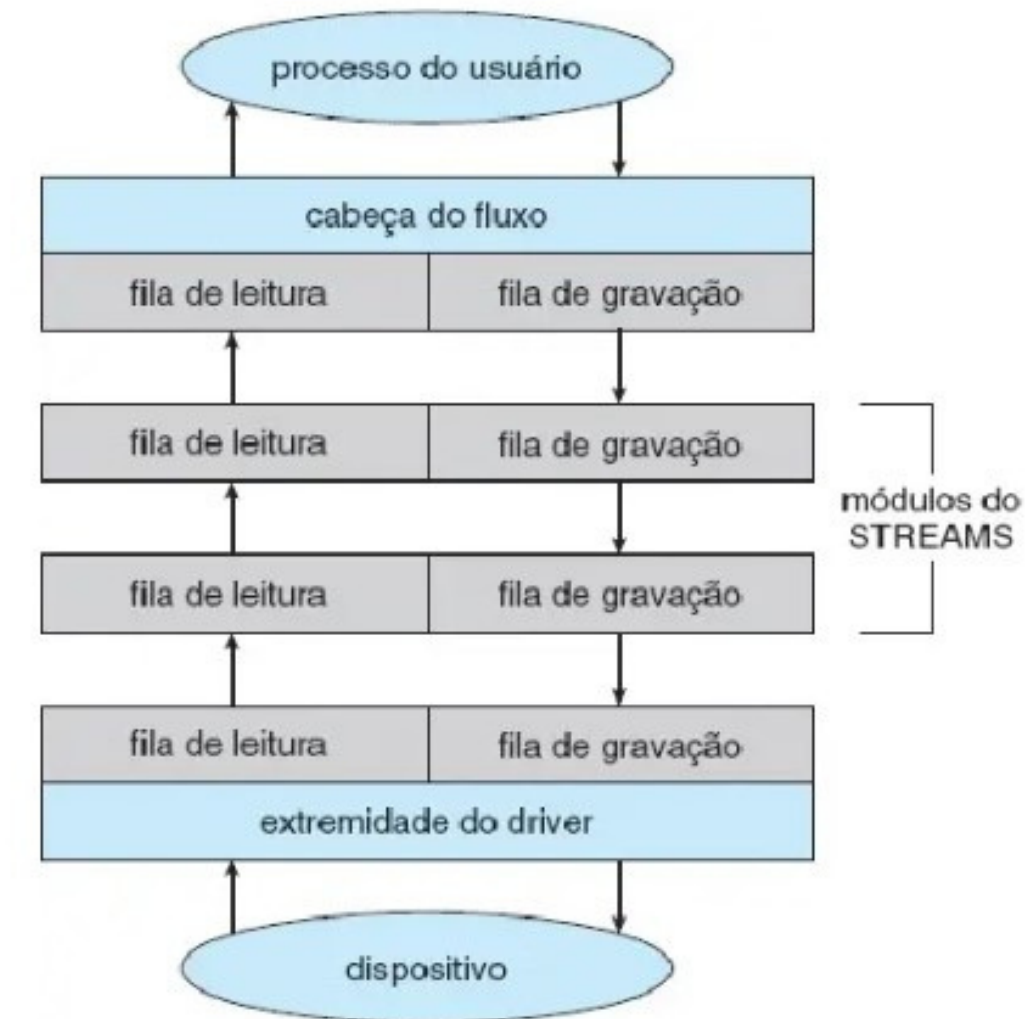



Figura 13.14 A estrutura do STREAMS.

- Os módulos fornecem a funcionalidade de processamento do STREAMS; eles são impulsionados para um fluxo por meio da chamada de sistema `ioctl()`.
- Por exemplo, um processo pode abrir um dispositivo de porta serial por um fluxo e impulsionar um módulo para manipular a edição de entradas. Já que mensagens são trocadas entre filas em módulos adjacentes, a fila em um módulo pode estourar uma fila adjacente. Para impedir que isso ocorra, uma fila pode suportar controle de fluxo.
- Sem controle de fluxo, uma fila aceita todas as mensagens e as envia imediatamente à fila no módulo adjacente sem armazená-las em buffer. Uma fila que suporta controle de fluxo armazena as mensagens em buffer e não as aceita sem o espaço suficiente no buffer. Esse processo envolve trocas de mensagens de controle entre filas em módulos adjacentes.

- 
- Um processo de usuário grava dados em um dispositivo usando as chamadas de sistema `write ()` ou `putmsg ()`.
 - A chamada de sistema `write ()` grava dados brutos no fluxo, enquanto `putmsg ()` permite que o processo de usuário especifique uma mensagem.
 - Independentemente da chamada de sistema usada pelo processo de usuário, a cabeça do fluxo copia os dados em uma mensagem e os distribui para a fila do próximo módulo da sequência.
 - Essa cópia de mensagens continua até que a mensagem seja copiada na extremidade do driver e, então, no dispositivo. Da mesma forma, o processo de usuário lê dados a partir da cabeça do fluxo usando as chamadas de sistema `read ()` ou `getmsg ()`.
 - Se `read ()` for usada, a cabeça do fluxo obtém a mensagem na fila adjacente e retorna dados comuns (um fluxo de bytes desestruturado) para o processo.
 - Se `getmsg ()` for usada, uma mensagem é retornada para o processo.
 -

REFERÊNCIA :

https://www.academia.edu/44329279/Fundamentos_de_Sistemas_operacionais_9a_Edic_a_o_LTC