Aula 07
Desempenho de E/S e Linux

DISCIPLINA: SISTEMAS OPERACIONAIS II

PROFESSOR: JHEYMESSON APOLINÁRIO CAVALCANTI

1. Desempenho

As operações de E/S constituem um fator importante no desempenho dos sistemas, pois…

Impõem demandas pesadas sobre a CPU:

- Execução dos códigos de drivers de dispositivos;
- Organização de processos no scheduling;
- Mudanças de contexto devidas às interrupções;
- Etc.

Expõem as ineficiências dos mecanismos de manipulação de interrupções no kernel;

Sobrecarregam o barramento da memória nas cópias de dados entre controladores e memória física.

1. Desempenho

O tráfego de rede também pode causar uma alta taxa de mudanças de contexto;

Seja o exemplo do login remoto de um computador para outro;

- Cada caractere digitado na máquina local deve ser transportado para a máquina remota.

1. Desempenho

Em relação às interrupções...

Os computadores modernos podem manipular milhares de interrupções por segundo;

No entanto, a manipulação de interrupções é uma tarefa relativamente dispendiosa;

As interrupções causam:

- Mudança de estado/contexto;
- Execução do manipulador de interrupções;
- Restauração estado/contexto;

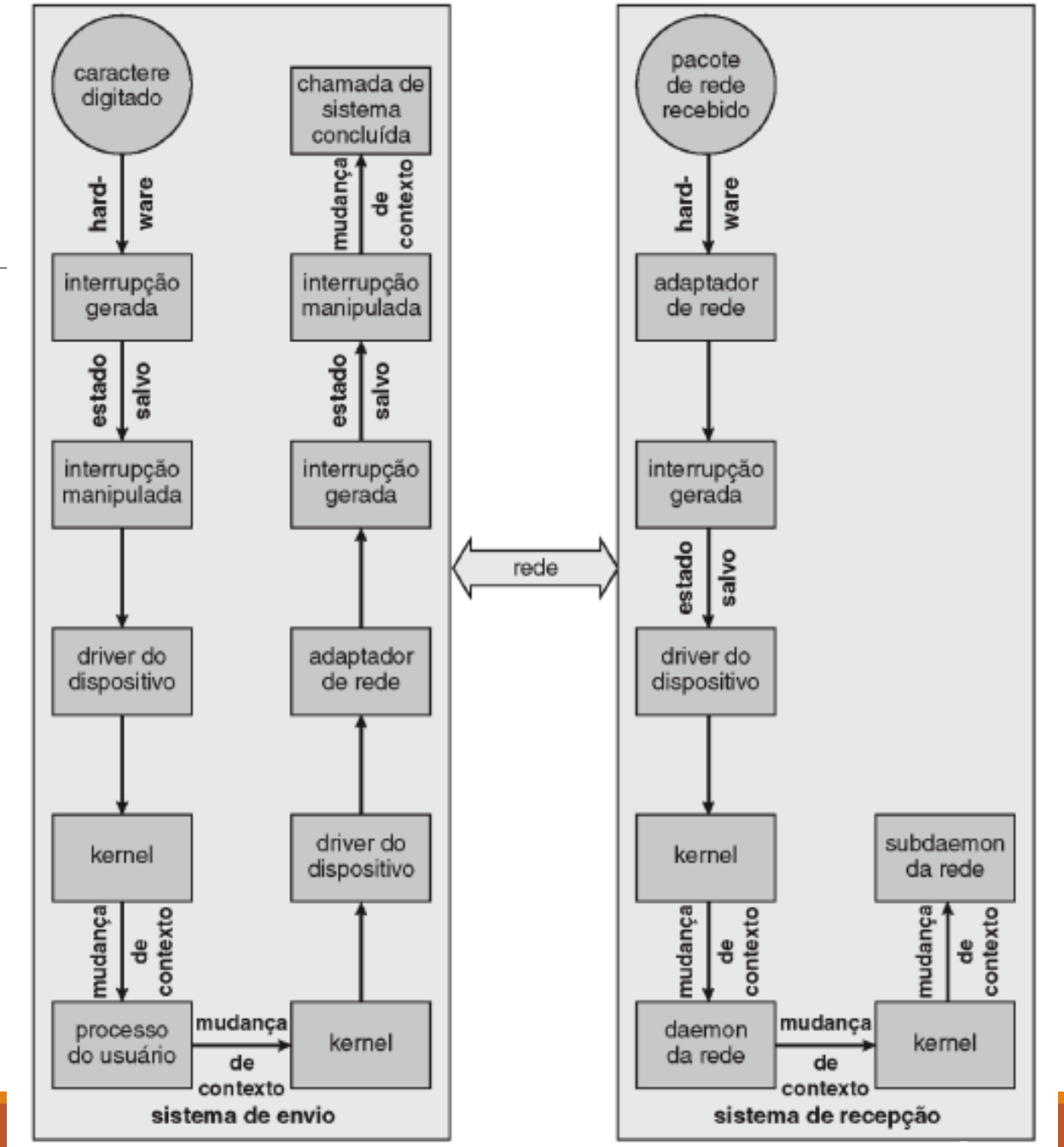
Dependendo da quantidade de dispositivos de E/S disponíveis, a E/S programada (polling) pode ser mais eficiente que a E/S dirigida por interrupções.

1. Desempenho

Como pode ser visto, há muitas mudanças de contexto;

Note que, usualmente, o receptor ainda ecoa o caractere de volta para o emissor;

Essa abordagem duplica o trabalho.



1. Desempenho

Uma alternativa para reduzir as mudanças de contexto:

Uso de processadores Front-End separados para a E/S (Canais de E/S):

- Visa a redução da carga de interrupções na CPU principal;
- Encontrada em mainframes e outros sistemas de ponta;
- É uma lógica semelhante à do DMA.

1. Desempenho

Como melhorar o desempenho?

- Reduzir o número de mudanças de contexto;
- Reduzir o número de vezes que os dados são copiados na memória durante a transferência entre dispositivo e aplicação;
- Reduzir a frequência de interrupções pelo uso de transferências grandes, controladores inteligentes e sondagens (polling);
- Utilizar o DMA (ou processadores Front-End);
- Balancear o uso da CPU, do subsistema de memória, do barramento e da E/S.

1. Desempenho

Qual o melhor local para se implementar uma funcionalidade de E/S?

- No hardware do dispositivo?
- No driver do dispositivo?
- No software da aplicação?

1. Desempenho

O mais comum, no início, é implementarmos no nível da aplicação:

- Código mais flexível;
- Bugs normalmente não derrubam o sistema;

No entanto...

- É ineficiente;
- Pode causar sobrecarga nas mudanças de contexto;
- Não se beneficia das estruturas internas e das funcionalidades do kernel.

1. Desempenho

Se funcionar no nível da aplicação, pode-se implementar a funcionalidade no kernel:

- Melhor desempenho;

No entanto...

- Maior esforço devido à complexidade e tamanho;
- Pode corromper dados e derrubar o sistema.

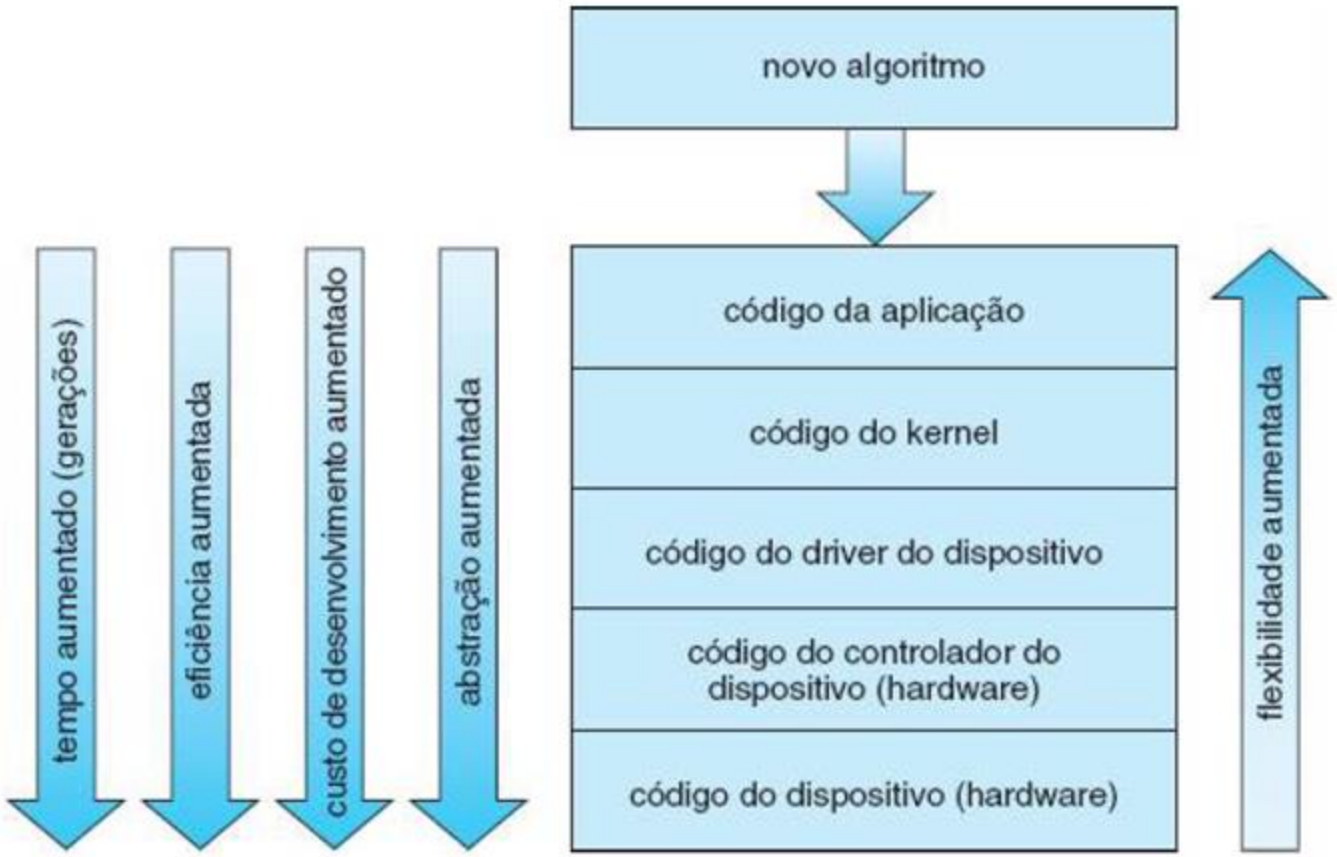
O melhor desempenho pode vir de implementação especializada em hardware:

- Melhor desempenho ainda;

No entanto...

- Maior dificuldade e custos;
- Menor flexibilidade;
- Maior tempo de desenvolvimento.

1. Desempenho



1. Introdução

Assim como todos os computadores, aqueles executando com Linux têm dispositivos de E/S, como:

- Discos;
- Impressoras;
- Redes conectadas a eles;
- Etc.

Para que programas acessem esses dispositivos, a solução do Linux foi integrá-los (os dispositivos) em sistemas de arquivos por meio dos chamados **arquivos especiais**.

1. Introdução

Em outras palavras, no Linux, todos os dispositivos de E/S são feitos para parecerem arquivos;

Por conta disso, é possível acessar os dispositivos por chamadas de sistema utilizadas no acesso a arquivos comuns;

Ex:

- *cp file /dev/lp* copia *file* para a impressora, fazendo que ele seja impresso (presumindo que o usuário tenha permissão para acessar */dev/lp*).

Programas podem abrir, ler e escrever arquivos especiais exatamente da mesma maneira que eles fazem com arquivos regulares;

Dessa maneira, nenhum mecanismo especial é necessário para fazer E/S.

1. Introdução

Cada dispositivo de E/S é associado a um nome de caminho, normalmente em */dev*;

Exemplo:

- Um disco pode ser */dev/hd1*;
- Uma impressora pode ser */dev/lp*;
- A rede pode ser */dev/net*;
- Etc.

2. Arquivos especiais

Arquivos especiais são divididos em duas categorias: bloco e caractere;

Arquivo especial de bloco:

- Consiste em uma sequência de blocos numerados;
- Cada arquivo de bloco pode ser individualmente endereçado e acessado;
- Em outras palavras, um programa pode abrir um arquivo especial de bloco e ler, digamos, o bloco 124 sem primeiro ter de ler os blocos 0 a 123;
- Arquivos de blocos especiais são tipicamente usados para discos;
- Ex:
  - dev/sda, dev/sdb, etc.

2. Arquivos especiais

Arquivo especial de caractere:

- São normalmente usados para dispositivos que realizam a entrada ou saída de um fluxo de caracteres;
- Teclados, impressoras, redes, mouses, etc.;
- Não é possível (ou mesmo significativo) buscar o bloco 124 em um mouse, por exemplo.
- Ex:
  - Dev/mouse0.

2. Arquivos especiais

Existe drivers de dispositivos que lidam com os dispositivos correspondentes a cada arquivo especial;

Cada driver tem o que é chamado de um número de **dispositivo principal,** que serve para identificá-lo;

Se um driver suporta múltiplos dispositivos (ex: dois discos do mesmo tipo), cada dispositivo tem um número de **dispositivo secundário** que o identifica.

2. Arquivos especiais

Em alguns casos, um único driver pode lidar com dois dispositivos relacionados;

Exemplo:

- O driver correspondendo aos controles */dev/tty* controlam tanto o teclado quanto a tela, muitas vezes pensados como um único dispositivo, o terminal.
- Instrução:
  - echo 1 > /dev/tty

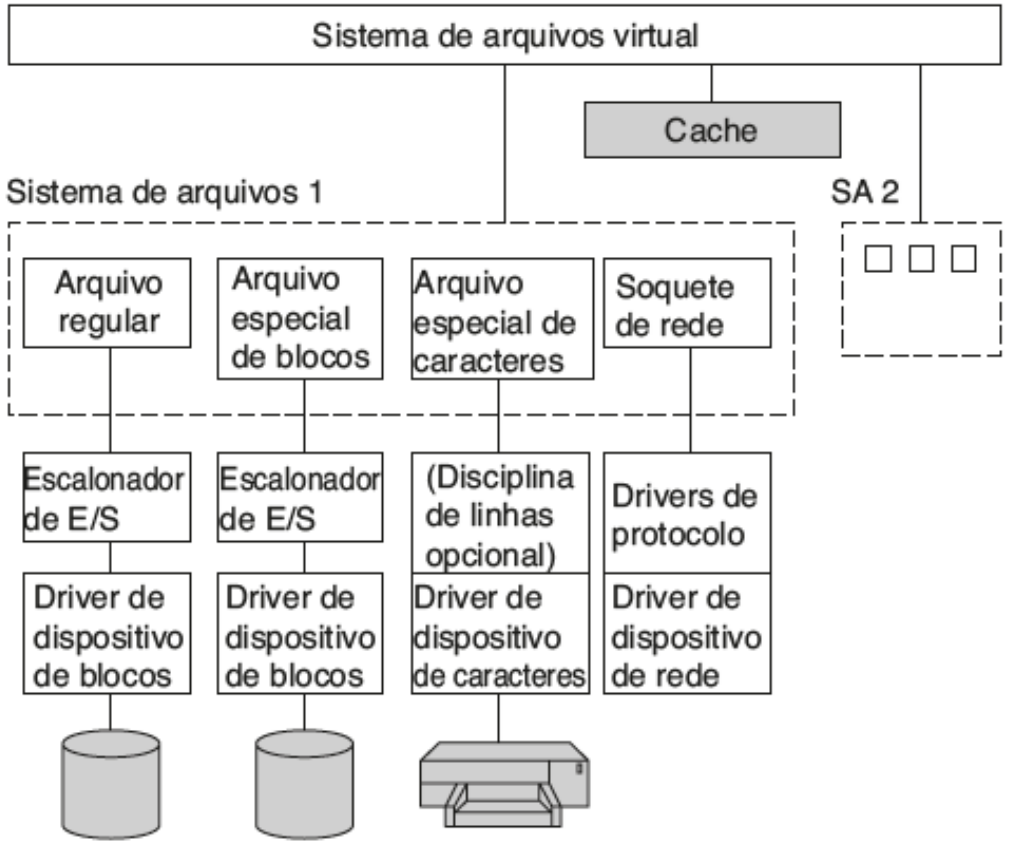
2. Arquivos especiais

Outras funções:

Mostrar o scheduler de I/O em uso corrente: *cat /sys/block/sda/queue/scheduler*

3. Implementação de E/S no Linux

O sistema de E/S do Linux mostrando em detalhes um sistema de arquivos.



3. Implementação de E/S no Linux

Cada driver é dividido em duas partes, que fazem parte do núcleo do Linux e executam em modo núcleo;

A metade de cima faz a interface com o resto do Linux e executa no contexto do chamador;

A metade de baixo interage com o dispositivo e executa no contexto de núcleo;

Drivers têm permissão para fazer chamadas para procedimentos de núcleo para:

- Alocação de memória;
- Gerenciamento de temporizador;
- Controle de DMA;
- Etc.

O conjunto de funções do núcleo que pode ser chamado pelo driver é definido em um documento denominado **Interface Driver-Núcleo**.

3. Implementação de E/S no Linux

O sistema de E/S é dividido em dois componentes maiores: o tratamento de arquivos especiais de blocos e o tratamento de arquivos especiais de caracteres;

O objetivo da parte do sistema que realiza E/S em arquivos especiais de blocos (por exemplo, discos) é minimizar o número de transferências que precisam ser feitas;

Para alcançar esse objetivo, o Linux tem uma cache entre os drivers do disco e o sistema de arquivos.

3. Implementação de E/S no Linux

A cache é uma tabela no núcleo para armazenamento de milhares dos blocos usados recentemente;

Quando um bloco de um disco é necessário por qualquer que seja a razão, uma verificação é feita para ver se ele está na cache;

Se ele estiver, o bloco é tirado dali e um acesso de disco é evitado, resultando em melhoria no desempenho do sistema;

Se o bloco não está na cache da página, ele é lido do disco e dali copiado para onde ele for necessário.

3. Implementação de E/S no Linux

A fim de reduzir a latência de movimentos repetitivos da cabeça do disco, o Linux conta com um **escalonador de E/S**;

O seu propósito é reordenar ou reunir solicitações de leitura/escrita para dispositivos de bloco.

Há muitas variantes de escalonadores, otimizados para diferentes tipos de cargas de trabalho;

Ex:

- deadline;
- cfq;
- noop.

3. Implementação de E/S no Linux

A interação com dispositivos de caracteres é mais simples;

Como os dispositivos de caracteres produzem ou consomem fluxos de caracteres, ou bytes de dados, o suporte para o acesso aleatório faz pouco sentido;

Essa interação se dá de forma “convencional”.

3. Implementação de E/S no Linux

A interação com dispositivos de rede é diferente;

Embora os dispositivos de rede também produzam/consumam fluxos de caracteres, sua natureza assíncrona os torna menos adequados para a integração fácil sob a mesma interface que os outros dispositivos de caracteres;

O driver de dispositivo de rede produz pacotes consistindo de múltiplos bytes de dados, junto com cabeçalhos de rede;

Esses pacotes são então roteados através de uma série de drivers de protocolos, e em última análise passados para a aplicação do espaço usuário.