Conceitos básicos e serviços dos Sistemas Operacionais

Prof. Marcos Ribeiro Quinet de Andrade
Instituto de Ciência e Tecnologia - ICT
Universidade Federal Fluminense - UFF

Tipos de serviços do S.O.

- Um S.O. fornece um ambiente para a execução de programas através de **serviços** para os processos e para os usuários

- Apesar da forma como esses serviços são oferecidos variar de sistema para sistema (e entre diferentes versões de um mesmo serviço), existem algumas **classes de serviços** que são comuns a todos os sistemas operacionais

Tipos de serviços do S.O.

- O serviços mais comuns gerenciados pelos sistemas operacionais são:
  - Execução de programas;
  - Operações de entrada/saída;
  - Manipulação do sistema de arquivos;
  - Detecção de erros;
  - Alocação de recursos;
  - Proteção.

Conceitos Básicos de Sistemas Operacionais

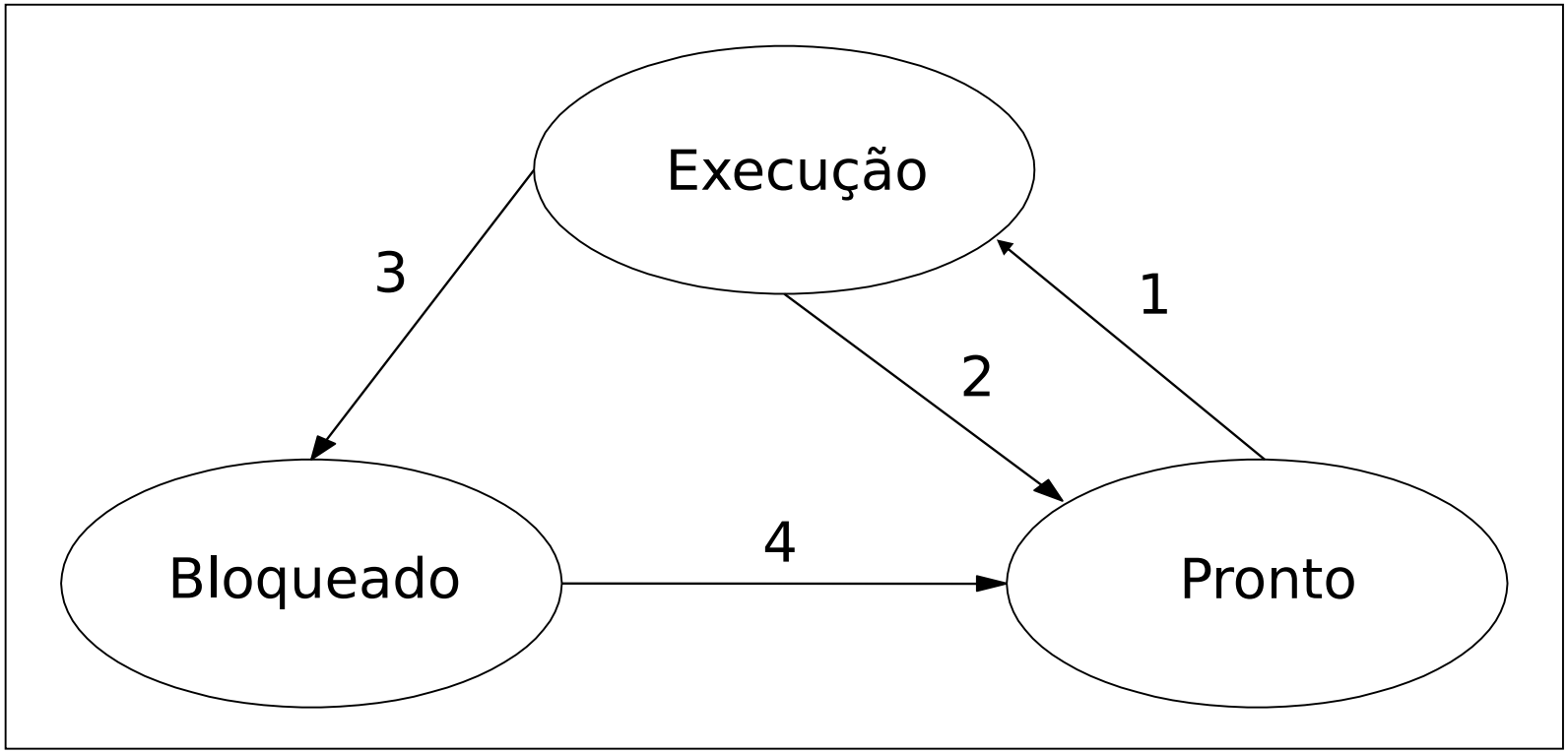
- Principais conceitos:
  - Processos;
  - Gerenciamento de memória;
  - Chamadas ao sistema;
  - Estrutura do sistema operacional.

Processos

- Processo: chave do S.O.;
  - Caracterizado por um programa em execução;
  - Cada processo possui:
    - Um espaço de endereçamento;
    - Uma lista de alocação de memória (mínimo, máximo);
    - Um conjunto de registradores (contador de programa);
  - O Sistema Operacional controla todos os processos, sejam os criados pelos usuários, e os criados por ele mesmo;

Processos

- Estados básicos de um processo:



Processos

- Cada um dos estados de um processo são caracterizados por:
  - **Execução**: em um sistema monoprocessado, apenas um processo pode estar fazendo uso da seção de processamento da UCP por vez; em sistemas multiprocessados, apesar de existirem vários processos em execução simultânea, cada núcleo de processamento executa somente um por vez
  - **Bloqueado**: quando um processo não pode prosseguir com sua execução, pois necessita que algum evento ocorra antes;
  - **Pronto**: o processo encontra-se carregado na memória, pronto para iniciar sua execução, bastando somente ser selecionado pelo escalonador de processos.

Processos

- Ex.: processo bloqueado (suspenso)
  - Quando o SO suspende um processo *P1* temporariamente para executar um processo *P2*, o processo *P1* deve ser reiniciado exatamente no mesmo estado no qual estava ao ser suspenso.
  - Para tanto, todas as informações a respeito do processo *P1* são armazenadas em uma **tabela de processos** (*process table*). Essa tabela é implementada como um vetor ou uma lista encadeada de estruturas.

Processos

- Um processo pode resultar na execução de outros processos, chamados de processos-filhos:
  - Características para a hierarquia de processos:
    - Comunicação (Interação) e Sincronização;
    - Segurança e proteção;
    - Uma árvore de, no máximo, três níveis;
- Escalonadores de processos – processo que escolhe qual será o próximo processo a ser executado;
  - Diversas técnicas para escalonamento de processos;

Processos

- Comunicação e sincronismo entre processos – possíveis soluções:
  - Semáforos;
  - Monitores;
  - Instruções especiais em hardware;
  - Troca de mensagens;

Gerenciamento de Memória

- Gerenciamento elementar (década de 60)
  - Sistemas monoprogramados;
  - Sem paginação:
    - Apenas um processo na memória;
    - Acesso a toda a memória;

- Gerenciamento mais avançado (atualidade)
  - Sistemas multiprogramados;
  - Mais de um processo na memória;
  - Chaveamento de processos: por entrada/saída ou por limite de tempo (sistema de tempo compartilhado);

Compartilhamento de Memória

- Partições Fixas
  - Cada processo é alocado em uma dada partição da memória (pré-definida);
  - Partições são liberadas quando o processo termina;

- Partições Variáveis
  - A memória é alocada de acordo com o tamanho e número de processos;
  - Otimiza o uso da memória;

Comunicação entre usuário e o S.O.

- **Chamadas ao Sistema** (*system calls*) fornecem uma interface entre um programa em execução e o S.O. Estão, geralmente, disponíveis como instruções nas linguagens de baixo nível ou até mesmo em linguagens de alto nível, como C.

- Podem ser classificadas em duas categorias:
  - Controle de processos.
  - Gerenciamento de arquivos e de dispositivos de E/S.

Chamadas ao Sistema

- **Modos de Acesso**:
  - Modo usuário;
  - Modo *kernel,* ou Supervisor ou Núcleo;
  - São determinados por um conjunto de bits localizados no registrador de status do processador: PSW (*program status word*);
    - Por meio desse registrador, o hardware verifica se a instrução pode ou não ser executada pela aplicação;
  - Protege o próprio *kernel* do Sistema Operacional na RAM contra acessos indevidos;

Chamadas ao Sistema

- Modo usuário:
  - Aplicações não têm acesso direto aos recursos da máquina, ou seja, ao hardware;
  - Quando o processador trabalha no modo usuário, a aplicação só pode executar **instruções sem privilégios**, com um acesso reduzido de instruções;
  - Por que? Para garantir a **segurança** e a **integridade do sistema**;

Chamadas ao Sistema

- Modo *kernel:*
  - Aplicações têm acesso direto aos recursos da máquina, ou seja, ao hardware;
  - **Operações com privilégios**;
  - Quando o processador trabalha no modo *kernel*, a aplicação tem **acesso ao conjunto total de instruções**;
  - Apenas o S.O. tem acesso às instruções privilegiadas;

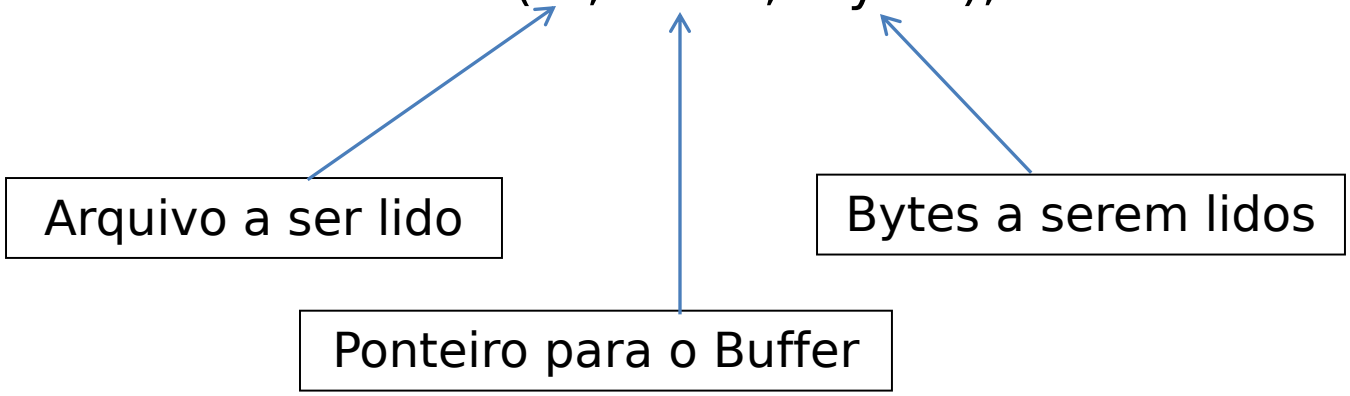
Alteração entre modos

- Se uma aplicação precisa realizar alguma instrução privilegiada, ela realiza uma **chamada ao sistema *(system call)***, que altera do modo usuário para o modo *kernel*;
- Chamadas de sistemas são a **porta de entrada** para o modo *kernel*;
  - São a interface entre os programas do usuário no modo usuário e o Sistema Operacional no modo *kernel*;
  - As chamadas diferem de SO para SO, no entanto, os conceitos relacionados às chamadas são similares independentemente do SO;

Execução de chamadas ao sistema

- **TRAP**: instrução que permite o acesso ao modo *kernel*; **transfere o controle para o SO**
  - Exemplo: a seguinte instrução do UNIX:

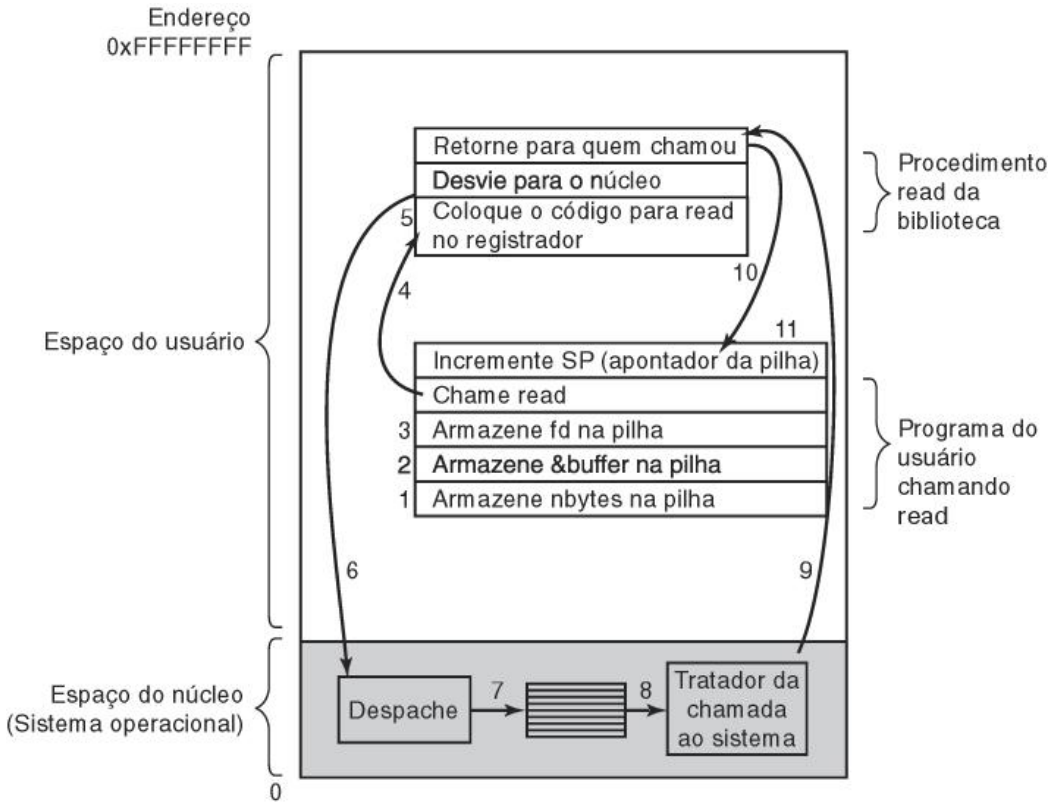
count = read(fd,buffer,nbytes);



O programa sempre deve checar o retorno da chamada de sistema para saber se algum erro ocorreu!!!

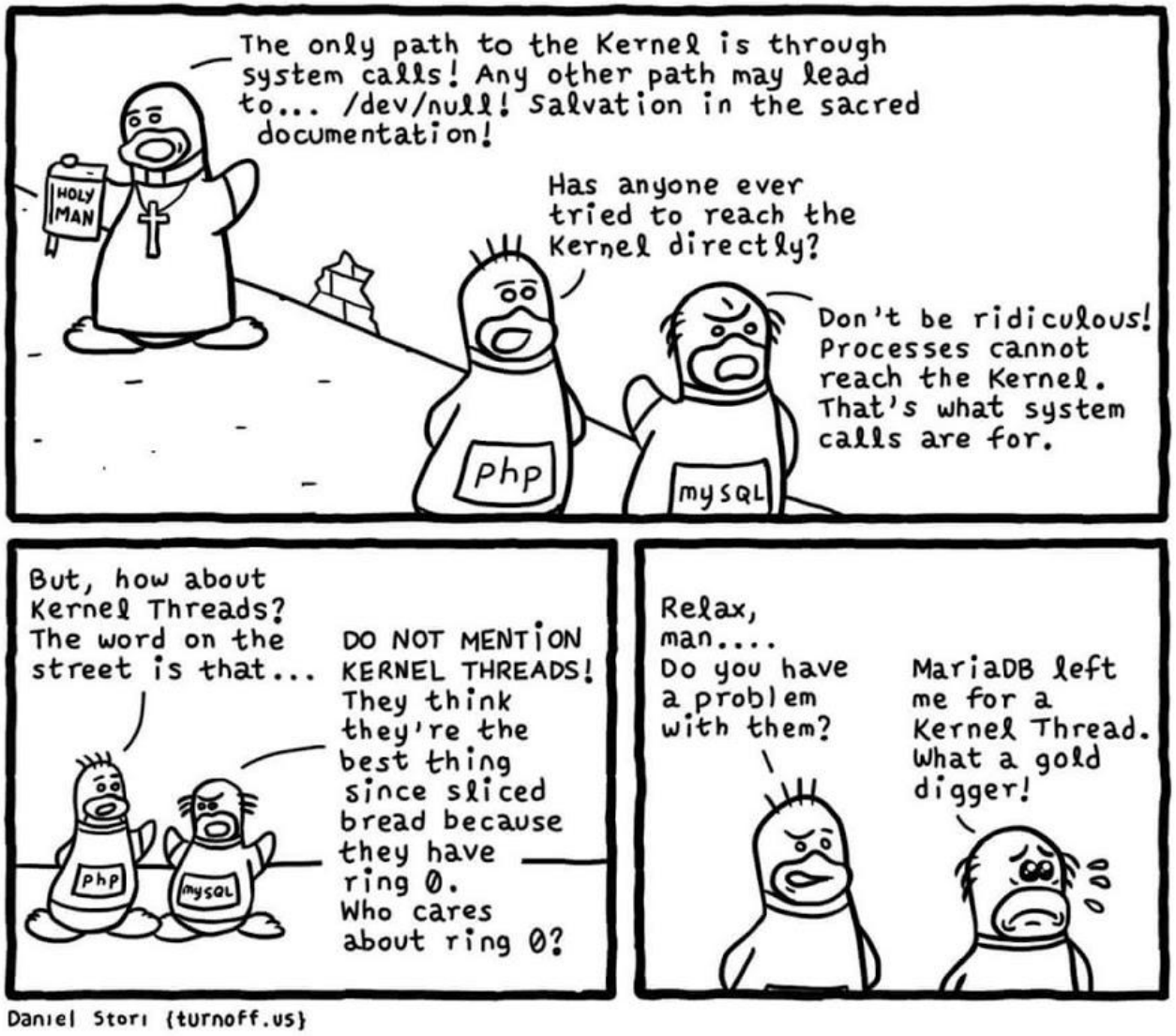
Um exemplo de chamada ao sistema

- Os 11 passos para fazer uma chamada ao sistema para o comando "*read (arq, buffer, nbytes)*"



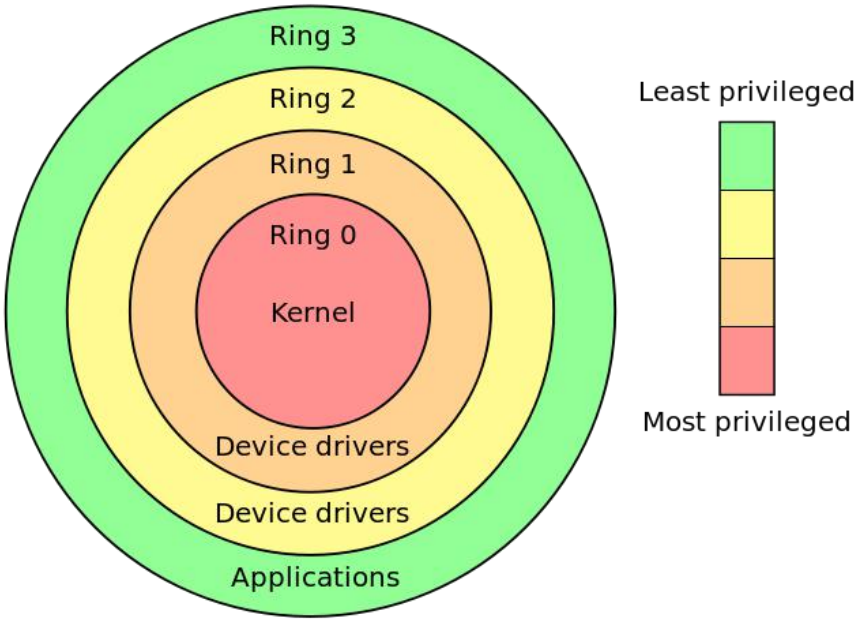
- Após o passo 5, é executado um TRAP, passando do modo usuário para o modo sistema

Chamadas ao Sistema



Anéis de Proteção

- *Rings* ou simplesmente anéis de proteção, definem um sistema conhecido por domínios de proteção hierárquica, para garantir a segurança do SO;
- São implementados portões de controle entre os níveis para maior controle dos processos.



Exemplos de chamadas ao sistema

Gerenciamento de processos

Chamada	Descrição
pid = fork( )	Crie um processo filho idêntico ao processo pai
pid = waitpid(pid, &statloc, options)	Aguarde um processo filho terminar
s = execve(name, argv, environp)	Substitua o espaço de endereçamento do processo
exit(status)	Termine a execução do processo e retorne o estado

Gerenciamento de arquivos

Chamada	Descrição
fd = open(file, how, ...)	Abra um arquivo para leitura, escrita ou ambas
s = close(fd)	Feche um arquivo aberto
n = read(fd, buffer, nbytes)	Leia dados de um arquivo para um buffer
n = write(fd, buffer, nbytes)	Escreva dados de um buffer para um arquivo
position = lseek(fd, offset, whence)	Mova o ponteiro de posição do arquivo
s = stat(name, &buf)	Obtenha a informação de estado do arquivo

Exemplos de chamadas ao sistema

Gerenciamento do sistema de diretório e arquivo

Chamada	Descrição
s = mkdir(name, mode)	Crie um novo diretório
s = rmdir(name)	Remova um diretório vazio
s = link(name1, name2)	Crie uma nova entrada, name2, apontando para name1
s = unlink(name)	Remova uma entrada de diretório
s = mount(special, name, flag)	Monte um sistema de arquivo
s = umount(special)	Desmonte um sistema de arquivo

Diversas

Chamada	Descrição
s = chdir(dirname)	Altere o diretório de trabalho
s = chmod(name, mode)	Altere os bits de proteção do arquivo
s = kill(pid, signal)	Envie um sinal a um processo
seconds = time(&seconds)	Obtenha o tempo decorrido desde 1º de janeiro de 1970

Chamadas ao Sistema API (*application program interface*) WIN32

Unix	Win32	Descrição
fork	CreateProcess	Crie um novo processo
waitpid	WaitForSingleObject	Pode esperar um processo sair
execve	(none)	CrieProcesso = fork + execve
exit	ExitProcess	Termine a execução
open	CreateFile	Crie um arquivo ou abra um arquivo existente
close	CloseHandle	Feche um arquivo
read	ReadFile	Leia dados de um arquivo
write	WriteFile	Escreva dados para um arquivo
lseek	SetFilePointer	Mova o ponteiro de posição do arquivo
stat	GetFileAttributesEx	Obtenha os atributos do arquivo
mkdir	CreateDirectory	Crie um novo diretório
rmdir	RemoveDirectory	Remova um diretório vazio
link	(none)	Win32 não suporta ligações (link)
unlink	DeleteFile	Destrua um arquivo existente
mount	(none)	Win32 não suporta mount
umount	(none)	Win32 não suporta mount
chdir	SetCurrentDirectory	Altere o diretório de trabalho atual
chmod	(none)	Win32 não suporta segurança (embora NT suporte)
kill	(none)	Win32 não suporta sinais
time	GetLocalTime	Obtenha o horário atual

Estrutura dos Sistemas Operacionais

Estrutura dos Sistemas Operacionais

- Principais tipos de estruturas:
  - Monolíticos;
  - Em camadas;
  - Micronúcleo;
  - Máquinas Virtuais;
  - Cliente-Servidor;

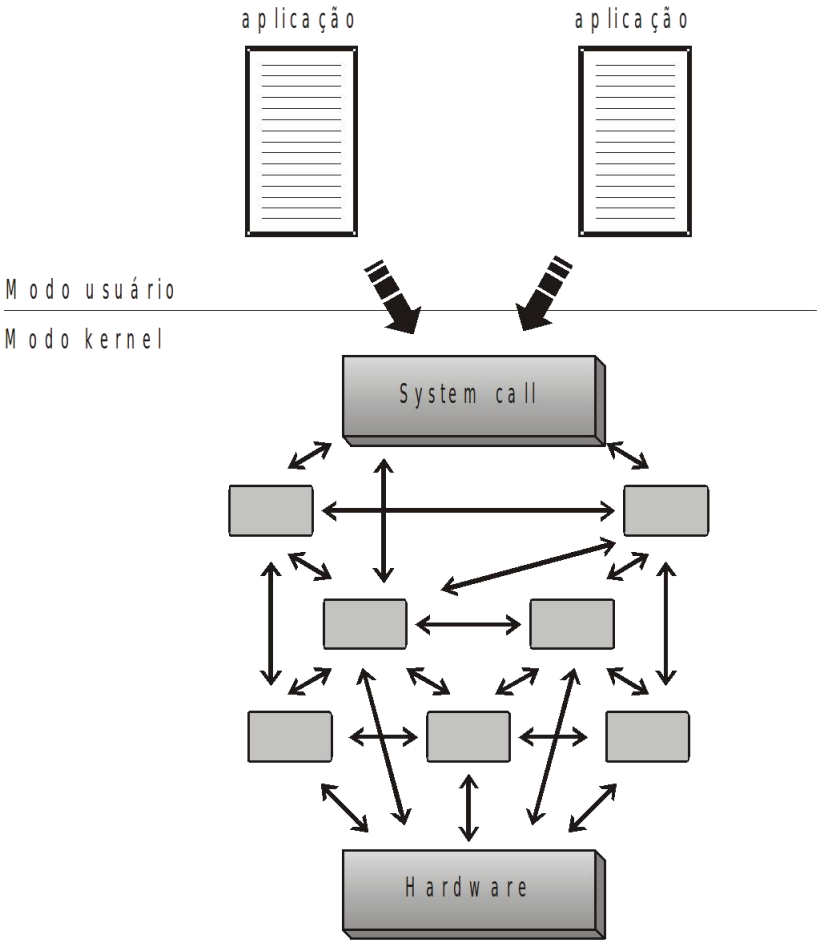
Arquiteturas de Sistemas Operacionais

- Arquitetura Monolítica
  - Todos os componentes do SO estão contidos no núcleo, comunicando-se diretamente entre si;
  - Rapidez na comunicação, mas complexidade no código;
  - 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;

Estrutura dos Sistemas Operacionais - Monolítico

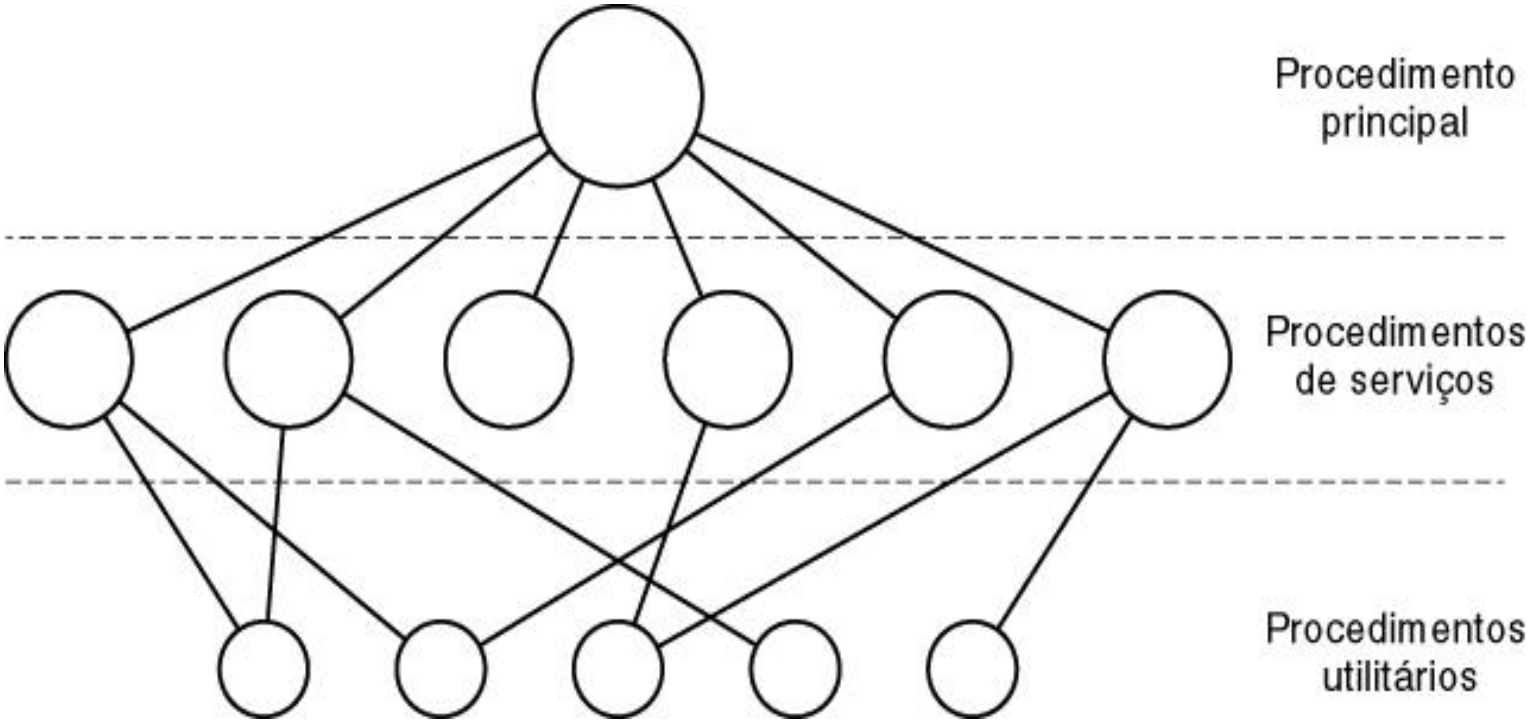
- Os serviços (chamadas) requisitados ao sistema são realizados por meio da colocação de parâmetros em registradores ou pilhas de serviços seguida da execução de uma instrução chamada *TRAP;*
- Cada processo possui uma interface bem definida com relação aos parâmetros e resultados para facilitar a comunicação com os outros processos;
- Simples;
- Primeiros sistemas UNIX e MS-DOS;

Estrutura dos Sistemas Operacionais - Monolítico



Estrutura dos Sistemas Operacionais - Monolítico

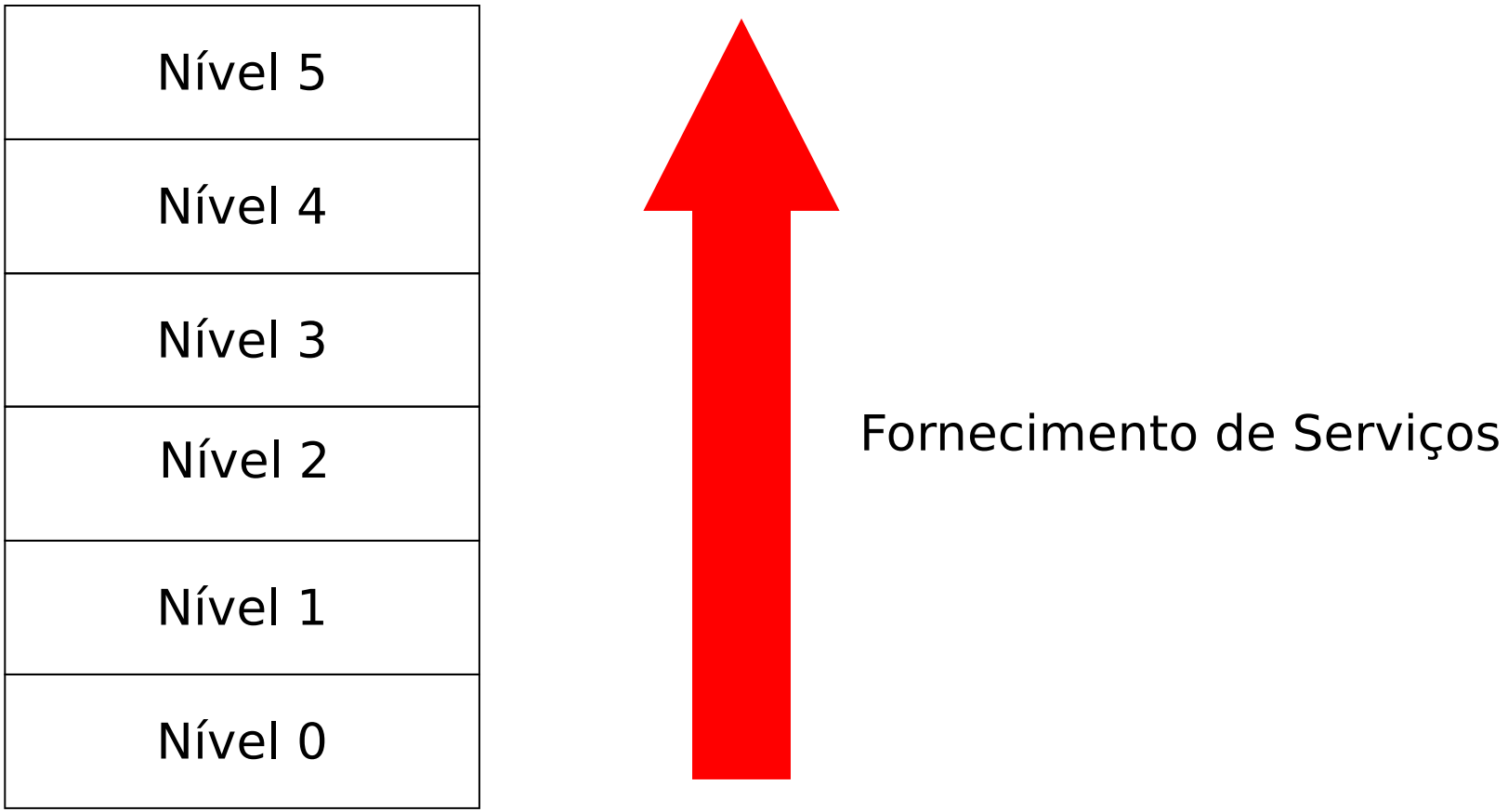
- Modelo de um sistema monolítico estruturado



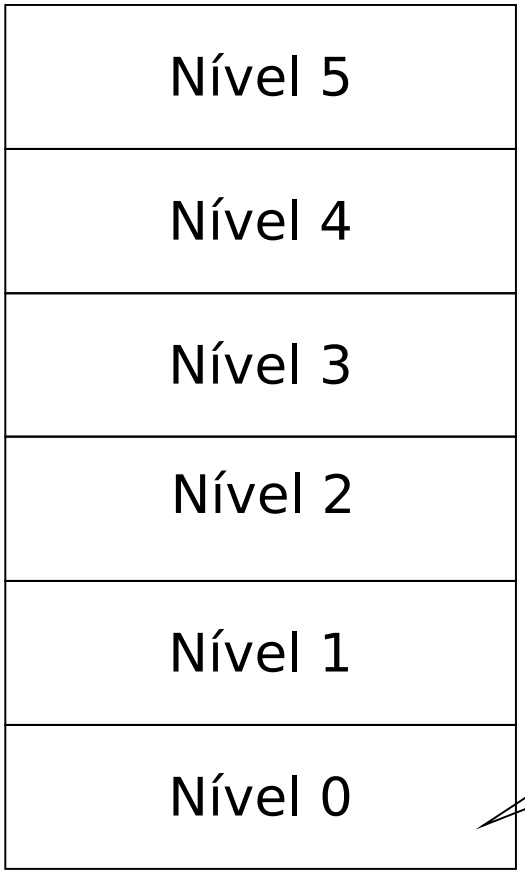
Estrutura dos Sistemas Operacionais – em Camadas

- Possui uma hierarquia de níveis;
- Componentes autocontidos, em camadas de componentes que realizam tarefas similares;
- Primeiro sistema em camadas: THE (idealizado por E.W. Dijkstra);
  - Possuia 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; o sistema torna-se mais lento;
- Atualmente: modelo de 2 camadas.

Estrutura dos Sistemas Operacionais – em Camadas

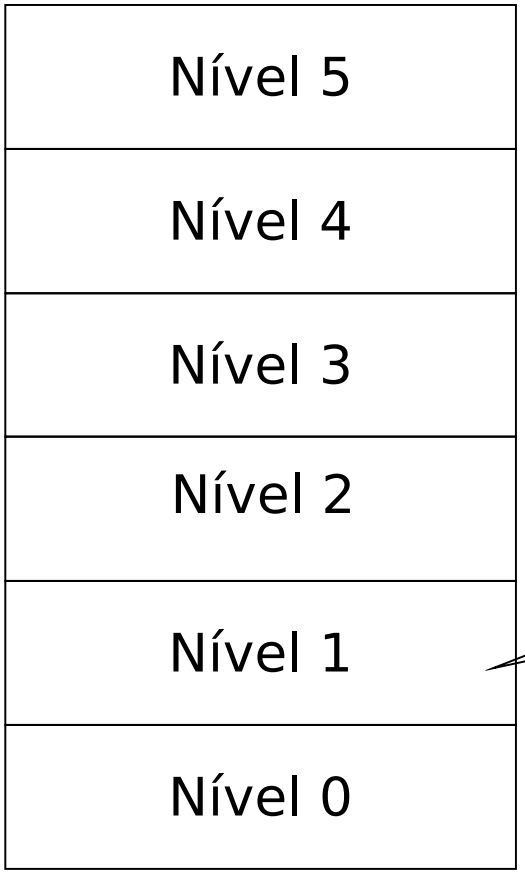


Estrutura dos Sistemas Operacionais – em Camadas



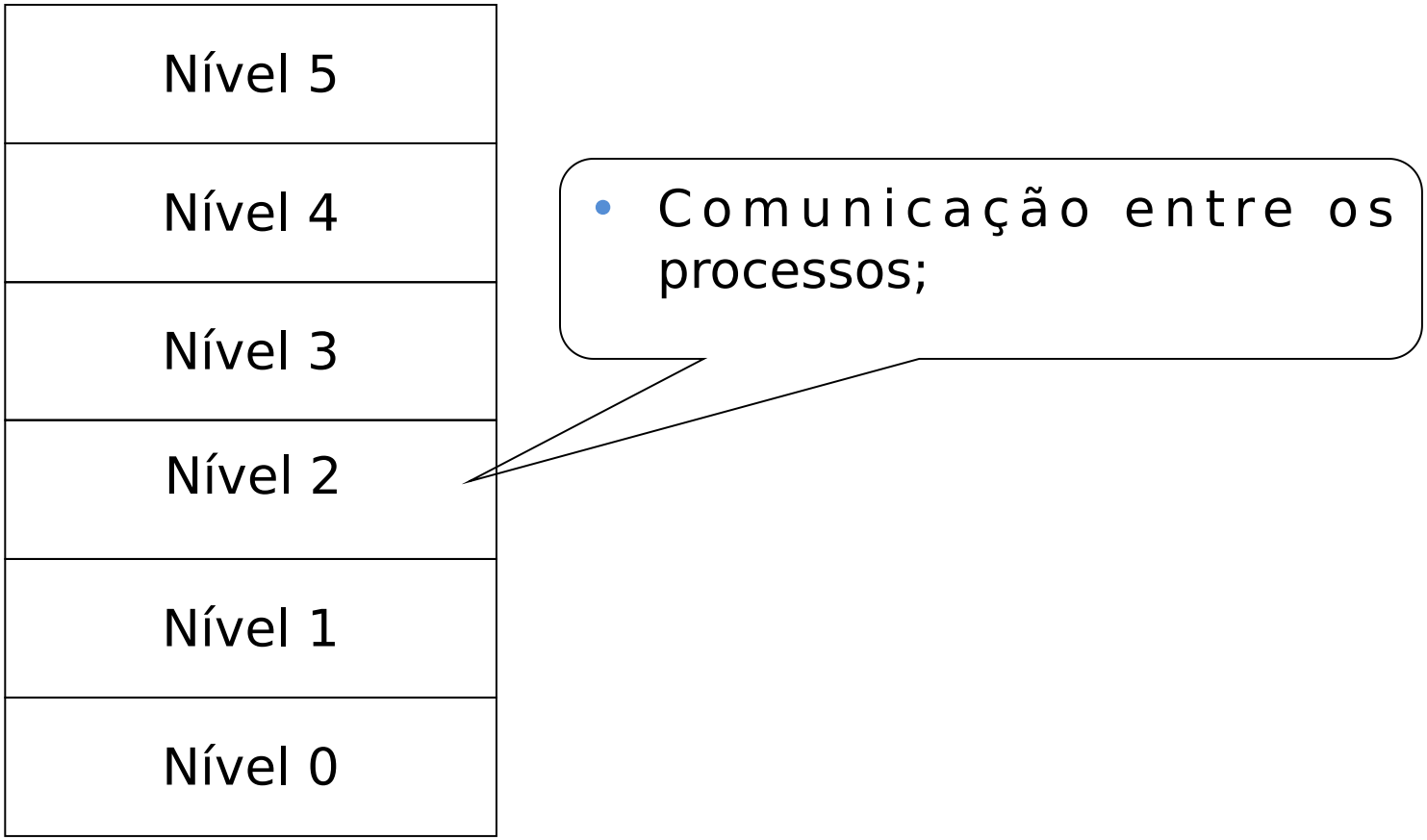
- Alocação do processador;
- Chaveamento entre os processos em execução – multiprogramação;

Estrutura dos Sistemas Operacionais – em Camadas

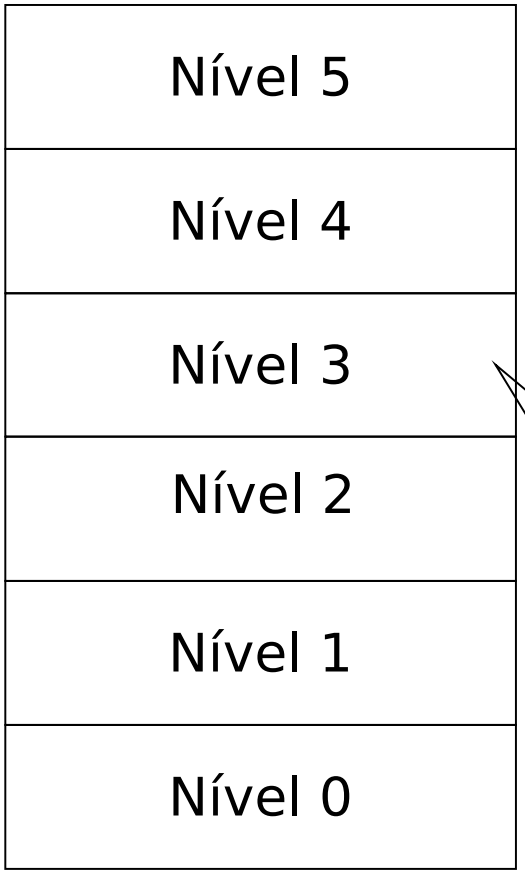


- Gerenciamento da memória;
- Alocação de espaço para processos na memória e no disco:
  - Processo dividido em partes (páginas) para ficarem no disco;

Estrutura dos Sistemas Operacionais – em Camadas

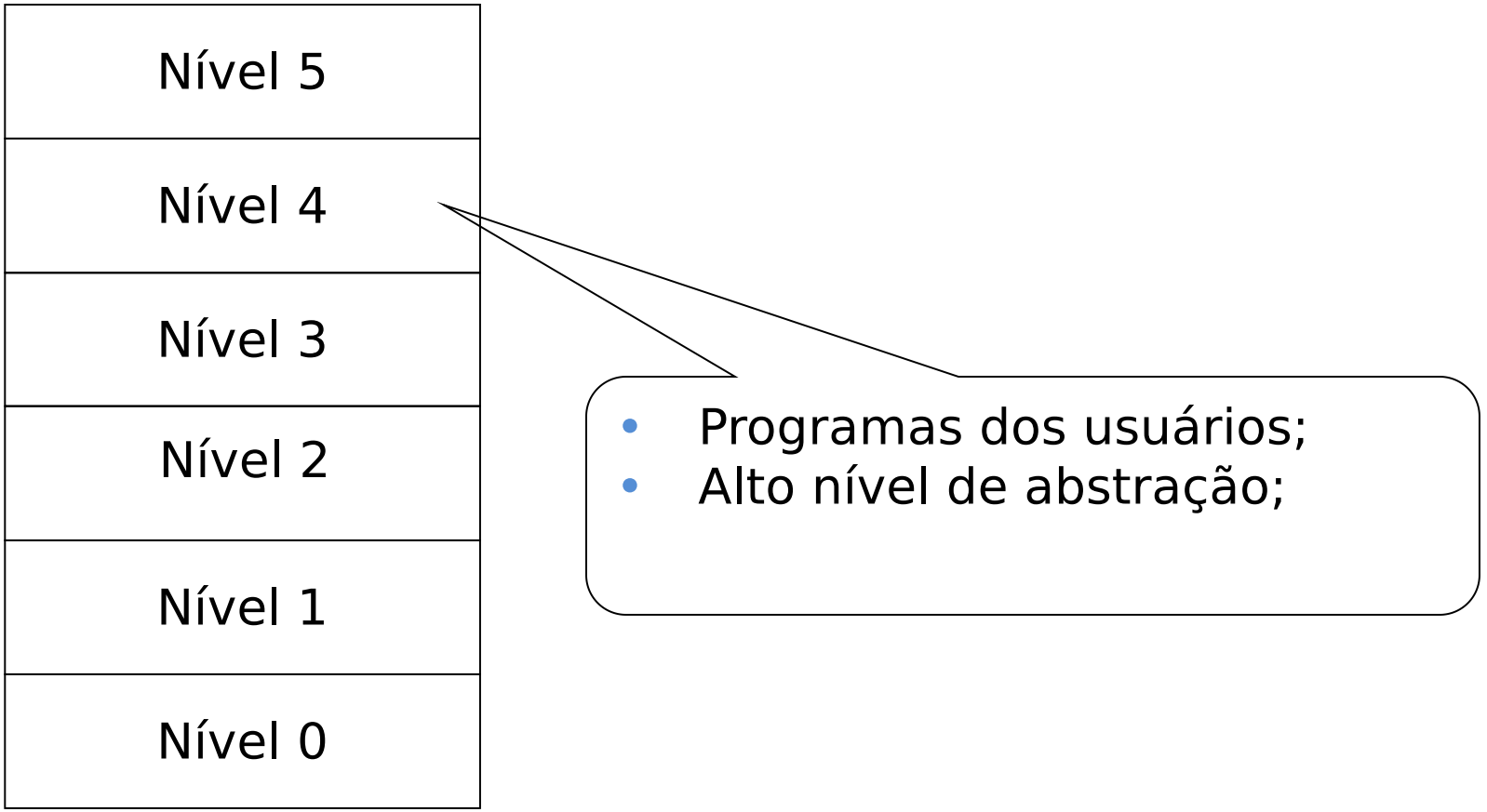


Estrutura dos Sistemas Operacionais – em Camadas

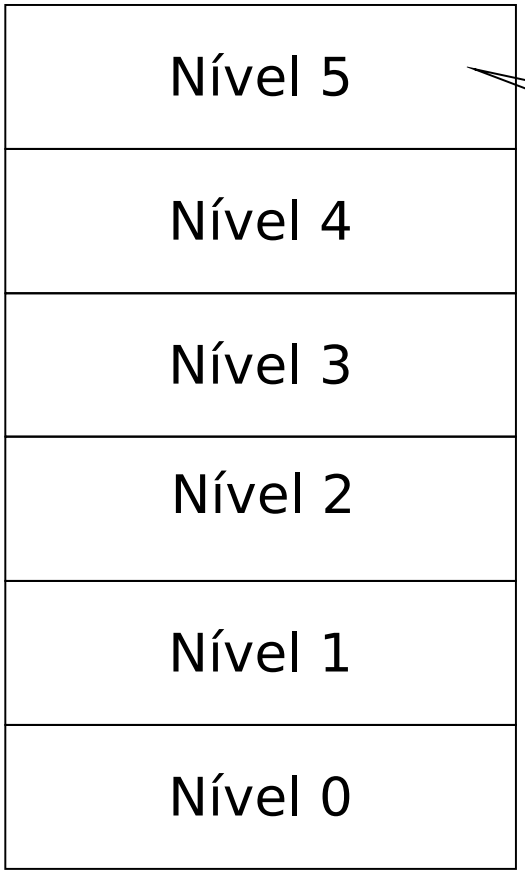


- Gerenciamento dos dispositivos de entrada/saída – armazenamento de informações de/para tais dispositivos;

Estrutura dos Sistemas Operacionais – em Camadas



Estrutura dos Sistemas Operacionais – em Camadas



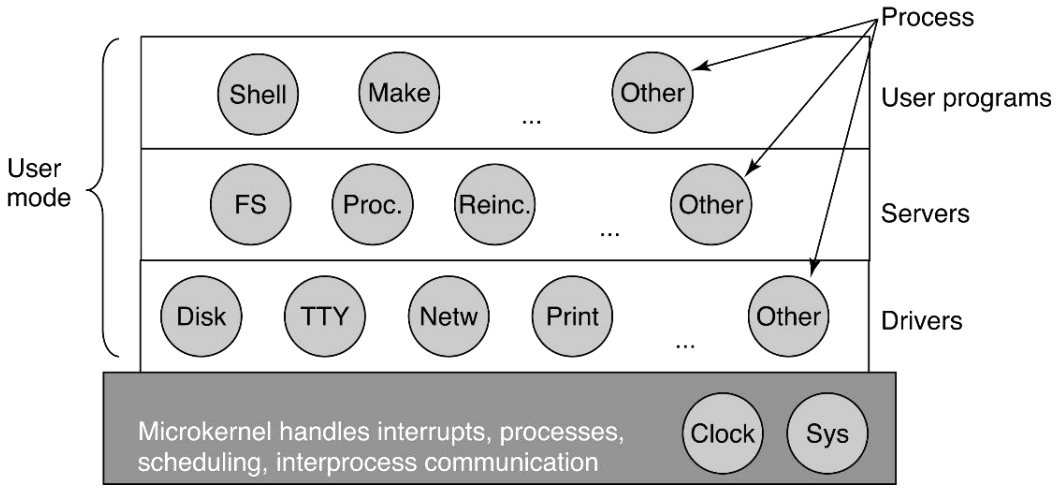
- Processo do operador do sistema;

Estrutura dos Sistemas Operacionais – Micronúcleo

- Nos sistemas em camadas, pode-se definir a fronteira entre os modos núcleo e usuário
  - Até então, todas as camadas entravam no núcleo, mas não é considerado uma boa prática
    - Erros em aplicações no núcleo podem derrubar todo o sistema
      - Estima-se que para todo programa existam 10 erros a cada 1000 linhas de código, algunss mais simples e outros mais graves → compromete a segurança e estabilidade do SO
  - A estrutura de micronúcleo propõe a divisão das funções do SO em módulos menores, com funcionalidades bem definidas
  - São usados em sistemas de tempo real, que necessitam de requisitos de confiabilidade muito altos

Estrutura dos Sistemas Operacionais – Micronúcleo

- Somente uma pequena parte dos serviços pode acessar diretamente o *hardware*
- A camada de servidores fazem a maior parte do trabalho do SO, como o sistema de arquivos (FS), gerenciador de processos (Proc.) e outros
- A ideia de um núcleo mínimo e a restrição de permissões de processos e drivers limita enormemente os danos que podem ser causados por um componente defeituoso

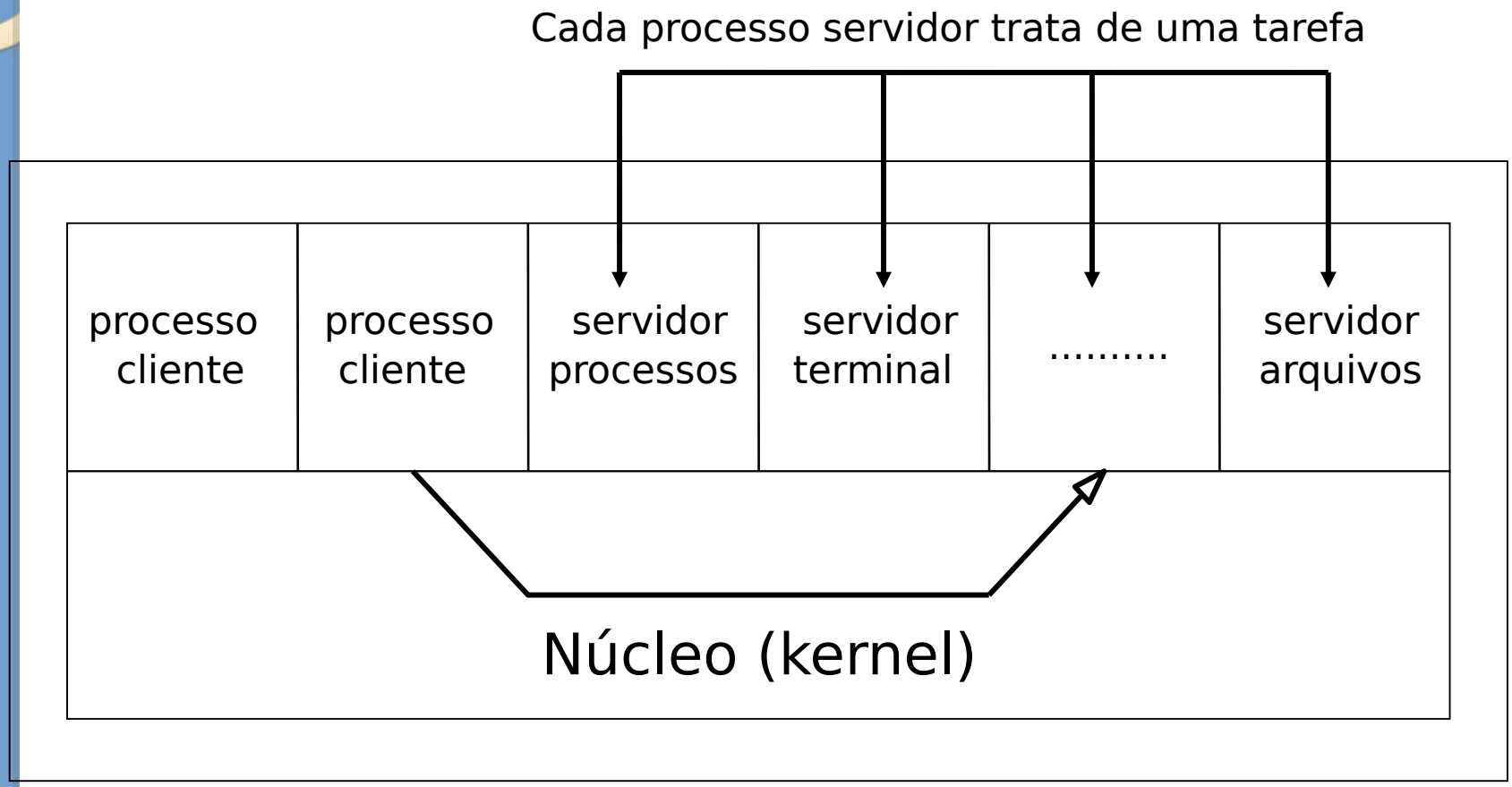


Estrutura dos Sistemas Operacionais – Cliente/Servidor

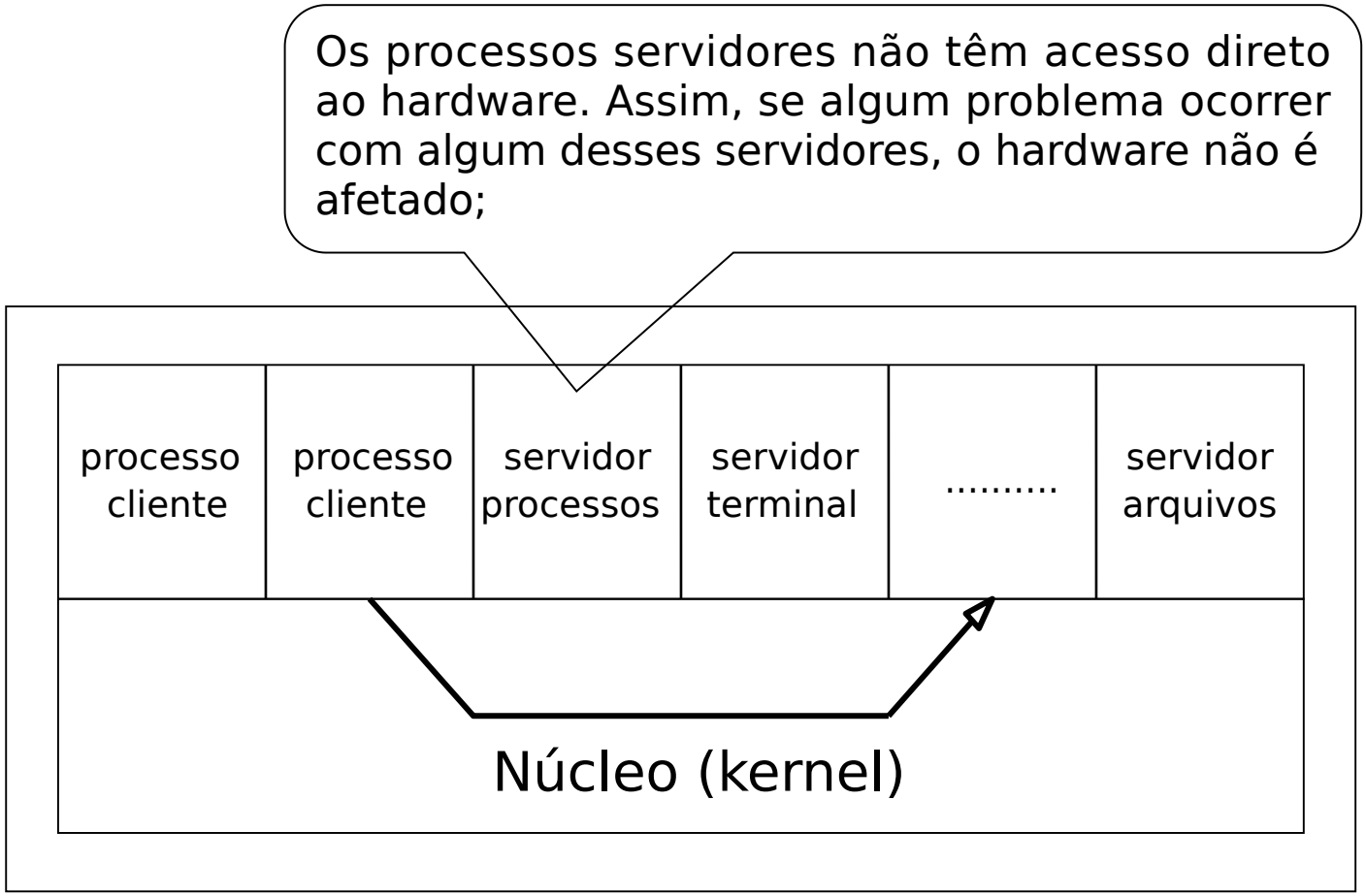
- A partir do modelo de micronúcleo, pode-se distinguir os processos em duas classes:
  - servidores: aqueles que prestam algum serviço;
  - clientes: aqueles que usam estes serviços.

- ***Kernel***: implementa a comunicação entre processos clientes e processos servidores com troca de mensagens → Núcleo mínimo;
- A maior parte do Sistema Operacional está implementado como processos de usuários (nível mais alto de abstração - não importa se é usado em um único computador ou em uma rede de computadores);
- Empregado pelos sistemas operacionais modernos;

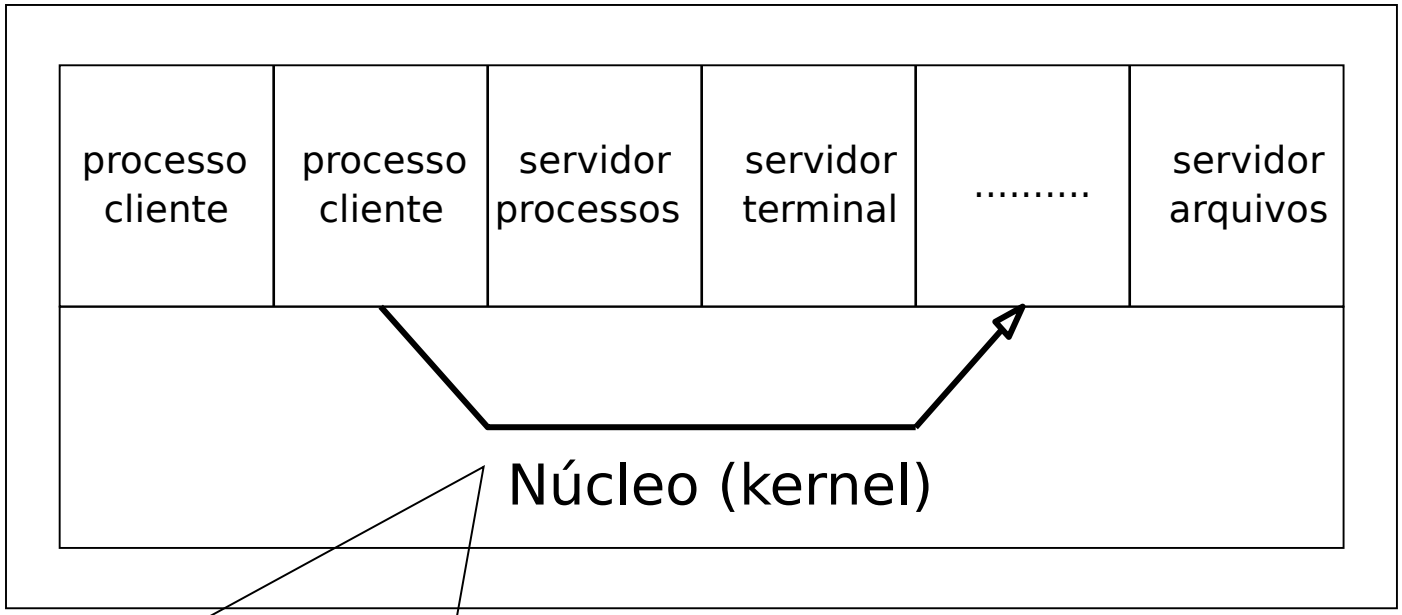
Estrutura dos Sistemas Operacionais – Cliente/Servidor



Estrutura dos Sistemas Operacionais – Cliente/Servidor



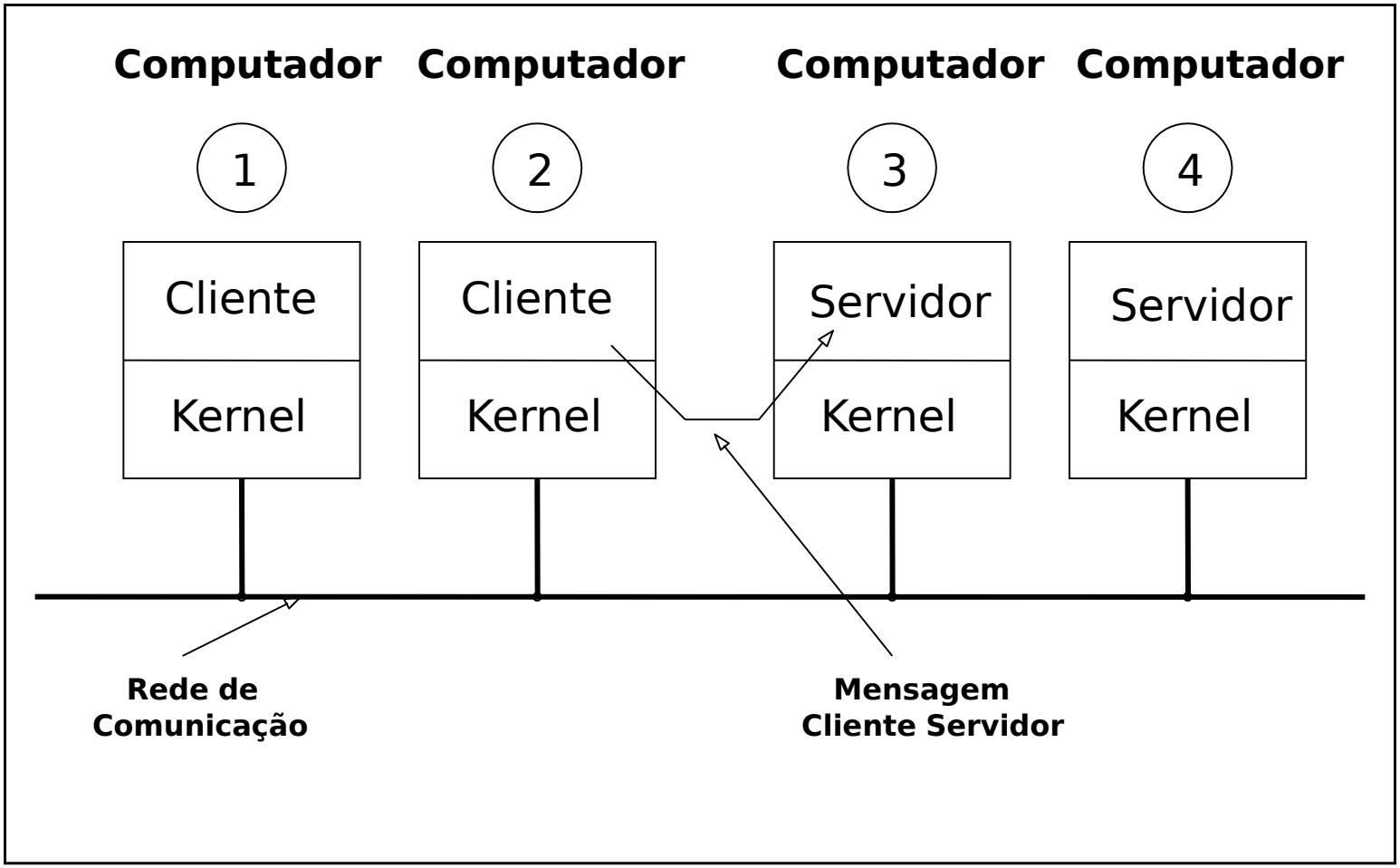
Estrutura dos Sistemas Operacionais – Cliente/Servidor



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;



Estrutura dos Sistemas Operacionais – Máquina Virtual

- Ideia surgiu 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;

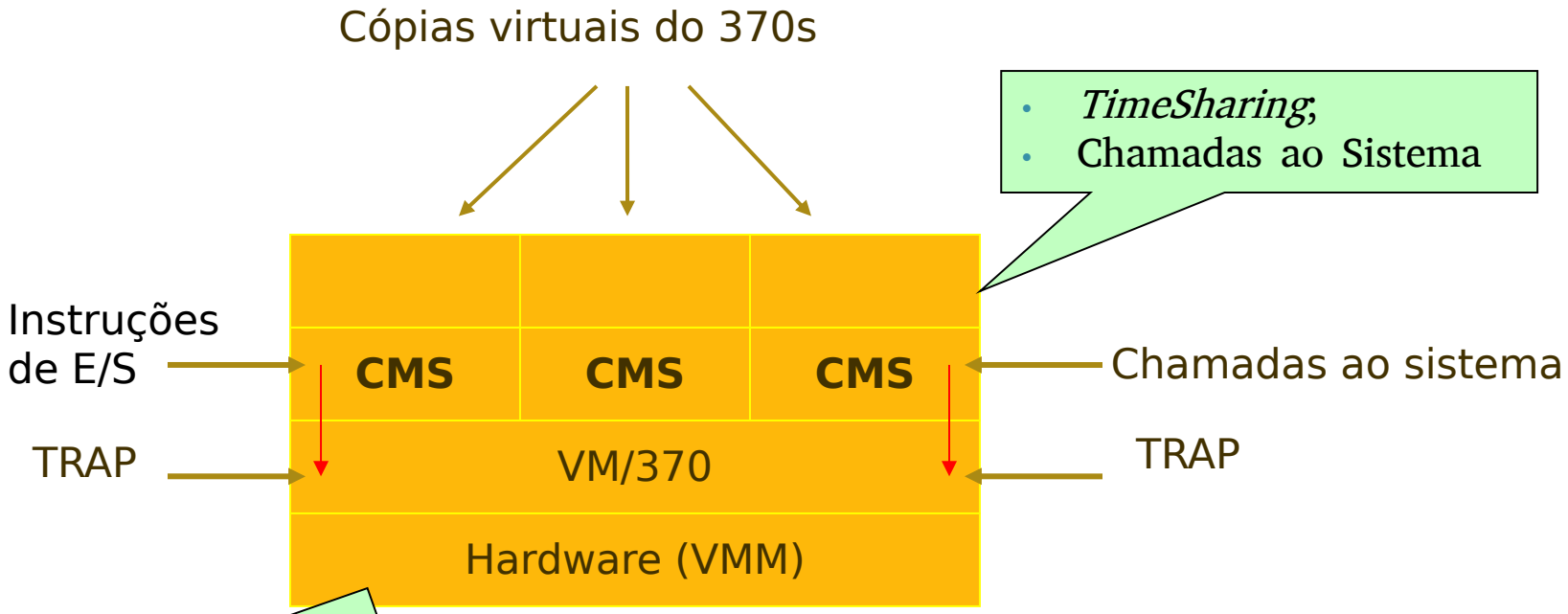
Estrutura dos Sistemas Operacionais – Máquina Virtual

- 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, posteriormente renomeado 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): executado sobre o *hardware* e implementa multiprogramação, provendo várias máquinas virtuais → é o coração do sistema;
    - Máquinas virtuais são cópias exatas do *hardware*, incluindo os modos *kernel* e usuário, E/S, interrupções e tudo mais;
    - Cada máquina virtual pode executar um Sistema Operacional diferente;
  - CMS (*Conversational Monitor System*):
    - Sistema operacional monousuário interativo;
    - *TimeSharing*;
    - Executa chamadas ao Sistema Operacional;

Estrutura dos Sistemas Operacionais – Máquina Virtual



Monitor da Máquina Virtual roda sobre o hardware e implementa multiprogramação

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;
    - Execução de um software em uma máquina virtual requer que a UCP seja virtualizada → problemas de executar instruções privilegiadas
    - Nos primeiros processadores Pentium, eram usados interpretadores de comandos para contornar a limitação, mas o desempenho era de 5 a 10 vezes menor