Universidade Federal da Bahia

Sistemas Operacionais

MATA58

Prof. Maycon Leone M. Peixoto

mayconleone@dcc.ufba.br

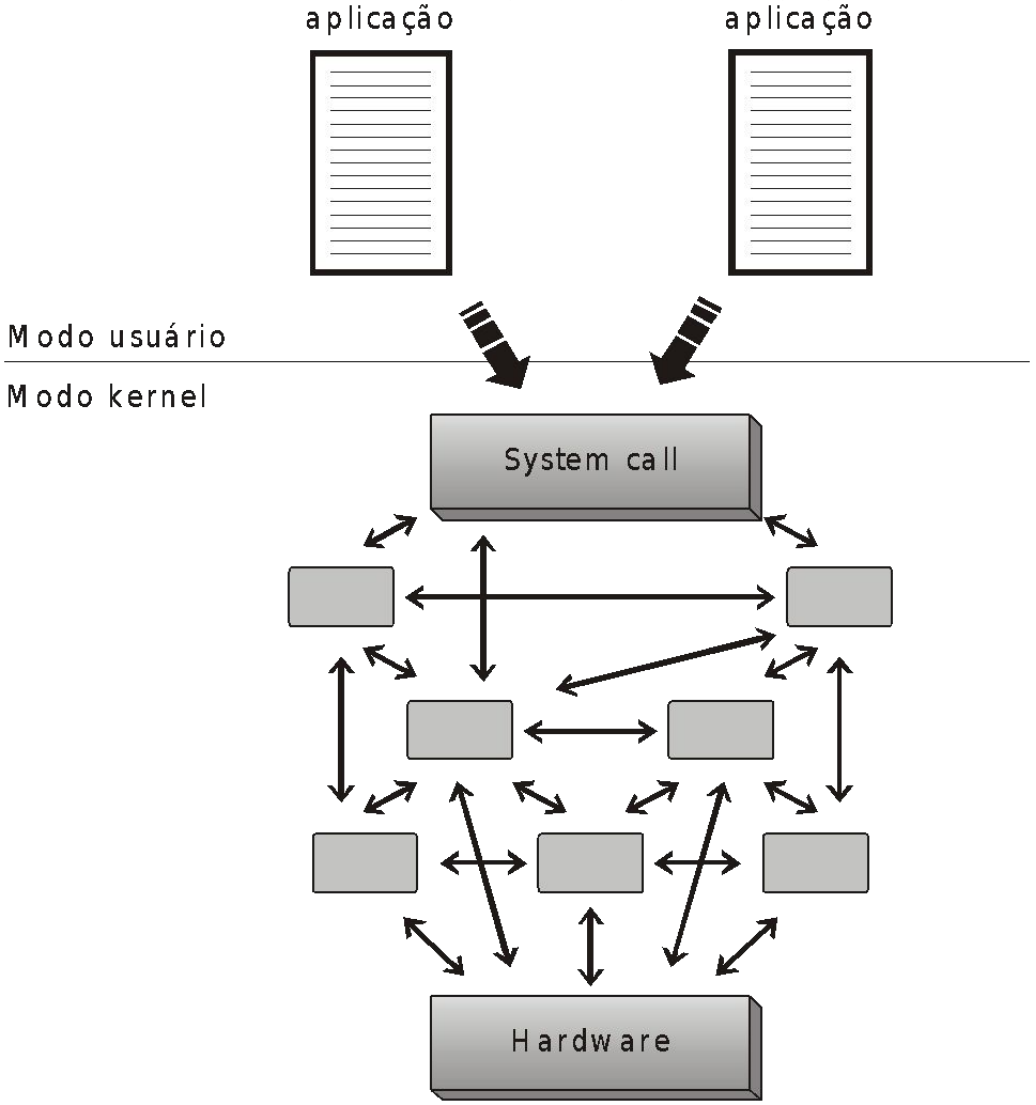
Estrutura dos Sistemas Operacionais

- Principais tipos de estruturas:
  - Monolíticos;
  - Em camadas;
  - Máquinas Virtuais;
  - Arquitetura *Micro-kernel*;
  - Cliente-Servidor;

Estrutura dos Sistemas Operacionais - Monolítico

- Todos os módulos do sistema são compilados individualmente e depois ligados uns aos outros em um único **arquivo-objeto**;
- O Sistema Operacional é um conjunto de processos que podem interagir entre si a qualquer momento sempre que necessário;
- Cada processo possui uma interface bem definida com relação aos parâmetros e resultados para facilitar a comunicação com os outros processos;
- Simples;
- Primeiros sistemas UNIX e MS-DOS;

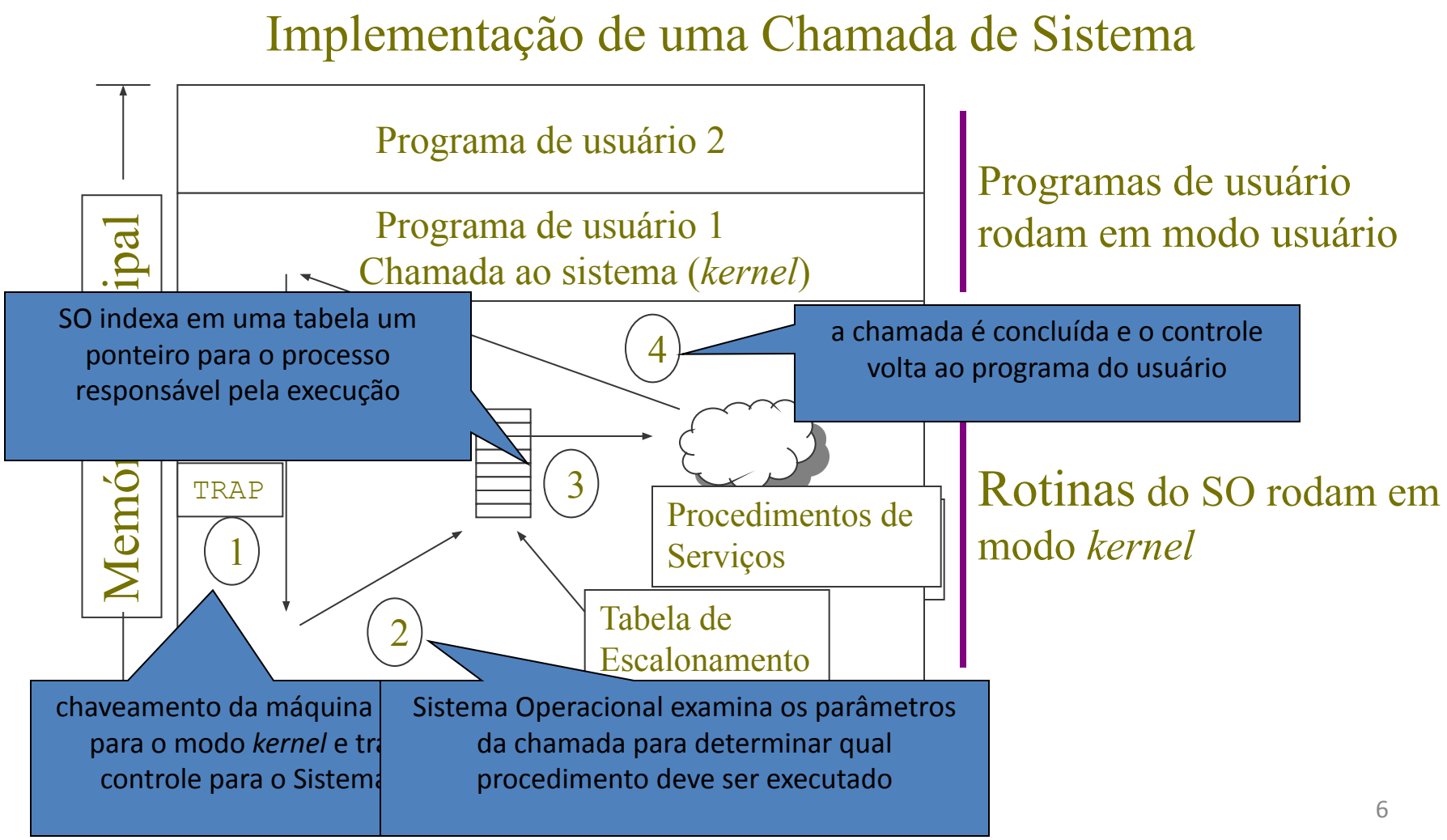
Estrutura dos Sistemas Operacionais - Monolítico



Estrutura dos Sistemas Operacionais - Monolítico

- Os serviços (chamadas) requisitados ao sistema são realizados por meio da colocação de parâmetros em registradores ou pilhas de serviços seguida da execução de uma instrução chamada *TRAP;*

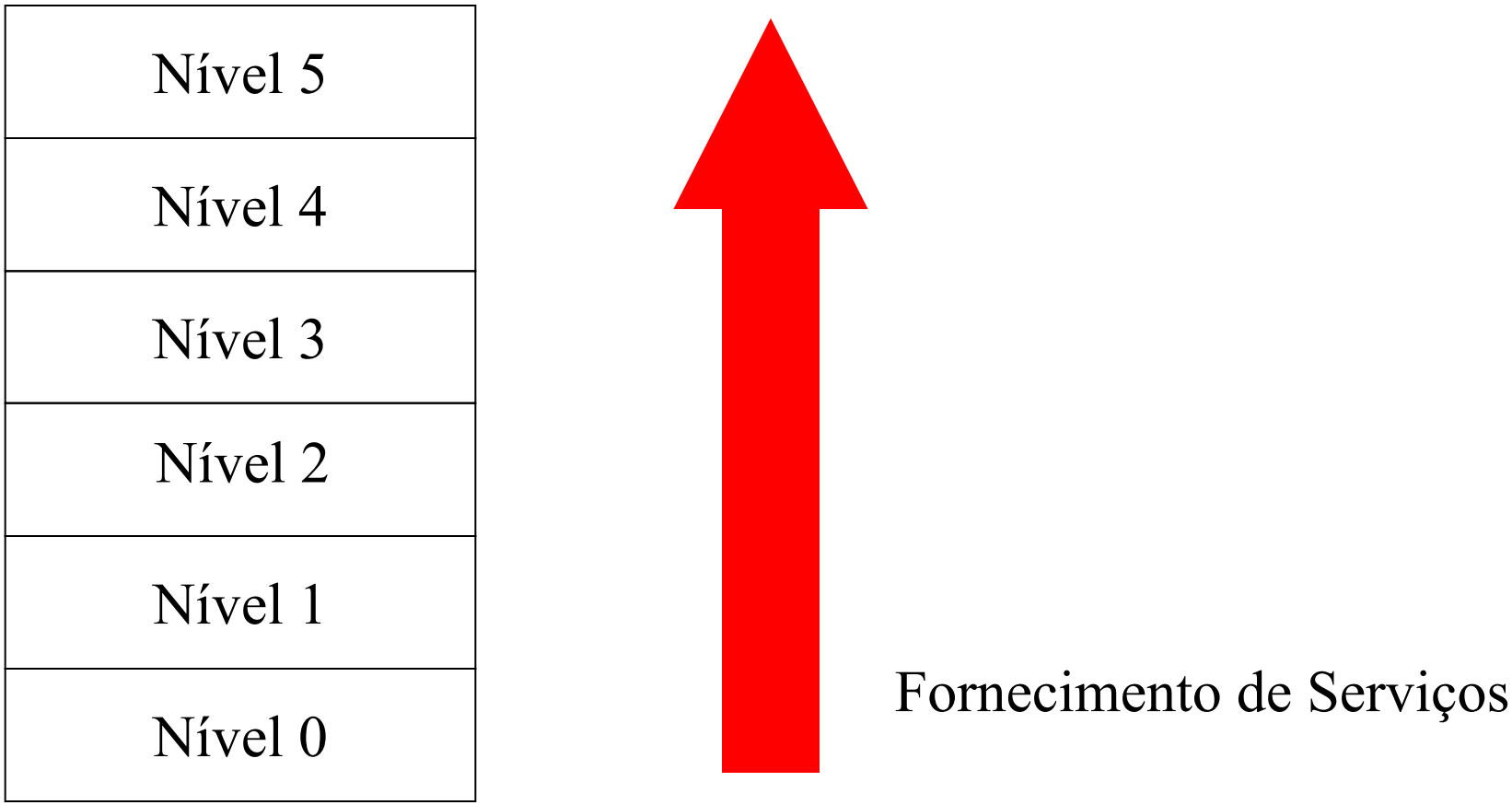
Estrutura dos Sistemas Operacionais - Monolítico



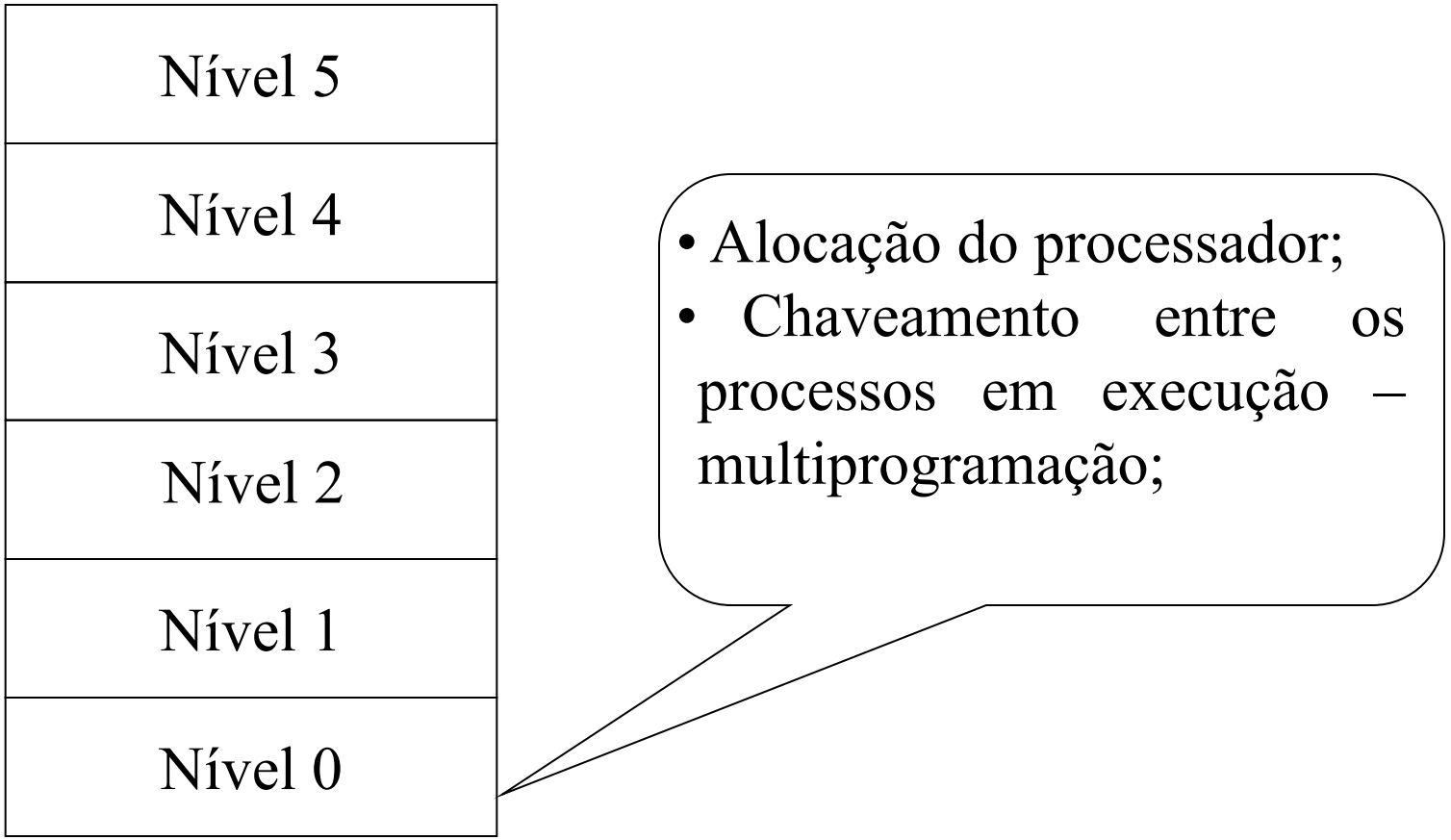
Estrutura dos Sistemas Operacionais – Em camadas

- Possui uma hierarquia de níveis;
- Primeiro sistema em camadas: THE (idealizado por E.W. Dijkstra);
  - Possuía 6 camadas, cada qual com uma função diferente;
  - Sistema em *batch* simples;
- Vantagem: isolar as funções do sistema operacional, facilitando manutenção e depuração
- Desvantagem: cada nova camada implica uma mudança no modo de acesso
- Atualmente: modelo de 2 camadas

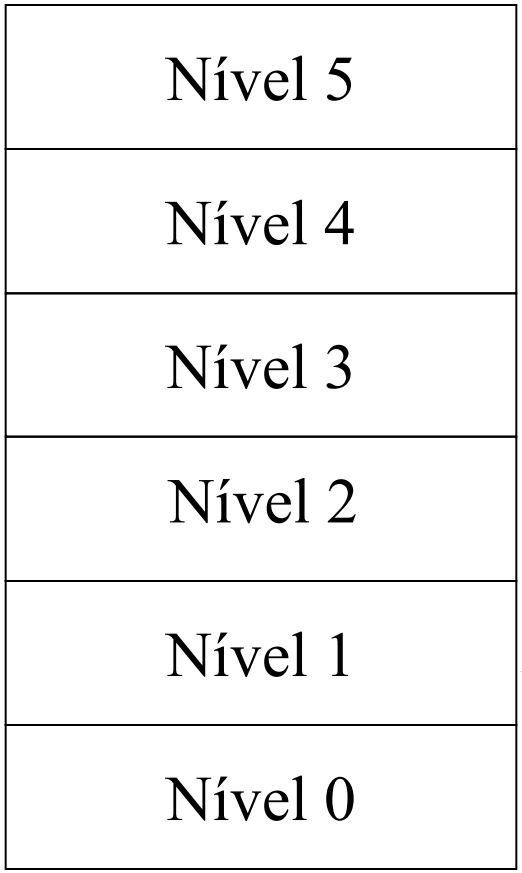
Estrutura dos Sistemas Operacionais – Em camadas



Estrutura dos Sistemas Operacionais – Em camadas

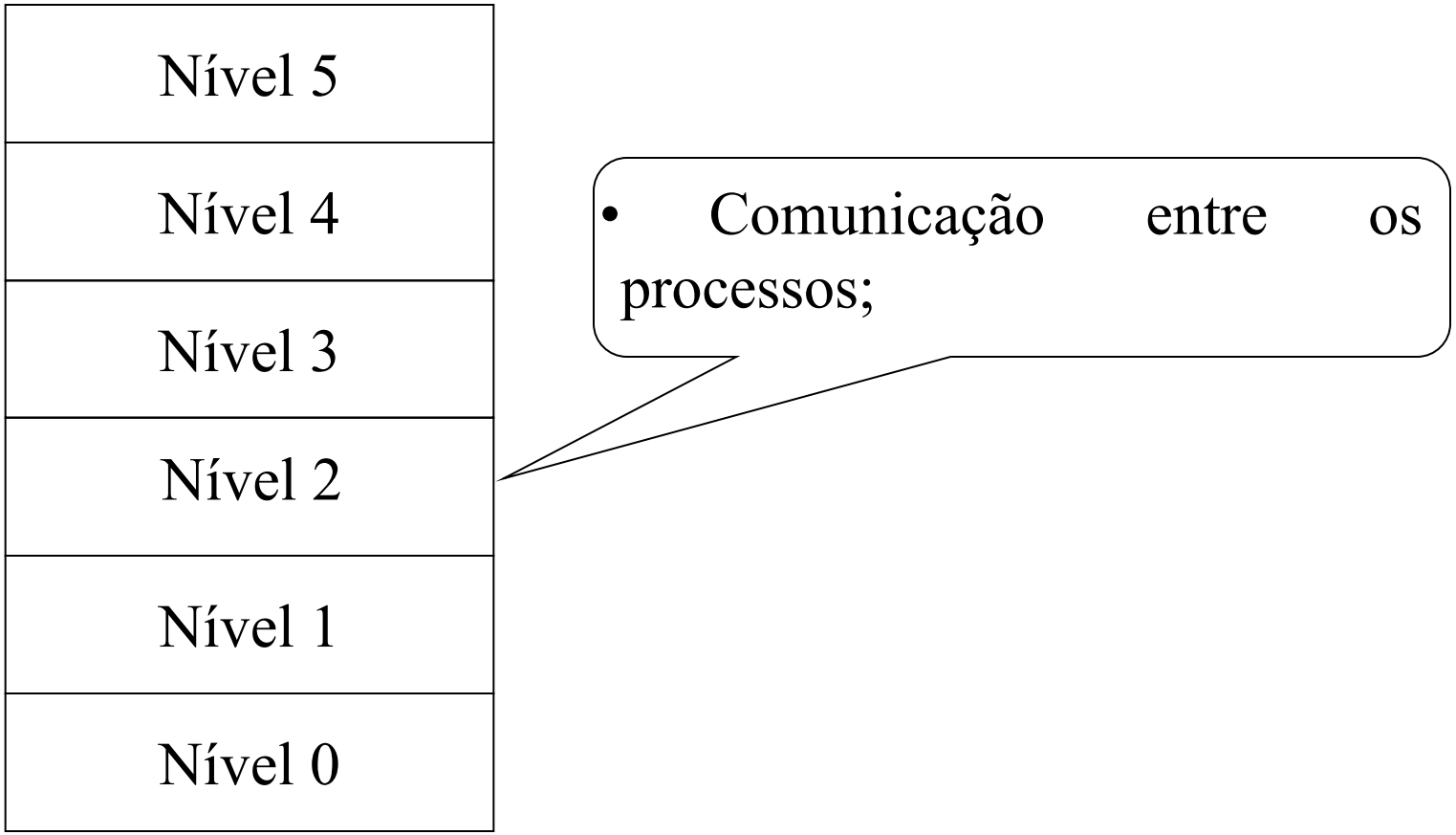


Estrutura dos Sistemas Operacionais – Em camadas

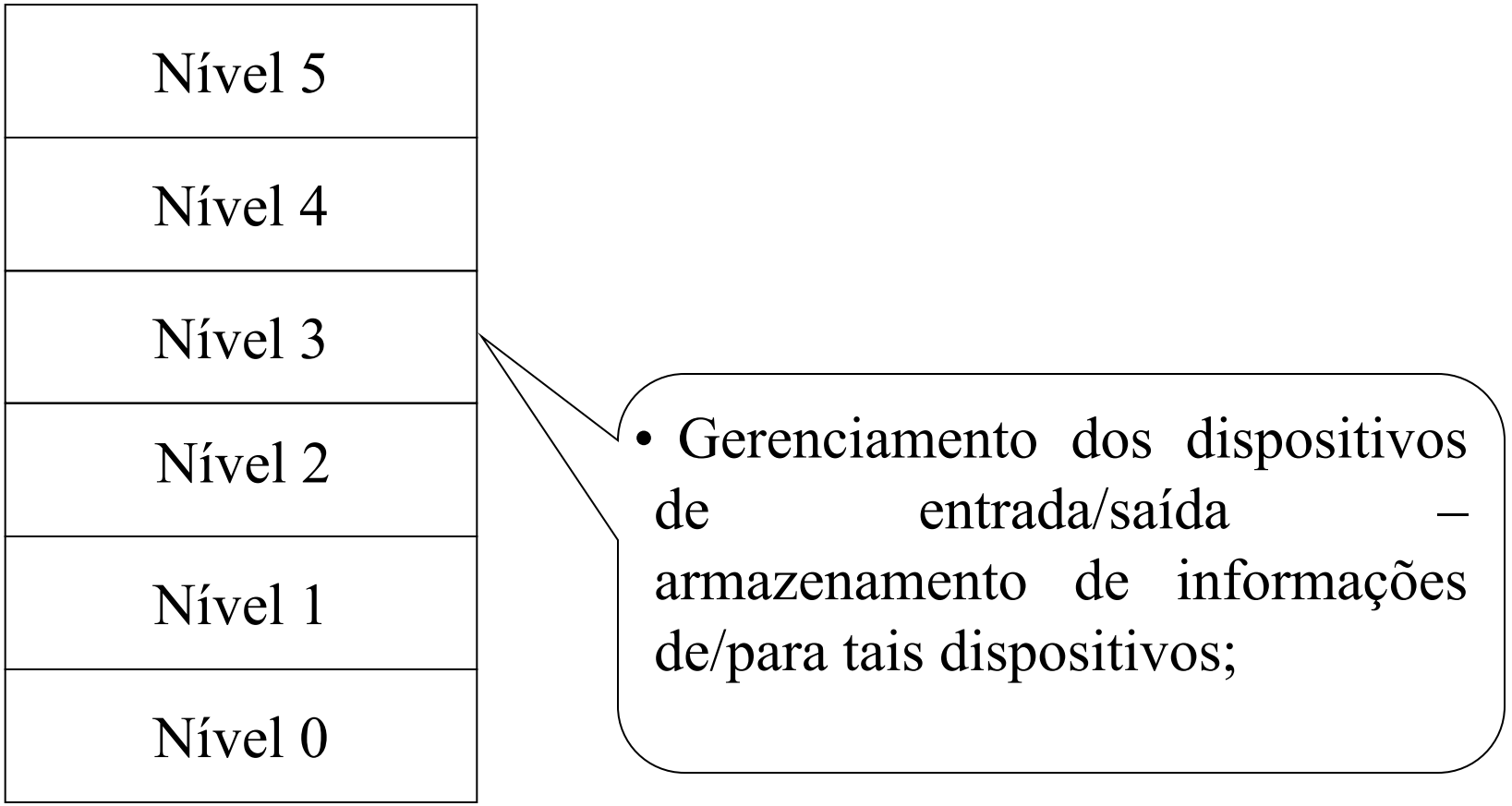


- Gerenciamento da memória;
- Alocação de espaço para processos na memória e no disco:
  - Processo dividido em partes (páginas) para ficarem no disco;

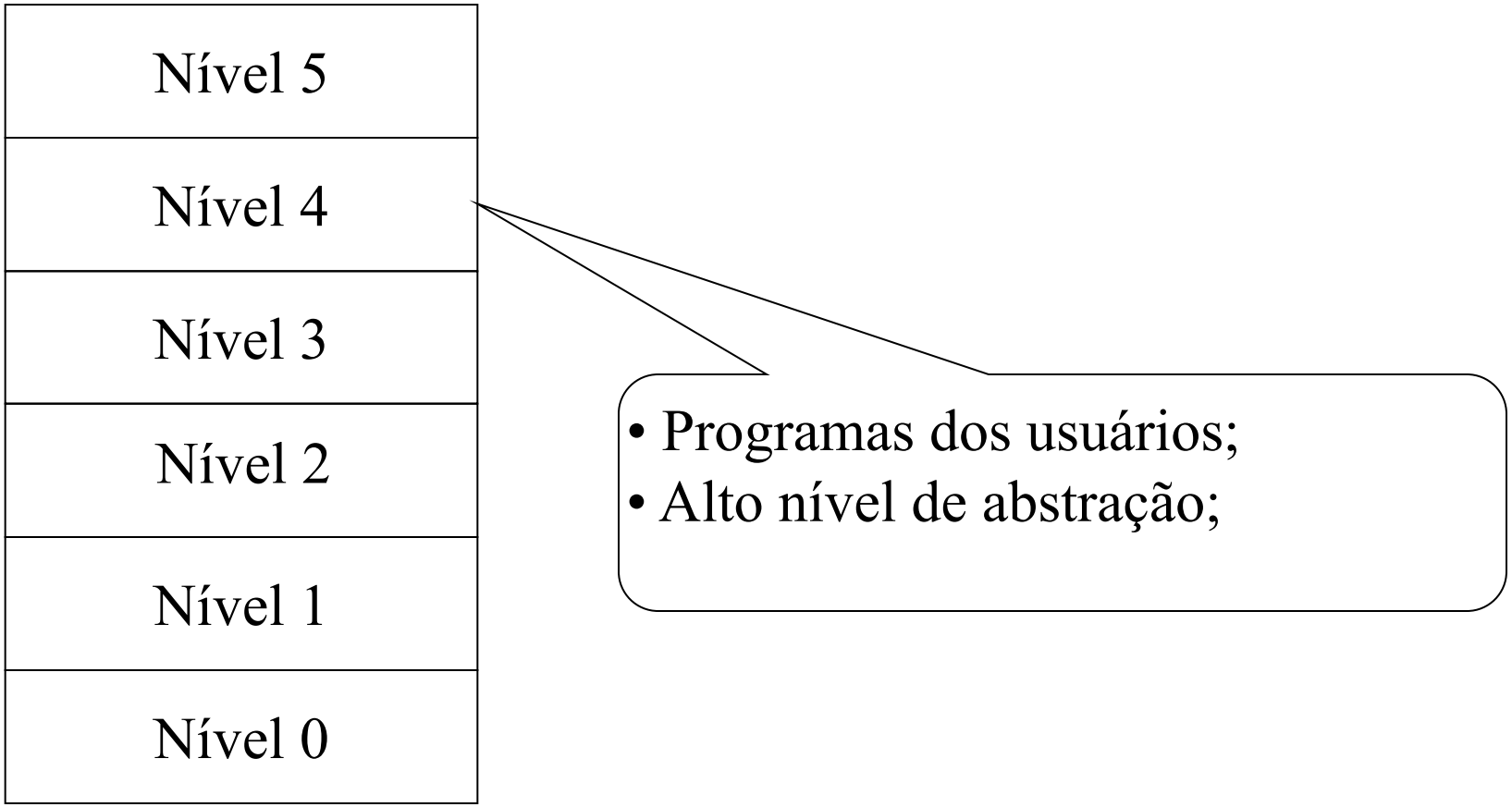
Estrutura dos Sistemas Operacionais – Em camadas



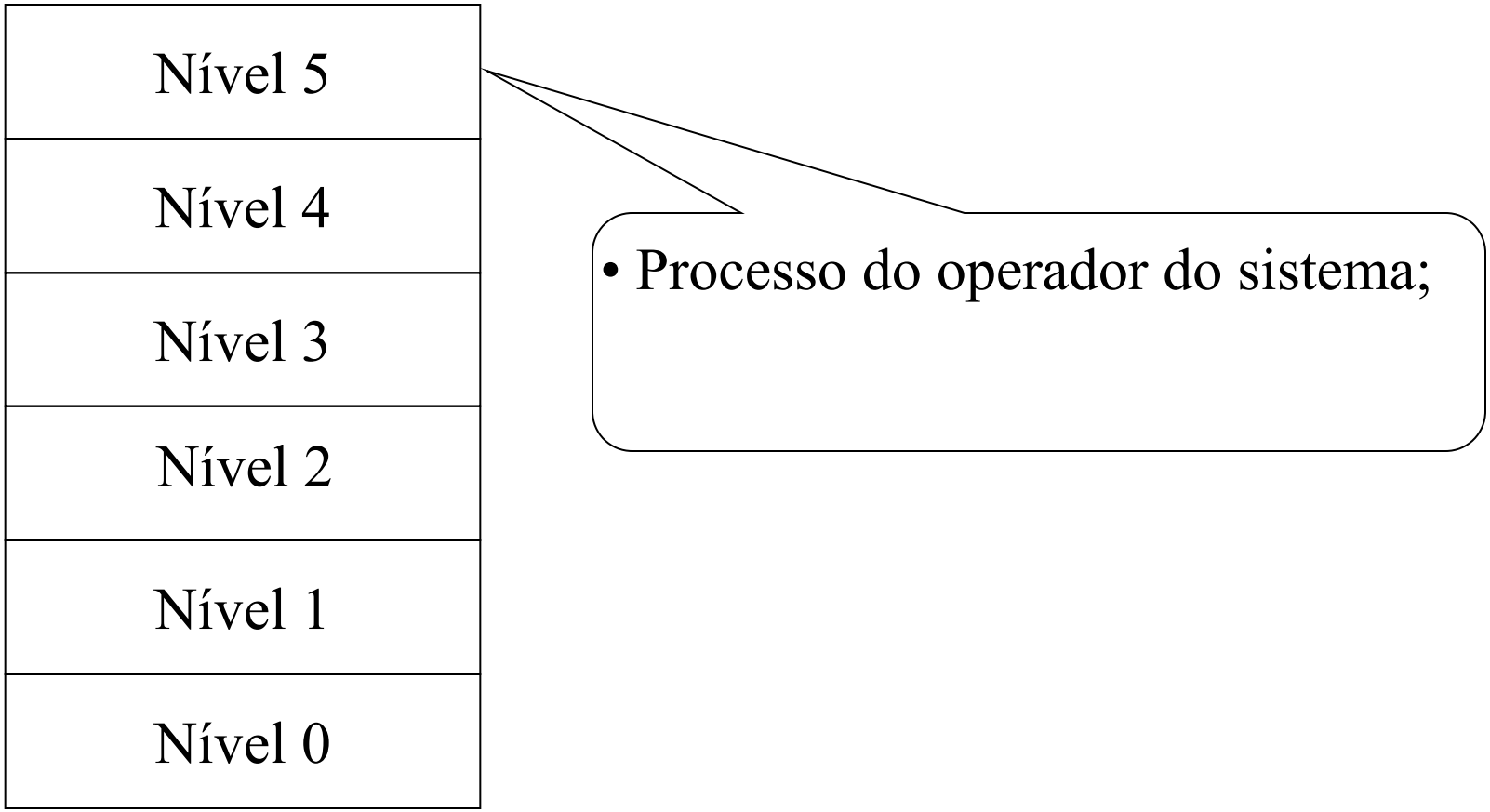
Estrutura dos Sistemas Operacionais – Em camadas



Estrutura dos Sistemas Operacionais – Em camadas



Estrutura dos Sistemas Operacionais – Em camadas



Estrutura dos Sistemas Operacionais – Máquina Virtual

- Idéia em 1960 com a IBM VM/370;
- Modelo de máquina virtual cria um nível intermediário entre o SO e o Hardware;
- Esse nível cria diversas **máquinas virtuais independentes e isoladas**, onde cada máquina oferece um cópia virtual do hardware, incluindo modos de acesso, interrupções, dispositivos de E/S, etc.;
- Cada máquina virtual pode ter seu próprio SO;

Estrutura dos Sistemas Operacionais – Máquina Virtual

- Evolução do OS/360 para o TSS/360:
  - Compartilhamento de tempo (*TimeSharing*);
  - Tanto a multiprogramação quanto a interface com o hardware eram realizadas pelo mesmo processo – sobrecarga gerando alto custo;
- Surge o CP/CMS VM/370 (*Mainframes IBM*)
  - Duas funções distintas em processos distintos:
    - Ambiente para multiprogramação;
    - Máquina estendida com interface para o hardware;

Estrutura dos Sistemas Operacionais – Máquina Virtual

- Principais conceitos:
  - Monitor da Máquina Virtual (VMM): roda sobre o hardware e implementa multiprogramação fornecendo várias máquinas virtuais é o coração do sistema;
  - CMS (*Conversational Monitor System*):
    - *TimeSharing*;
    - Executa chamadas ao Sistema Operacional;
  - Máquinas virtuais são cópias do hardware, incluindo os modos *kernel* e usuário;
  - Cada máquina pode rodar um Sistema Operacional diferente;

Estrutura dos Sistemas Operacionais – Máquina Virtual

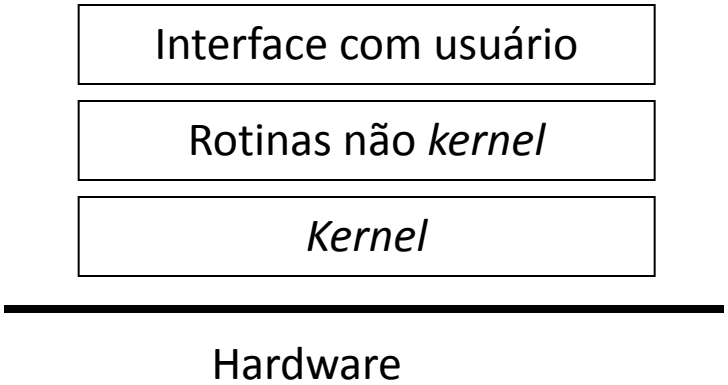
- Atualmente, a idéia de máquina virtual é utilizada em contextos diferentes:
  - Programas MS-DOS: rodam em computadores 32bits;
    - As chamadas feitas pelo MS-DOS ao Sistema Operacional são realizadas e monitoradas pelo monitor da máquina virtual (VMM);
      - Virtual 8086;
  - Programas JAVA (Máquina Virtual Java-JVM): o compilador Java produz código para a JVM (*bytecode*). Esse código é executado pelo interpretador Java:
    - Programas Java rodam em qualquer plataforma, independentemente do Sistema Operacional;

Estrutura dos Sistemas Operacionais – Máquina Virtual

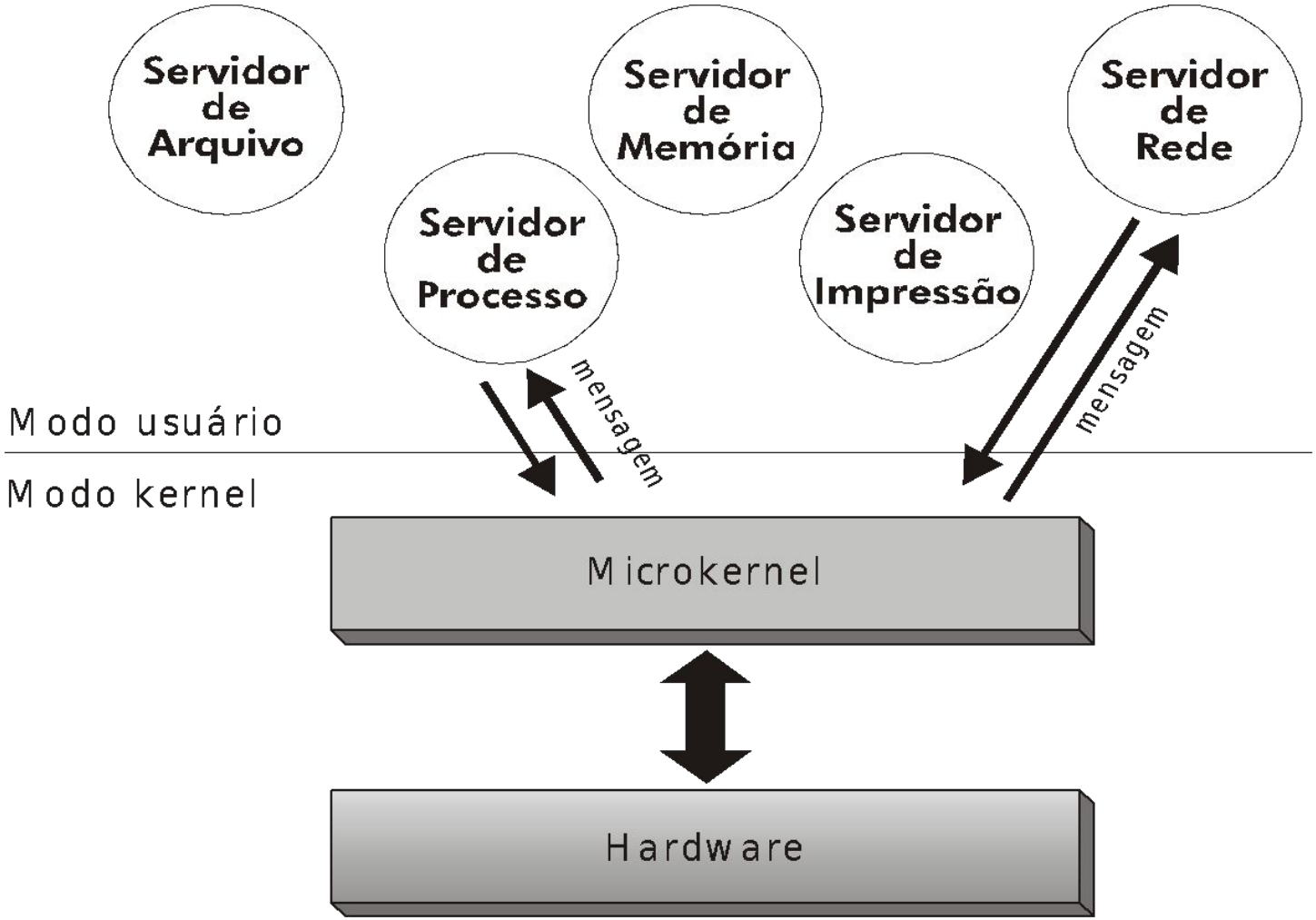
- Vantagens
  - Flexibilidade;
- Desvantagem:
  - Simular diversas máquinas virtuais não é uma tarefa simples sobrecarga;

Estrutura dos Sistemas Operacionais – Baseados em *Kernel* (núcleo)

- *Kernel* é o núcleo do Sistema Operacional
- Provê um conjunto de funcionalidades e serviços que suportam várias outras funcionalidades do SO
- O restante do SO é organizado em um conjunto de *rotinas não-kernel*



Estrutura dos Sistemas Operacionais – *Micro-Kernel*



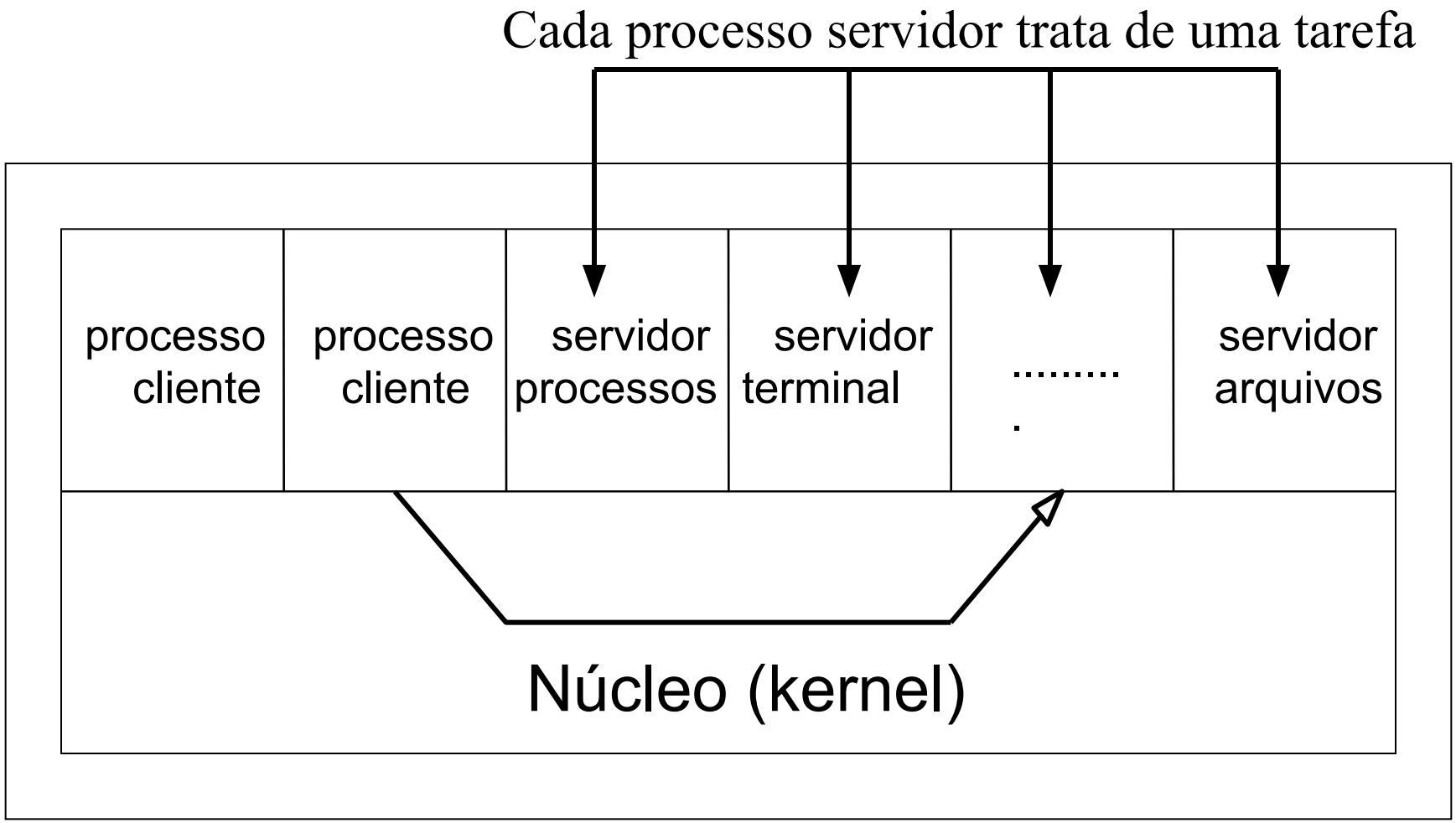
Estrutura dos Sistemas Operacionais – *Micro-Kernel*

- Minix 3
  - Código aberto www.minix3.org
  - 3200 linhas de código em C
    - Gerencia e escalona processos de chaveamento
    - Controla a comunicação entre os processos
    - 35 chamadas ao sistema
  - 800 linhas de código em Assembler
    - Funções de nível muito baixo
    - Contenção de interrupções e processos de chavameamento

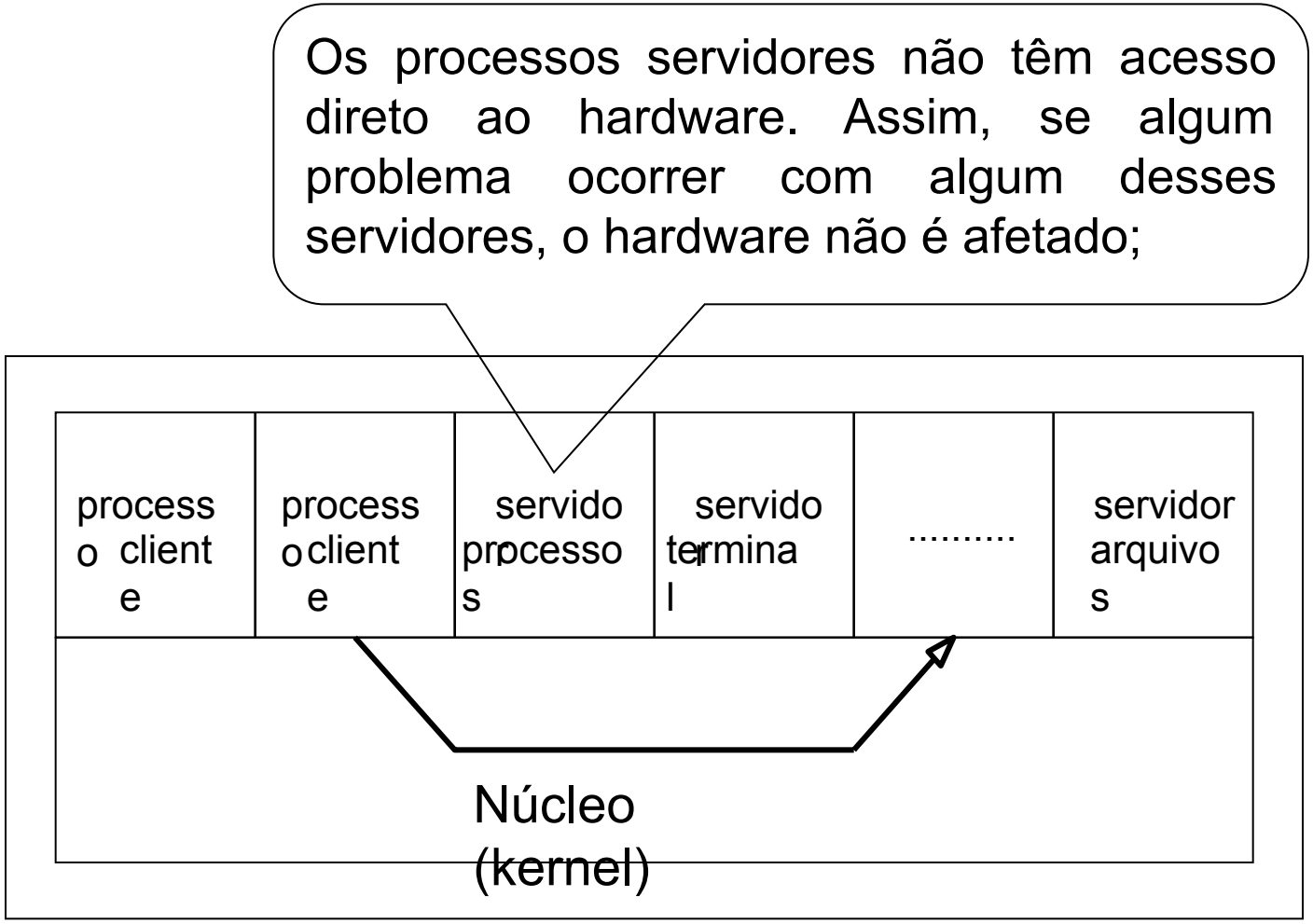
Estrutura dos Sistemas Operacionais – Cliente/Servidor

- Reduzir o Sistema Operacional a um nível mais simples:
  - ***Kernel***: implementa a comunicação entre processos clientes e processos servidores Núcleo mínimo;
  - Maior parte do Sistema Operacional está implementado como processos de usuários (nível mais alto de abstração);
  - Sistemas Operacionais Modernos;

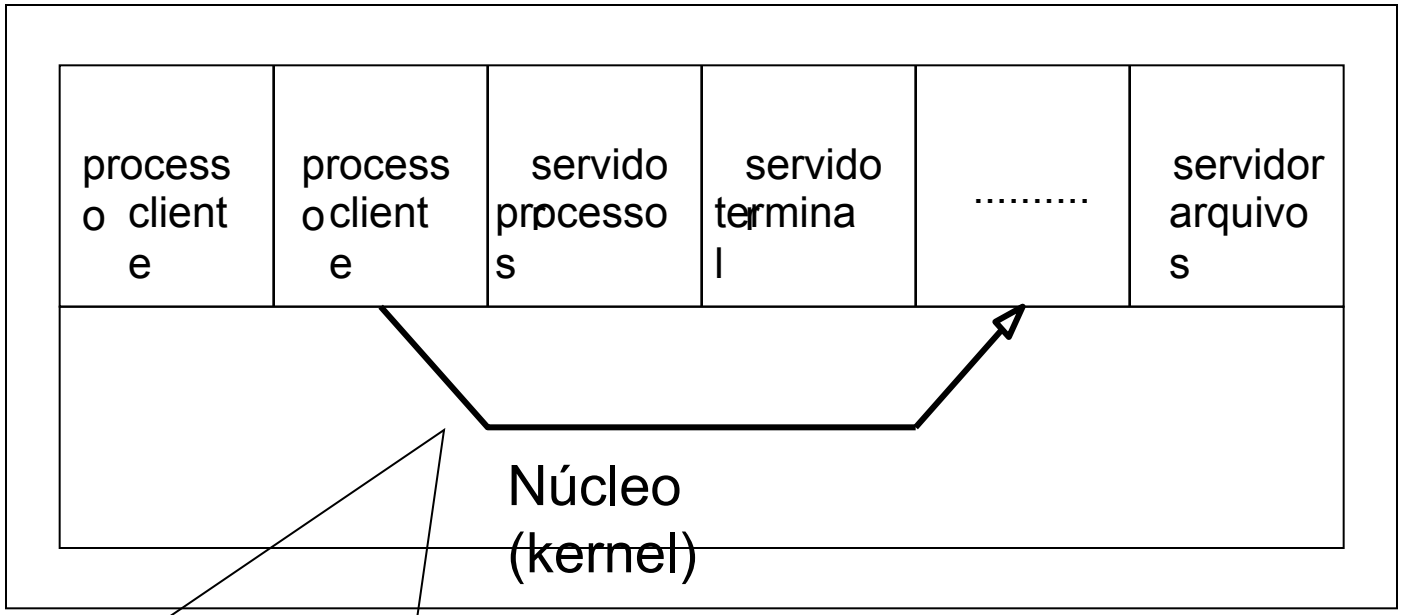
Estrutura dos Sistemas Operacionais – Cliente/Servidor



Estrutura dos Sistemas Operacionais – Cliente/Servidor



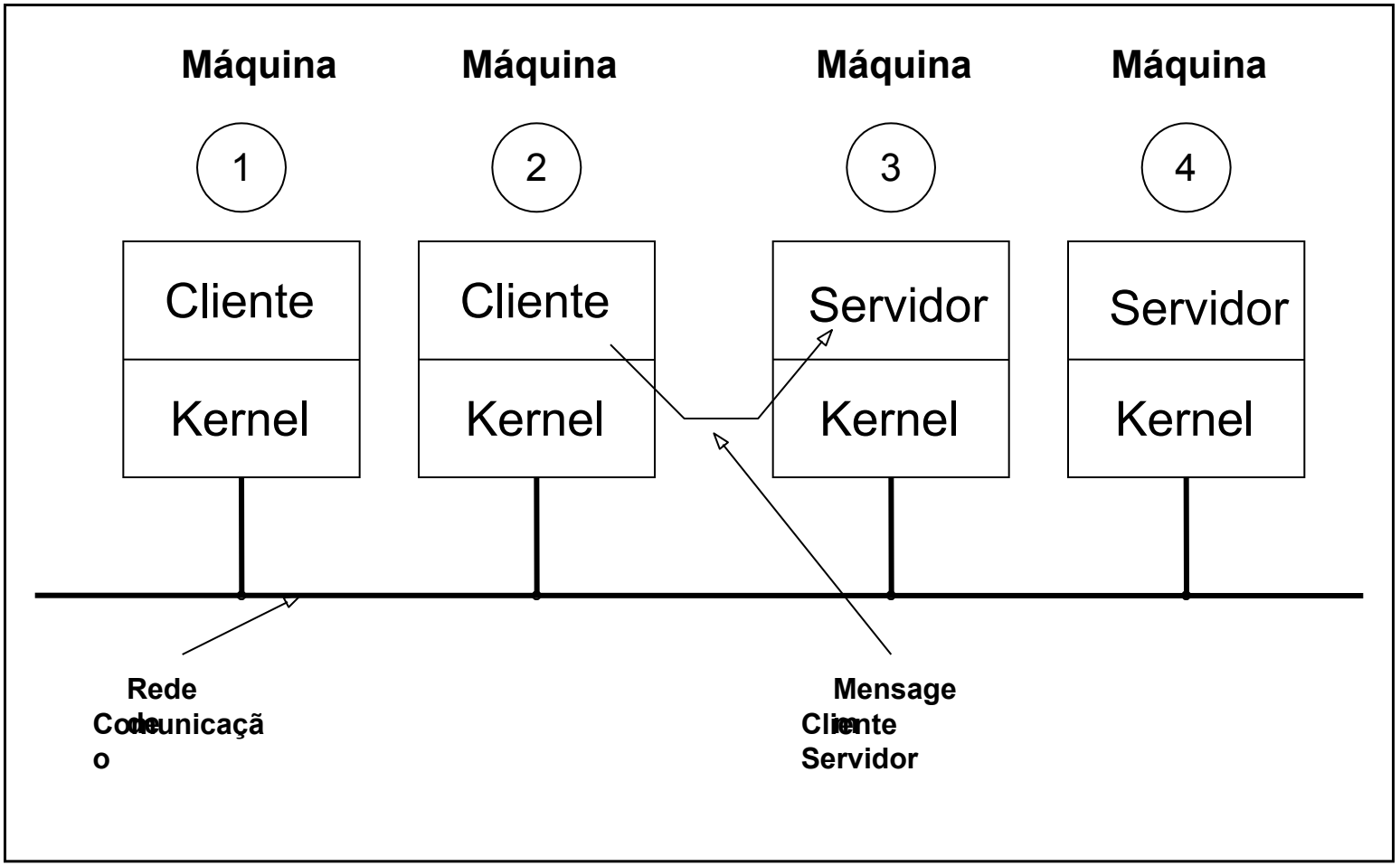
Estrutura dos Sistemas Operacionais – Cliente/Servidor



O mesmo não se aplica aos serviços que controlam os dispositivos de E/S, pois essa é uma tarefa difícil de ser realizada no modo usuário devido à limitação de endereçamento. Sendo assim, essa tarefa ainda é feita no *kernel*.

Estrutura dos Sistemas Operacionais – Cliente/Servidor

- Adaptável para Sistemas Distribuídos;



Processos

- Introdução
- Escalonamento de Processos
- Comunicação entre Processos
- Threads
- Deadlock

Processos

- Multiprogramação:
  - Pseudoparalelismo: coleção de processos sendo executados alternadamente na CPU;
- Um processo é caracterizado por um programa em execução, mas existe uma diferença sutil entre processo e programa:
  - Receita de Bolo.

Criando Processos

- Processos precisam ser criados e finalizados a todo o momento:
  - Inicialização do sistema;
  - Execução de uma chamada ao sistema de criação de processo realizada por algum processo em execução;
  - Requisição de usuário para criar um novo processo;
  - Inicialização de um processo em *batch* – *mainframes* com sistemas em *batch*;

Criando Processos

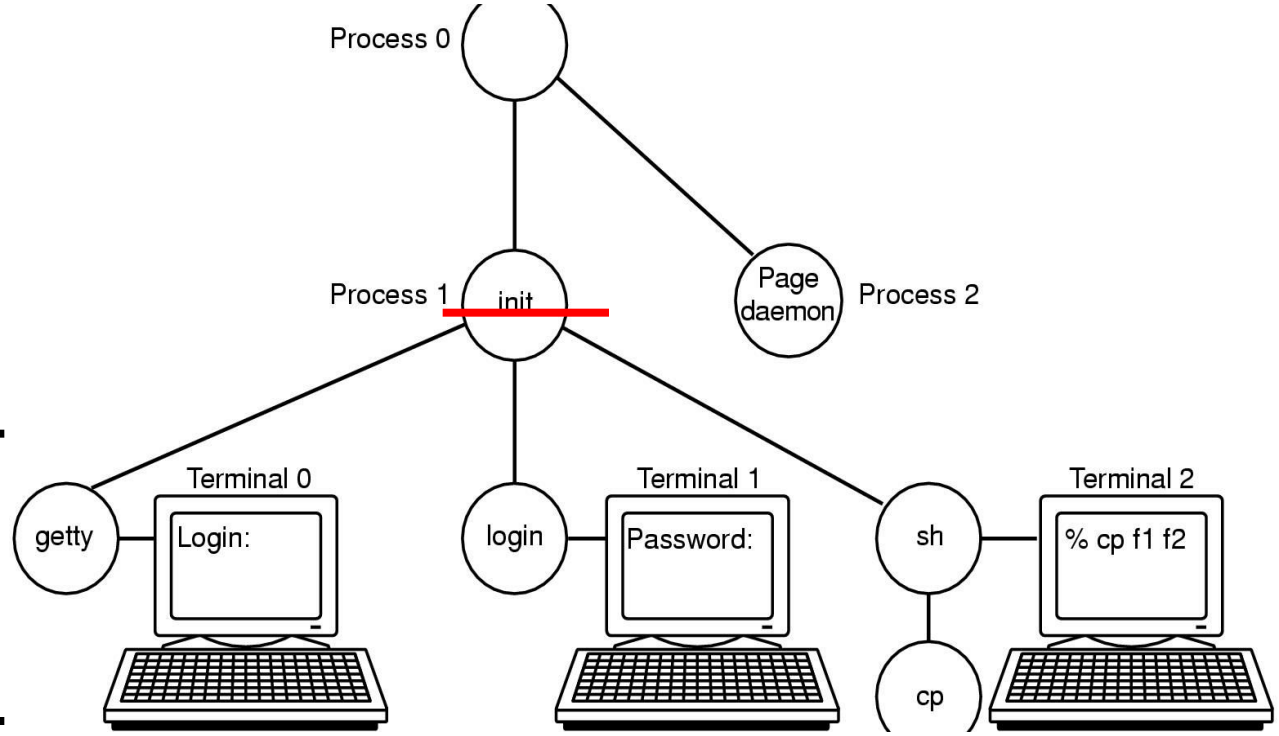
- Processos podem ser:
  - Específicos para usuários específicos:
    - Leitura de um arquivo;
    - Iniciar um programa (linha de comando ou um duplo clique no mouse);
  - Com funções específicas, que independem de usuários, que são criados pelo sistema operacional e que são processados em segundo plano (*daemons*):
    - Recepção e envio de emails;
    - Serviços de Impressão;

Criando Processos

- UNIX:
  - `Fork;`
    - Cria um processo idêntico (filho) ao processo que a chamou (pai), possuindo a mesma imagem de memória, as mesmas cadeias de caracteres no ambiente e os mesmos arquivos abertos;
    - Depois, o processo filho executa uma chamada para mudar sua imagem de memória e executar um novo programa
- Windows:
  - `CreateProcess`
    - Uma única função trata tanto do processo de criação quanto da carga do programa correto no novo processo

Criando Processos

- Exemplo UNIX:
  - Processo `init`: gera vários processos filhos para atender os vários terminais que existem no sistema;



Outros processos são gerados nos terminais

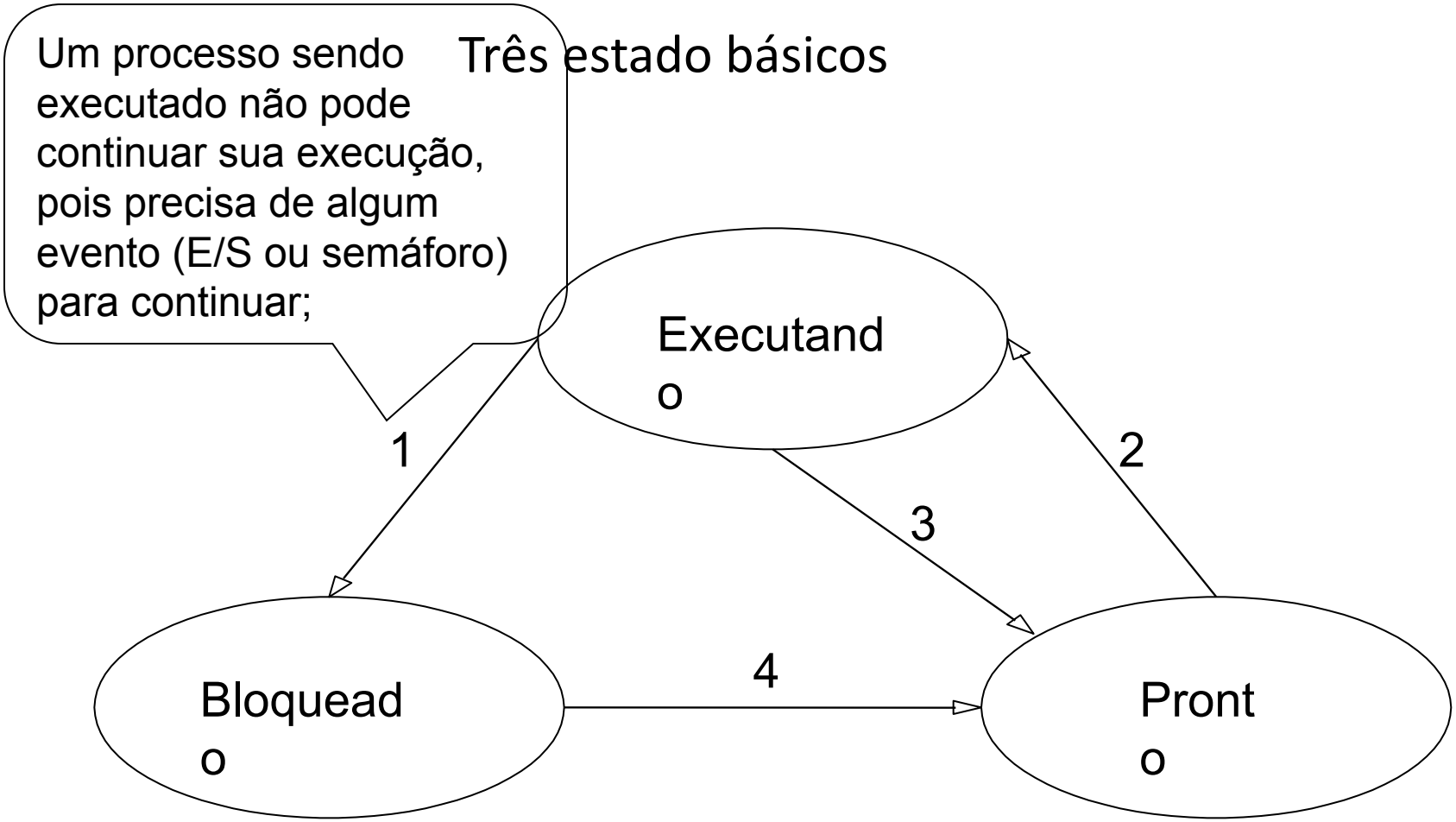
Finalizando Processos

- Condições:
  - Término normal (voluntário):
    - A tarefa a ser executada é finalizada;
    - Chamadas: `exit` *(UNIX) e* `ExitProcess` *(Windows)*
  - Término com erro (voluntário):
    - O processo sendo executado não pode ser finalizado: gcc filename.c, o arquivo filename.c não existe;

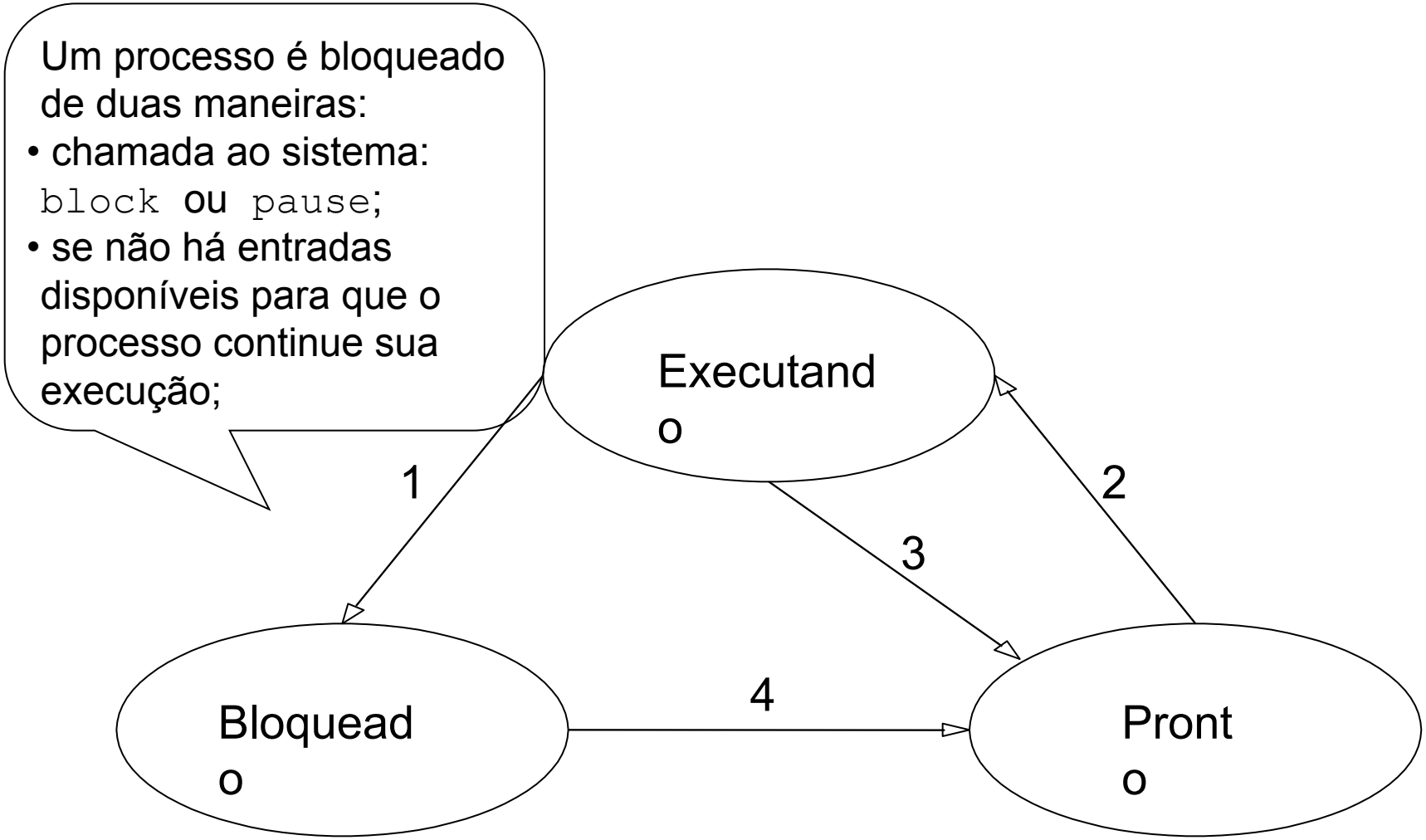
Finalizando Processos

- Condições (continuação):
  - Término com erro fatal (involuntário);
    - Erro causado por algum erro no programa (*bug*):
      - Divisão por 0 (zero)*;*
      - Referência à memória inexistente ou não pertencente ao processo;
      - Execução de uma instrução ilegal;
  - Término causado por algum outro processo (involuntário):
    - `Kill` (UNIX) e `TerminateProcess` (Windows)*;*

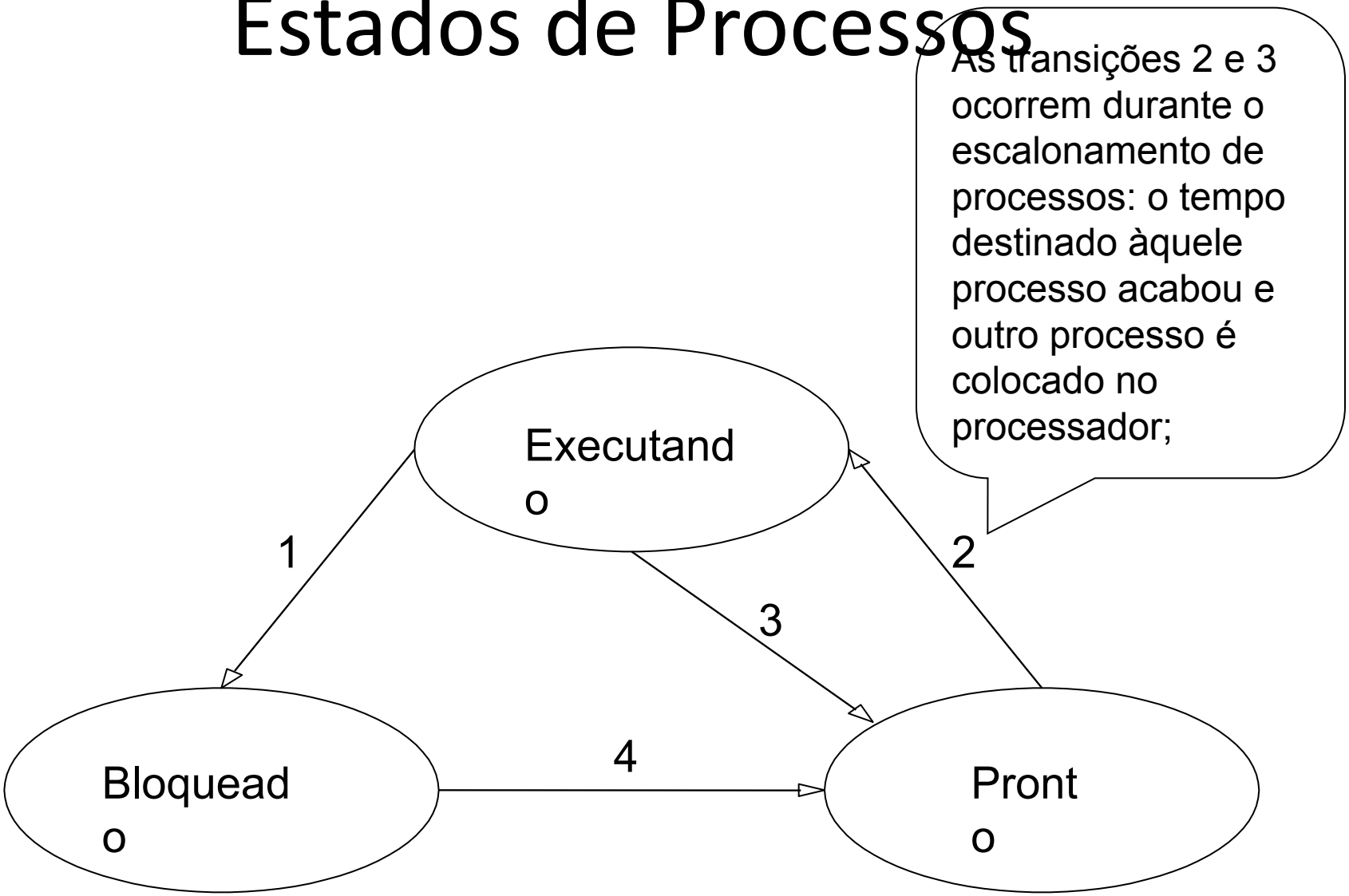
Estados de Processos



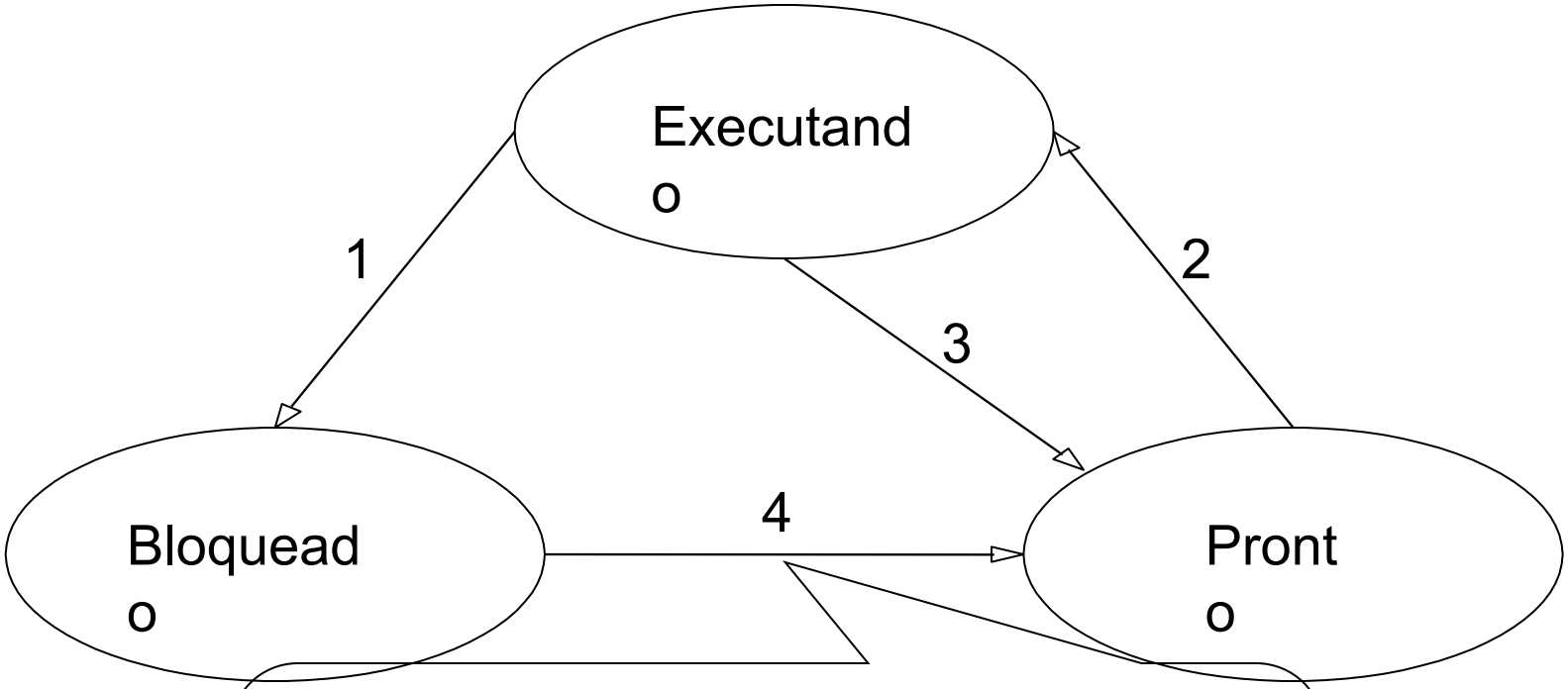
Estados de Processos



Estados de Processos



Estados de Processos



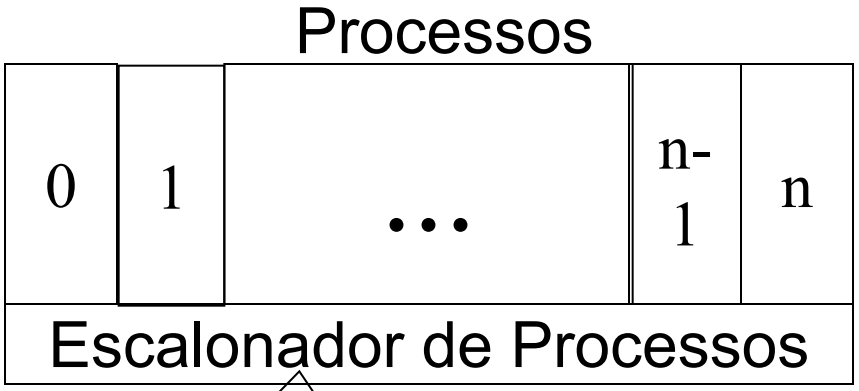
A transição 4 ocorre quando o evento esperado pelo processo bloqueado ocorre:

- se o processador está parado, o processo é executado imediatamente (2);
- se o processador está ocupado, o processo deve esperar sua vez;

Processos

- Processos *CPU-bound* (orientados à CPU): processos que utilizam muito o processador;
  - Tempo de execução é definido pelos ciclos de processador;
- Processos *I/O-bound* (orientados à E/S): processos que realizam muito E/S;
  - Tempo de execução é definido pela duração das operações de E/S;
- **IDEAL**: existir um balanceamento entre processos *CPU-bound* e *I/O-bound*;

Escalonador de Processos

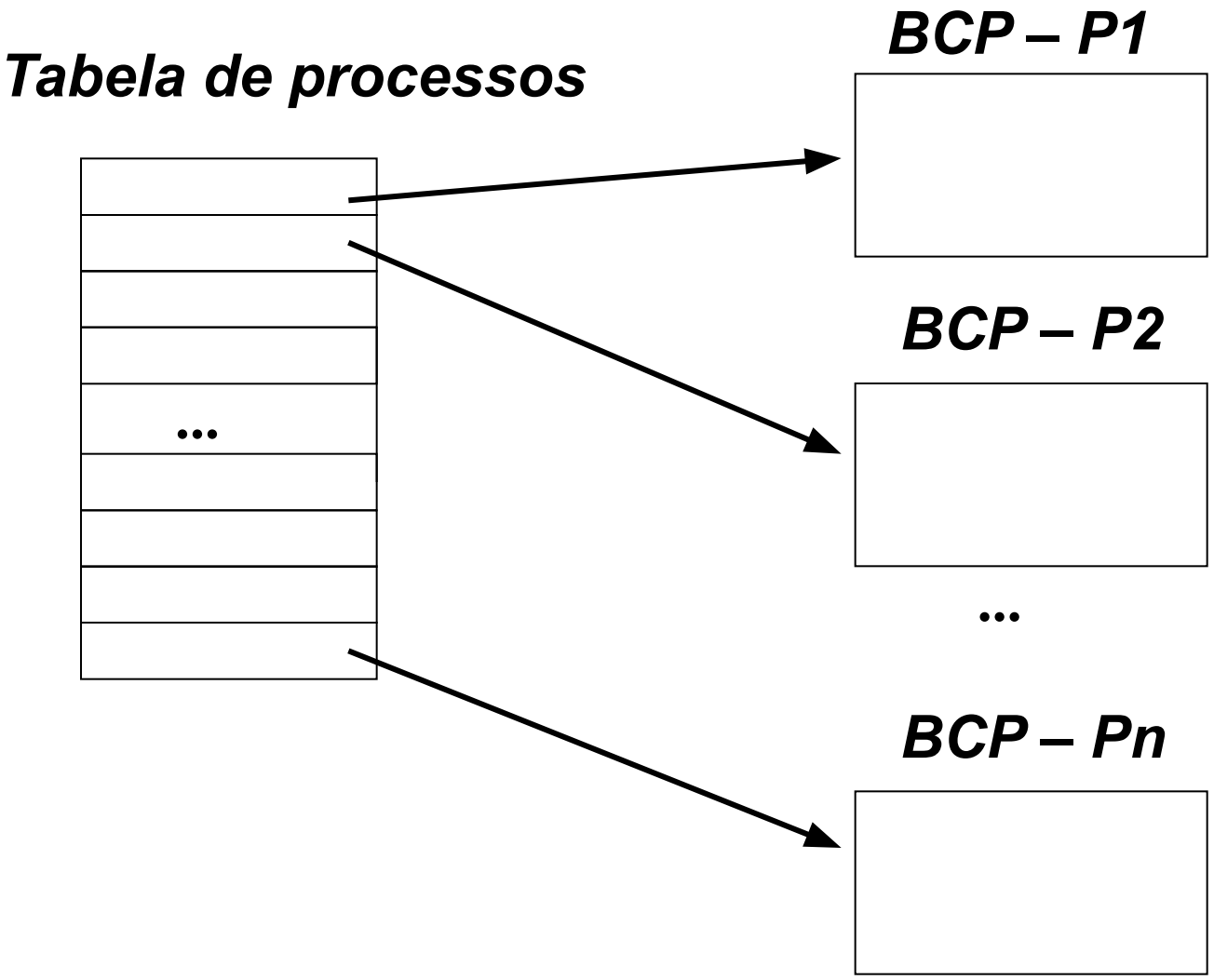


- Nível mais baixo do SO;
- Manipulação de interrupções e processos;

Implementação de Processos

- Tabela de Processos:
  - Cada processo possui uma entrada;
  - Cada entrada possui um ponteiro para o bloco de controle de processo (BCP) ou descritor de processo;
  - BCP possui todas as informações do processo contextos de hardware, software, endereço de memória;

Implementação de Processos



Implementação de Processos

Process management	**Memory management**	**File management**
Registers	Pointer to text segment	Root directory
Program counter	Pointer to data segment	Working directory
Program status word	Pointer to stack segment	File descriptors
Stack pointer		User ID
Process state		Group ID
Priority		
Scheduling parameters		
Process ID		
Parent process		
Process group		
Signals		
Time when process started		
CPU time used		
Children's CPU time		
Time of next alarm		

Algumas informações do BCP

Processos

- Introdução
- Escalonamento de Processos
- Comunicação entre Processos
- Threads
- Deadlock

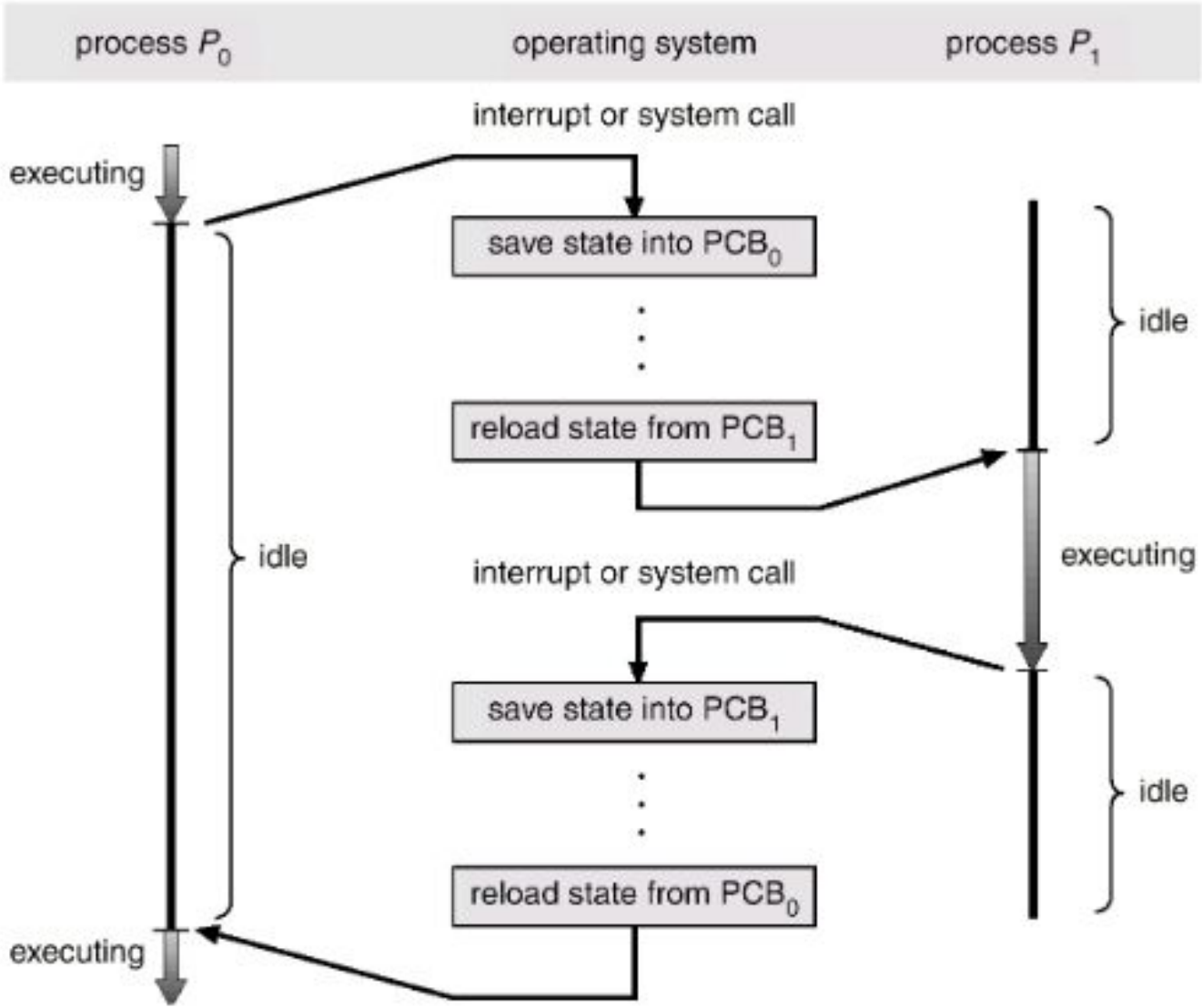
Escalonamento de Processos

- Escalonador de Processos escolhe o processo que será executado pela CPU;
- Escalonamento é realizado com o auxílio do hardware;
- Escalonador deve se preocupar com a eficiência da CPU, pois o chaveamento de processos é complexo e custoso:
  - Afeta desempenho do sistema e satisfação do usuário;
- Escalonador de processo é um processo que deve ser executado quando da **mudança de contexto** (troca de processo);

Escalonamento de Processos

- Mudança de Contexto:
  - Overhead de tempo;
  - Tarefa cara:
    - Salvar as informações do processo que está deixando a CPU em seu BCP conteúdo dos registradores;
    - Carregar as informações do processo que será colocado na CPU copiar do BCP o conteúdo dos registradores;

Troca de Contexto entre processos



Escalonamento de Processos

- Situações nas quais escalonamento é necessário:
  - Um novo processo é criado;
  - Um processo terminou sua execução e um processo pronto deve ser executado;
  - Quando um processo é bloqueado (semáforo, dependência de E/S), outro deve ser executado;
  - Quando uma interrupção de E/S ocorre o escalonador deve decidir por: executar o processo que estava esperando esse evento; continuar executando o processo que já estava sendo executado ou executar um terceiro processo que esteja pronto para ser executado;

Escalonamento de Processos

- Hardware de relógio fornece interrupções de relógio e a decisão do escalonamento pode ser tomada a cada interrupção ou a cada *k* interrupções;
- Algoritmos de escalonamento podem ser divididos em duas categorias dependendo de como essas interrupções são tratadas:
    - Preemptivo: escolhe um processo e o deixa executando por um tempo máximo;
    - Não-preemptivo: estratégia de permitir que o processo que está sendo executado continue sendo executado até ser bloqueado por alguma razão (semáforos, operações de E/S-interrupção) ou que libere a CPU voluntariamente;

Escalonamento de Processos

- Categorias de Ambientes:
  - **Sistemas em Batch**: usuários não esperam por respostas rápidas; algoritmos não-preemptivos ou preemptivos com longo intervalo de tempo;
  - **Sistemas Interativos**: interação constante do usuário; algoritmos preemptivos; Processo interativo espera comando e executa comando;
  - **Sistemas em Tempo Real**: processos são executados mais rapidamente;; tempo é crucial sistemas críticos;

Escalonamento de Processos

- Características de algoritmos de escalonamento:
  - Qualquer sistema:
    - **Justiça** (*Fairness*)*:* cada processo deve receber uma parcela justa de tempo da CPU;
    - **Balanceamento**: diminuir a ociosidade do sistema;
    - **Políticas do sistema** – prioridade de processos;

Escalonamento de Processos

- Características de algoritmos de escalonamento:
  - Sistemas em *Batch*:
    - **Vazão** (*throughput*): maximizar o número de *jobs* executados por hora;
    - **Tempo de retorno** (*turnaround time*): tempo no qual o processo espera para ser finalizado;
    - **Eficiência**: CPU deve estar 100% do tempo ocupada;
  - Sistemas Interativos:
    - **Tempo de resposta**: tempo esperando para iniciar execução;
    - **Proporcionalidade**: satisfação do usuários;

Escalonamento de Processos

- Características de algoritmos de escalonamento:
  - Sistemas em Tempo Real:
    - **Cumprimento dos prazos**: prevenir perda de dados;
    - **Previsibilidade**: prevenir perda da qualidade dos serviços oferecidos;

Exercícios

1. Na sua concepção, qual a importância dos Sistemas Operacionais?
2. O que você entende por Sistema Operacional?
3. Para você Sistema Operacional é o mesmo que Sistema Computacional?
4. O que é um processo?
5. Quais atividades são responsabilidade do Sistema Operacional no seu ponto de vista?
6. Quais são os estados que um processo pode assumir?
7. Faça o diagrama dos estados de um processo e explique os eventos que ocorrem entre eles.
8. Quais os tipos (modelos estruturais) de sistemas operacionais que existem?
9. O que significa um processo sofrer preempção?
10. Qual a relação entre programa e processo?

Exercícios

1. Na sua concepção, qual a importância dos Sistemas Operacionais?
2. O que você entende por Sistema Operacional?
3. Para você Sistema Operacional é o mesmo que Sistema Computacional?
4. O que é um processo?
5. Quais atividades são responsabilidade do Sistema Operacional no seu ponto de vista?
6. Quais são os estados que um processo pode assumir?
7. Faça o diagrama dos estados de um processo e explique os eventos que ocorrem entre eles.
8. Quais os tipos (modelos estruturais) de sistemas operacionais que existem?
9. O que significa um processo sofrer preempção?
10. Qual a relação entre programa e processo?

Próxima aula ...

Mais sobre Escalonamento de Processos...

- Introdução
- Escalonamento de Processos
  - Algoritmos de Escalonamento
- Comunicação entre Processos
- Threads
- Deadlock

Interfaces de um Sistema Operacional

- Usuário – SO:
  - **Shell ou Interpretador de comandos**
- Programas – SO:
  - **Chamadas ao Sistema**

Tarefa para casa.....

- Ler capítulo 1 e 2 do Tanenbaum
- Estudar chamadas ao sistema (Linux) e começar a pensar em como implementar um programa que utilize algumas delas....