Universidade Federal
de Mato Grosso
Campus Rondonópolis

Sistemas Operacionais

Aula 2 – Chamadas de Sistema e Estruturas de SO

Prof. Msc. Cleyton Slaviero

cslaviero@gmail.com

Na última aula...

- Porque é necessário um sistema operacional ?
  - *Número e complexidade de recursos*
- O que é um sistema operacional?
  - *Lembrete: abstração e "arbitração"*
- Histórico
- Conceitos básicos
  - Processo
  - Memória
  - Chamadas de Sistema

Universidade Federal
de Mato Grosso
Campus Rondonópolis

Roteiro de hoje

- Por que é necessário um sistema operacional
- O que é um Sistema Operacional
- Histórico
- Conceitos Básicos
- Chamadas ao Sistema (System Calls)
- Estrutura de Sistemas Operacionais

Universidade Federal
de Mato Grosso
Campus Rondonópolis

Interfaces de um Sistema Operacional

- Usuário ⟺ SO:
  - **Shell ou Interpretador de comandos**

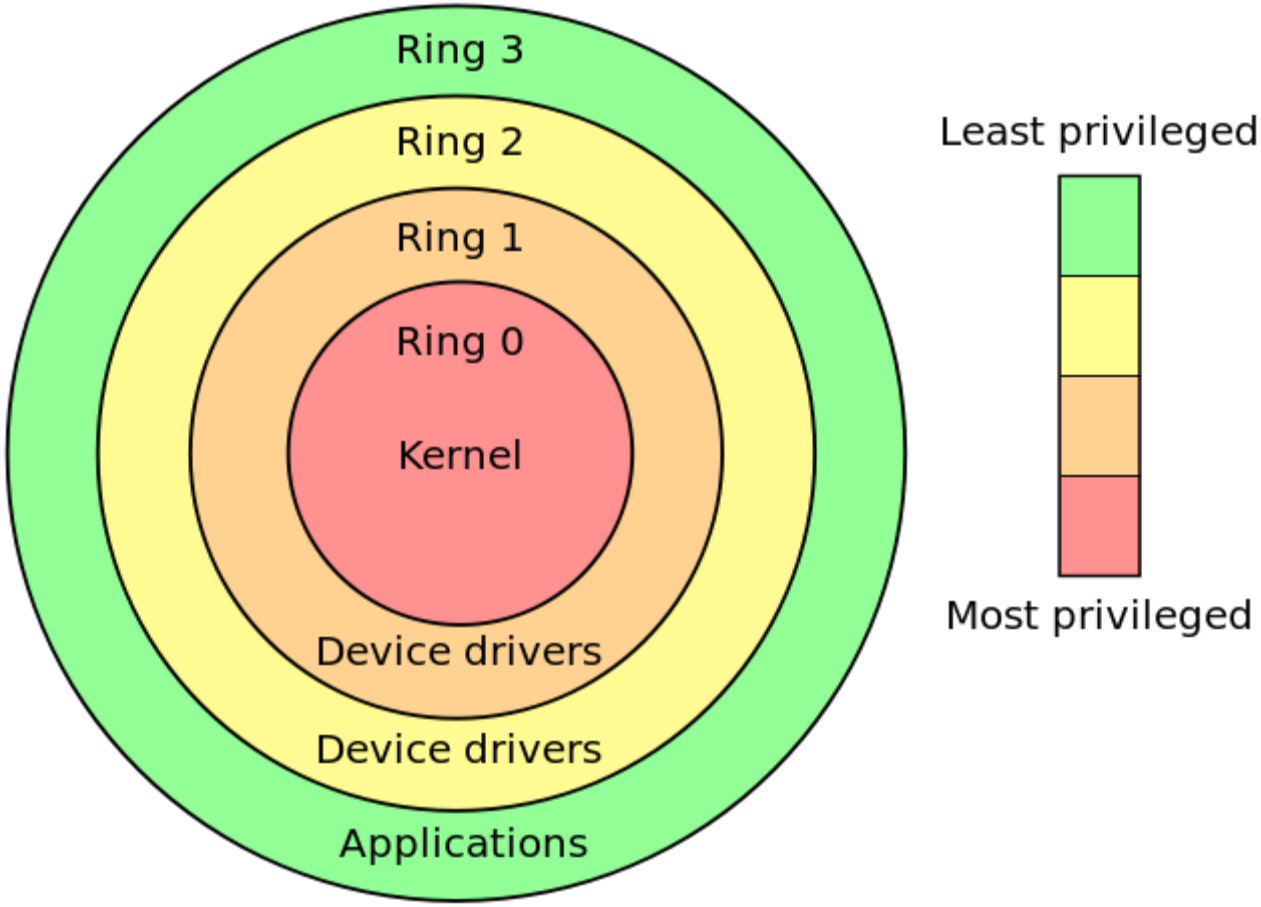
- Programas ⟺ SO:
  - **Chamadas ao Sistema**

Universidade Federal
de Mato Grosso
Campus Rondonópolis

Conceitos Básicos
Chamadas de 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;

Multiplos anéis de proteção do kernel – arquitetura x86



Conceitos Básicos
Chamadas de 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**;

Conceitos Básicos
Chamadas de 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 SO tem acesso às instruções privilegiadas;

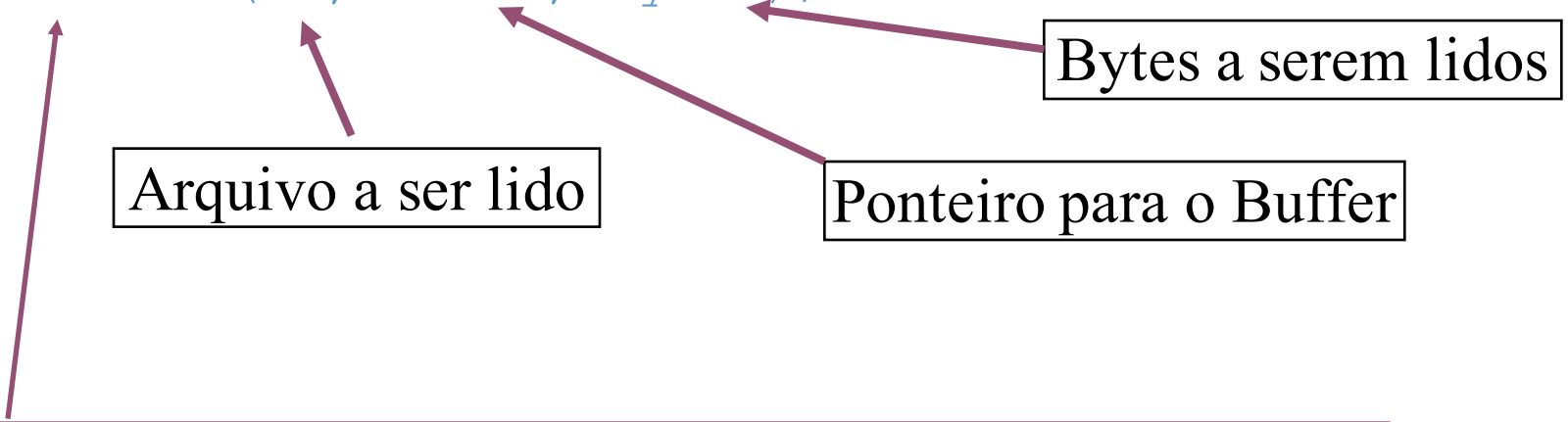
Conceitos Básicos
Chamadas de Sistema

- 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;
  - Apenas elas entram no kernel (e não chamadas de procedimento)

Conceitos Básicos
Chamadas de Sistema

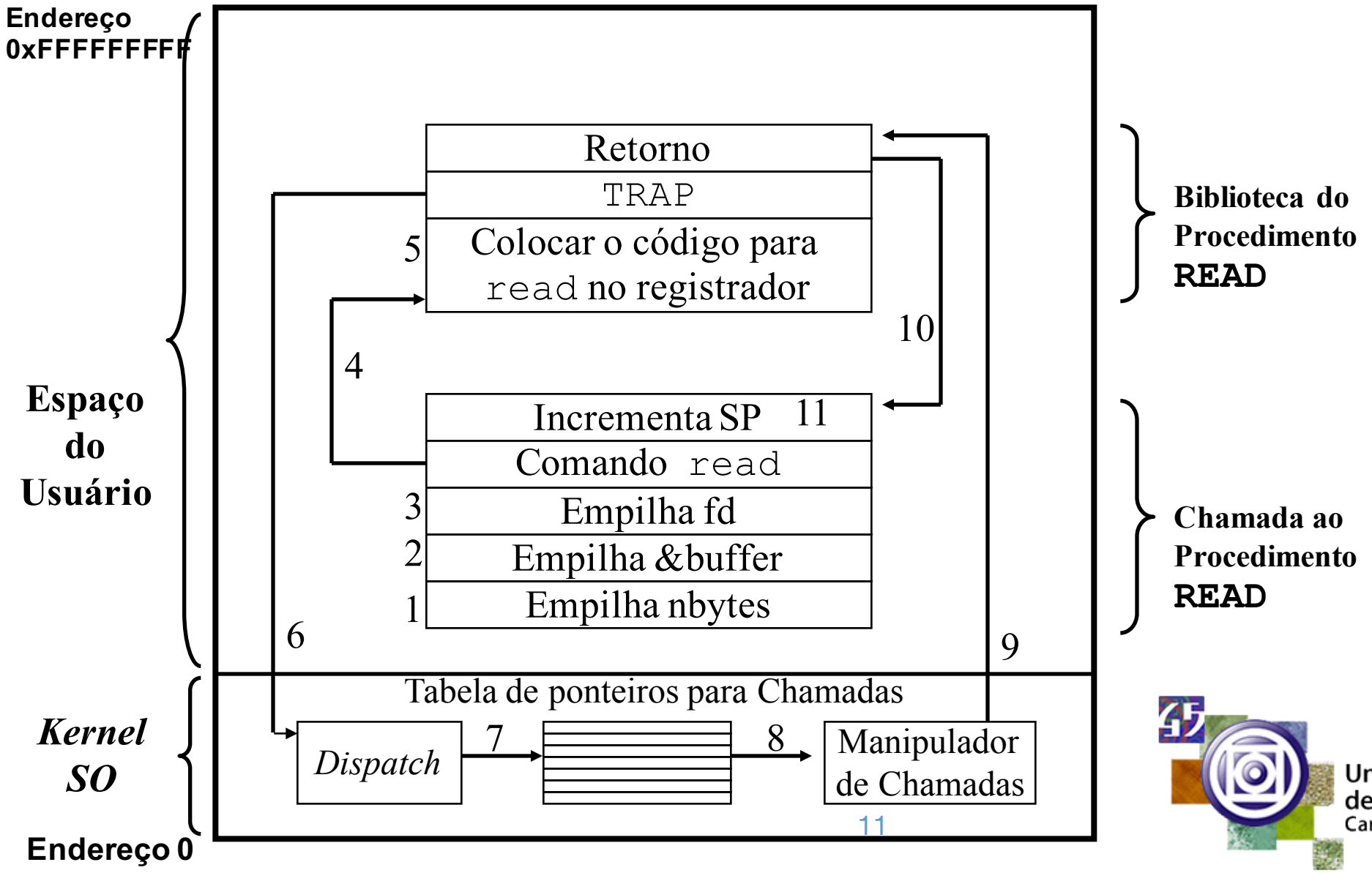
- **TRAP**: instrução que permite o acesso ao modo *kernel*;
- Exemplo:
  - Instrução do UNIX:

```
count = read(fd,buffer,nbytes);
```



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

Chamadas de Sistema



Universidade Federal de Mato Grosso
Campus Rondonópolis

Conceitos Básicos
Chamadas de Sistema

- Exemplos de chamadas da interface:
  - Chamadas para gerenciamento de **processos**:
    - `Fork (CreateProcess – WIN32)` – cria um processo;
    - Outros exemplos no POSIX (***P**ortable **O**perating **S**ystem **I**nterface*)

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

```
#define TRUE 1
while (TRUE) {   // repete pra sempre
      type prompt( ); // exibe prompt na tela
      read command(command, parameters);// lê input do terminal
      if (fork( ) != 0) { // faz o fork no processo
            waitpid(-1, &status, 0);// espera pela execução do filho
      } else {
            execve(command, parameters, 0); // executa comando
      }
}
```

Conceitos Básicos
Chamadas de Sistema

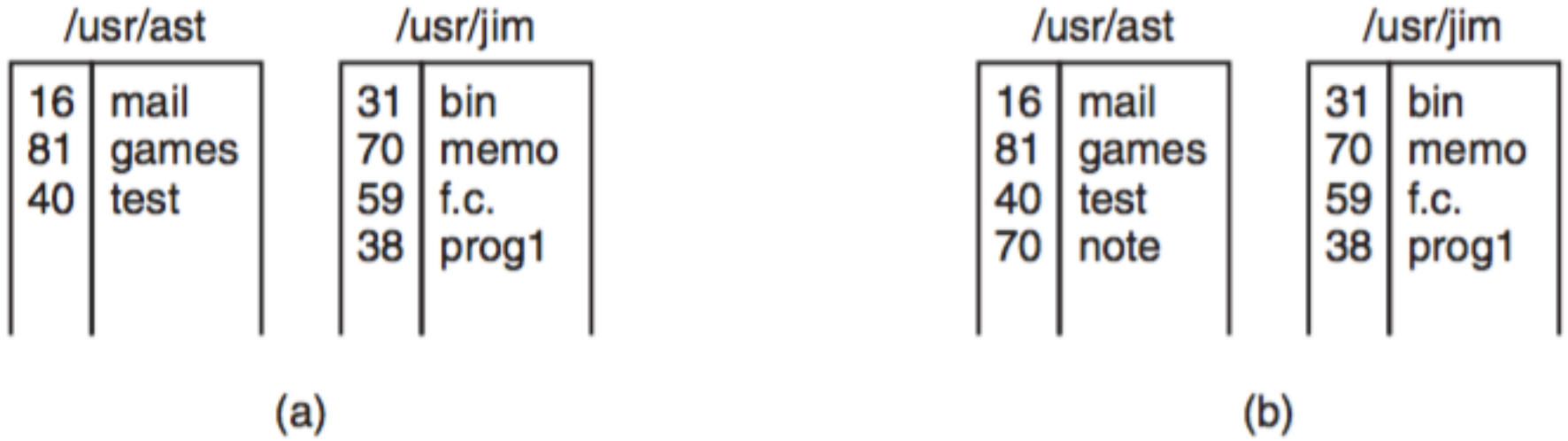
- Exemplos de chamadas da interface :
  - **Chamadas para gerenciamento de diretórios:**
    - `mount` – monta um diretório;
  - Chamadas para gerenciamento de arquivos:
    - `close (CloseHandle – WIN32)` – fechar um arquivo;
  - Outros exemplos no POSIX

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

Conceitos Básicos
Chamadas de Sistema

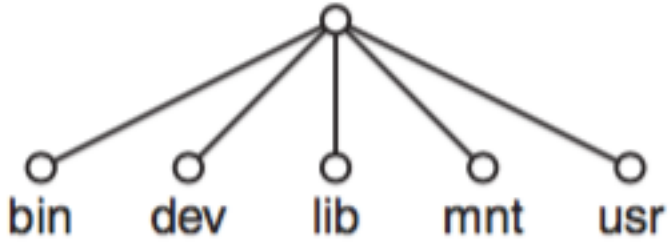
link("/usr/jim/memo", "/usr/ast/note");



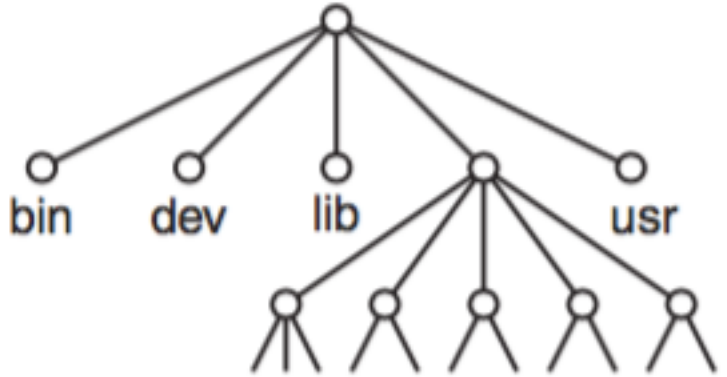
- i-nodes em cada pasta

Conceitos Básicos
Chamadas de Sistema

mount("/dev/sdb0", "/mnt", 0);



(a)



(b)

Conceitos Básicos
Chamadas de Sistema

- Exemplos de chamadas da interface :
  - Chamadas para gerenciamento de **diretórios**:
    - `mount` – monta um diretório;
  - **Chamadas para gerenciamento de arquivos:**
    - `close (CloseHandle – WIN32)` – fechar um arquivo;
  - Outros exemplos no POSIX

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

Conceitos Básicos
Chamadas de Sistema

- Para ler/escrever um arquivo:
  - Abrir o arquivo
    - Nome, caminho, modo de leitura
  - Retorna um descritor de arquivo (int)
- Cada arquivo tem um ponteiro para a posição atual
- UNIX mantém registrado o modo do arquivo, tamanho, ultima modificação, etc.
  - `Stat` para pedir informações

Conceitos Básicos
Chamadas de Sistema

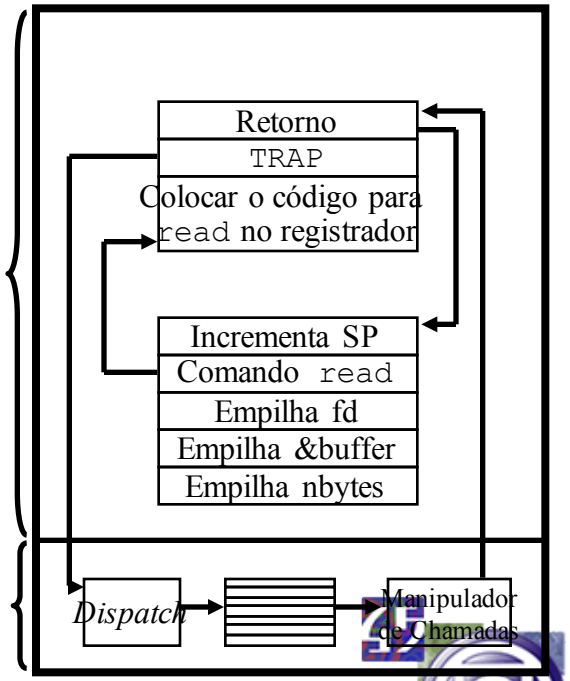
- Exemplos de chamadas da interface :
  - Outros tipos de chamadas:
    - `chmod`: modifica permissões;
  - Outros exemplos no POSIX

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

Conceitos Básicos
Chamadas de Sistema

- Chamadas da interface. UNIX vs. Windows:
  - **UNIX**
    - Chamadas da interface muito semelhantes às chamadas ao sistema
    - ~100 chamadas a procedimentos
  - **Windows**
    - Chamadas da interface totalmente diferente das chamadas ao sistema
    - APIWin32 (*Application Program Interface*)
      - Padrão de acesso ao sistema operacional
      - Facilita a compatibilidade
      - Possui **milhares** de procedimentos



Conceitos Básicos
Chamadas de Sistema

- Exemplos de chamadas da interface: Unix e API 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

Roteiro

- Por que é necessário um sistema operacional
- O que é um Sistema Operacional
- Histórico
- Conceitos Básicos
  - Processo;
  - Memória;
  - Chamadas de Sistema;
- Chamadas ao Sistema
- **Estrutura de Sistemas Operacionais**

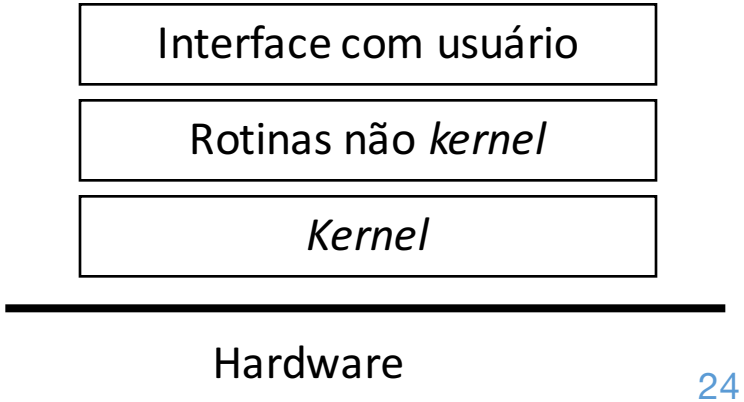
Estrutura dos Sistemas Operacionais

- Pode atuar de duas maneiras diferentes:
  - Como máquina estendida
    - Chamadas ao sistema - interface
    - Parte externa
  - Como gerenciador de recursos

    Parte interna

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*

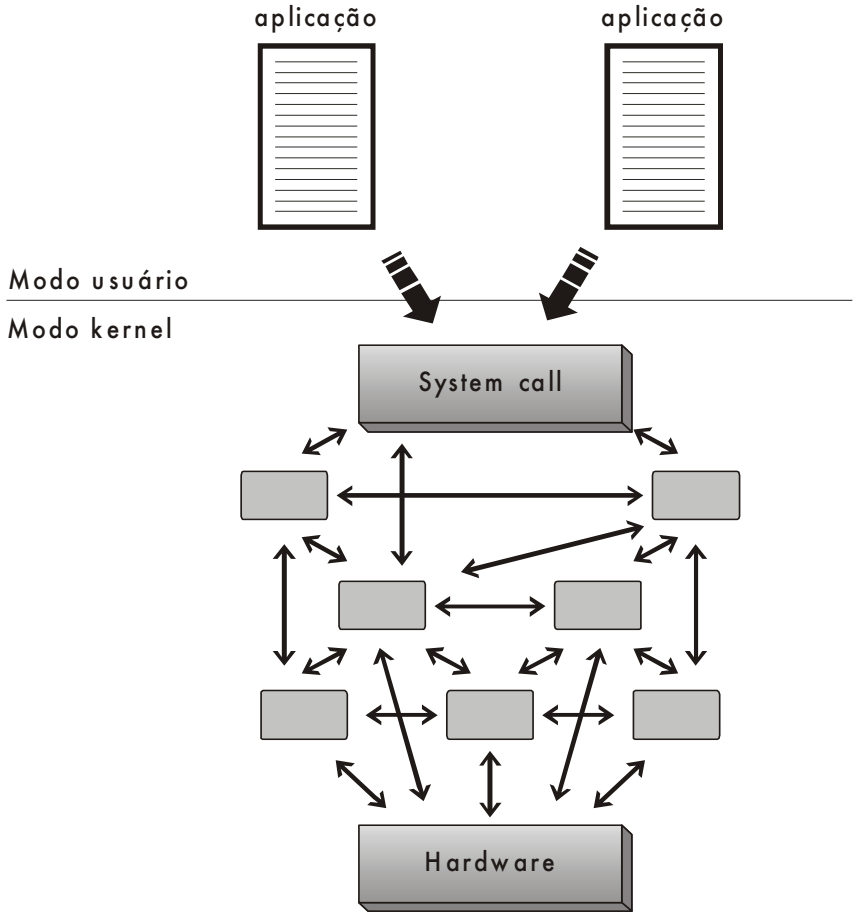


Estrutura dos Sistemas Operacionais

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

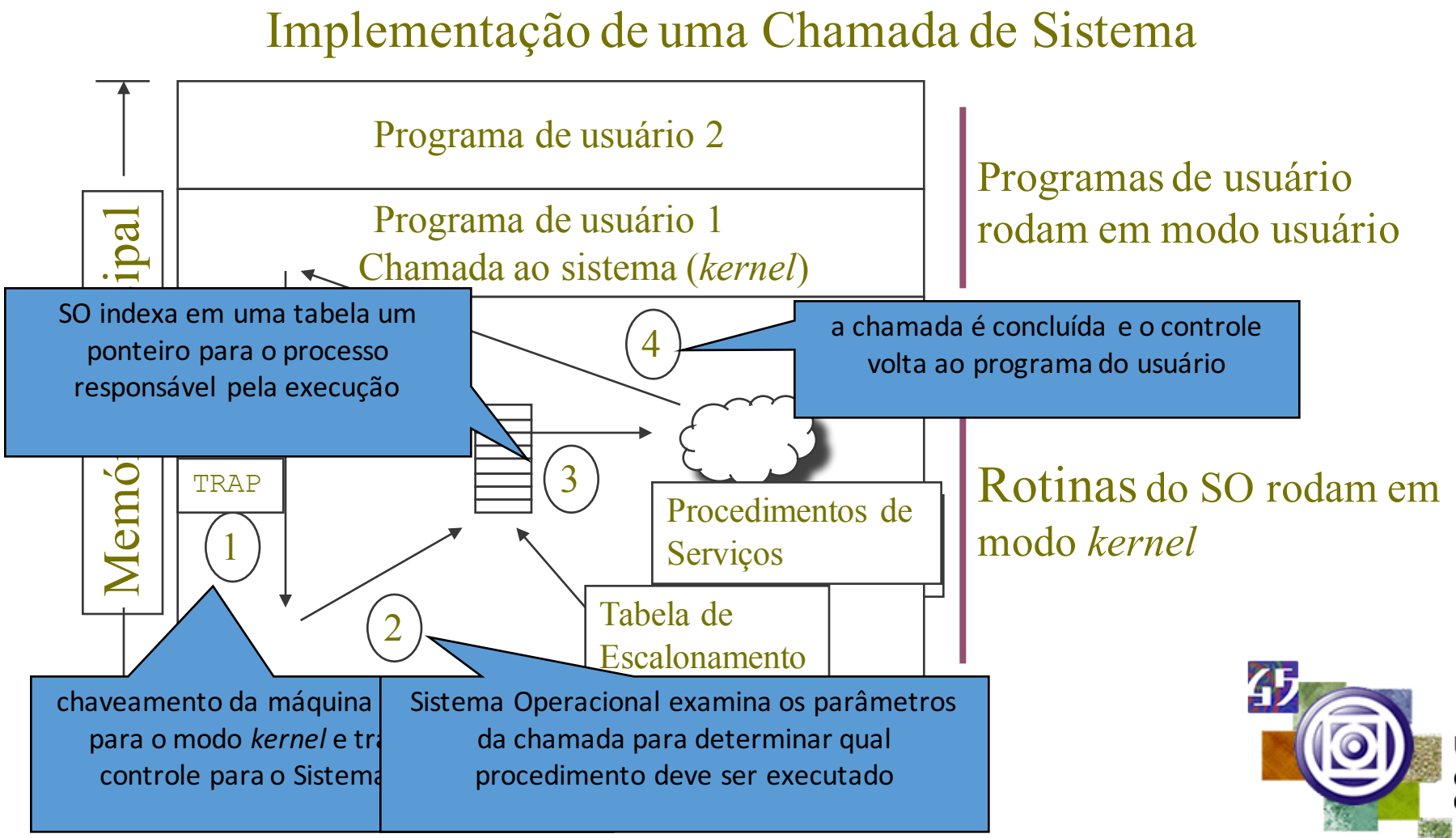
Estrutura dos Sistemas Operacionais - Monolítico

- Um programa único em modo núcleo
- Todos os **módulos** do sistema são **compilados individualmente** e depois ligados uns aos outros em um único **arquivo-objeto**
  - Vantagem: todos se chamam
  - Desvantagem: difícil de entender
- Uma falha é fatal!
- É possível ter alguma estrutura



Universidade Federal de Mato Grosso
Campus Rondonópolis

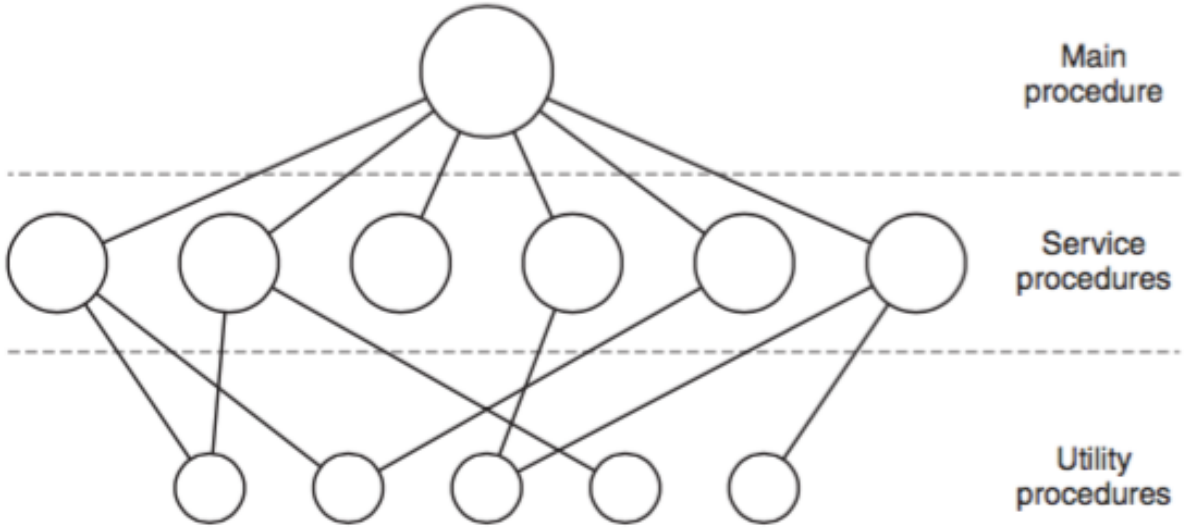
Estrutura dos Sistemas Operacionais - Monolítico



Universidade Federal de Mato Grosso
Campus Rondonópolis

Estrutura dos Sistemas Operacionais - Monolítico

- Uma estrutura básica
  - Programa principal
  - Procedimentos de serviço que executam as chamadas de sistema
  - Procedimentos utilitários para auxiliar procedimentos de serviço



Estrutura dos Sistemas Operacionais – Monolítico

- É possível carregar extensões
  - Shared libraries (UNIX) /DLLs (Win)
- Primeiros sistemas UNIX e MS-DOS;

Estrutura dos Sistemas Operacionais – Em camadas

- Possui uma hierarquia de níveis;
- Primeiro sistema em camadas: `THE` (idealizado por E.W. Dijkstra em 65-68);
  - 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

Camadas definidas no THE

Fornecimento de Serviços

Camada	Função
5	O operador
4	Programas do usuário
3	Gerenciamento de entrada/saída
2	Comunicação operador-processo
1	Gerenciamento da memória e do tambor magnético
0	Alocação de processador e multiprogramação

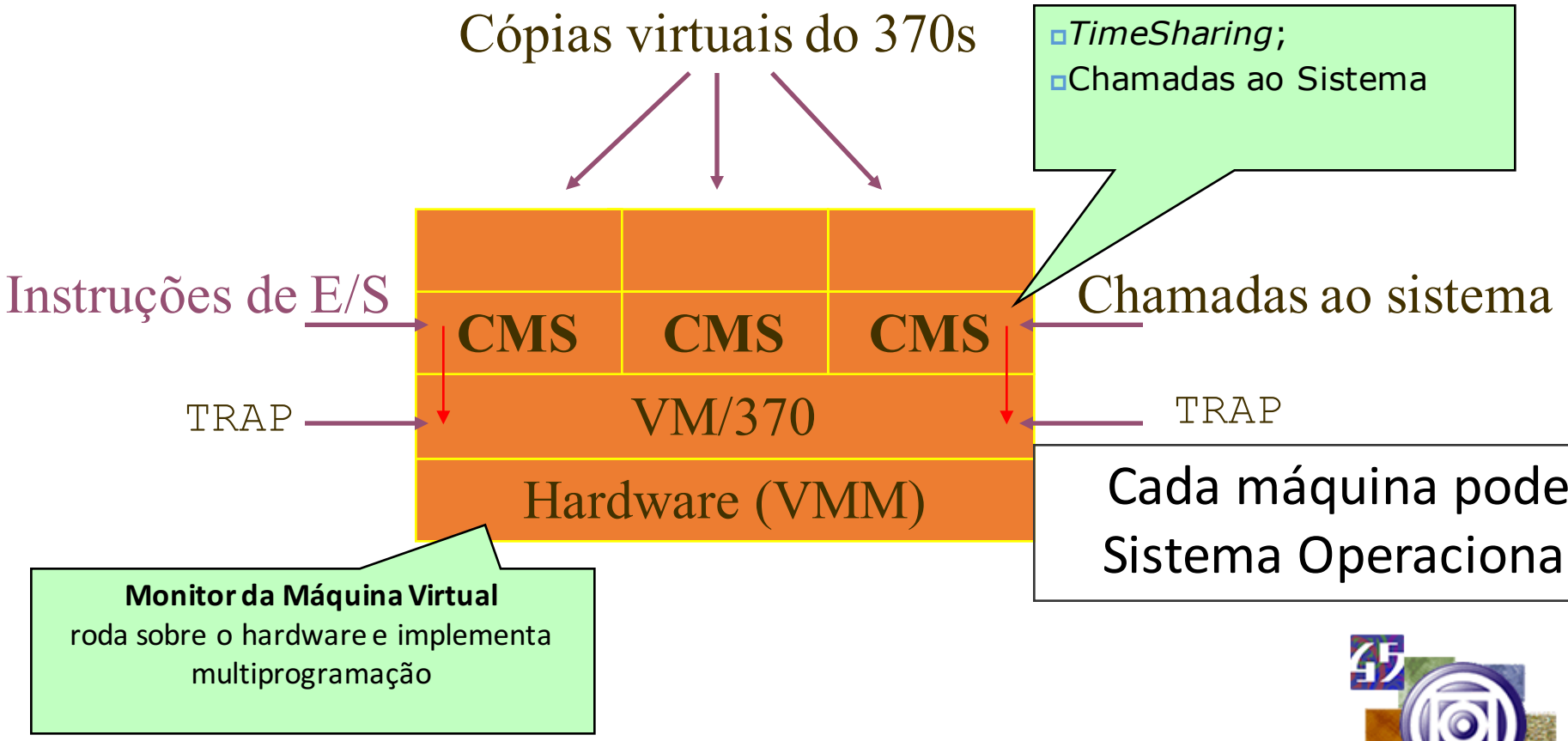
Estrutura dos Sistemas Operacionais – Máquina Virtual

- Ideia 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.;

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

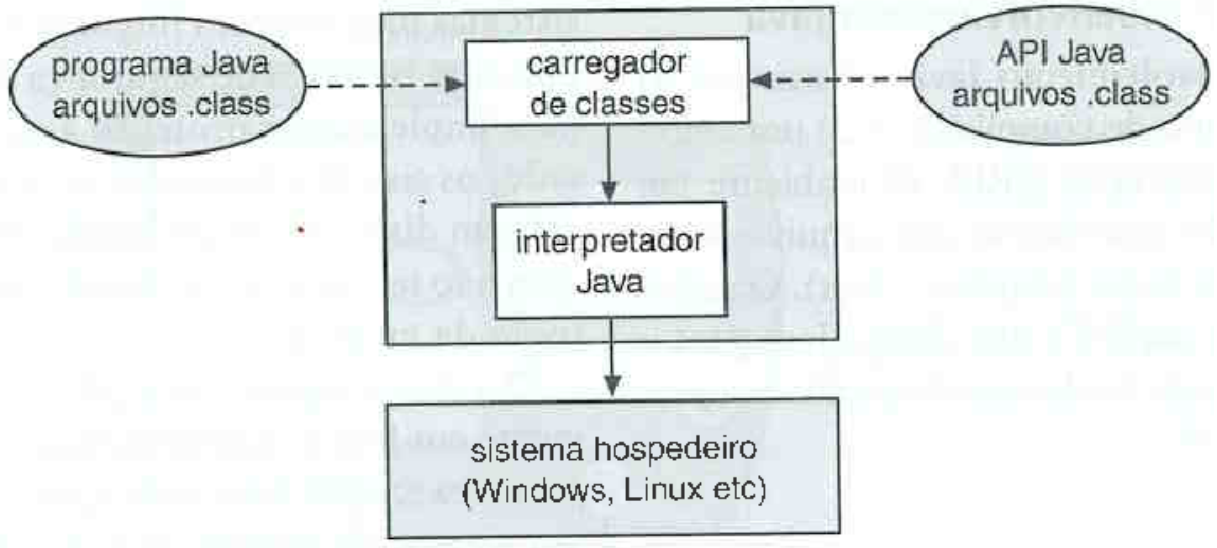


Estrutura dos Sistemas Operacionais – Máquina Virtual

- A ideia de máquina virtual foi posteriormente utilizada em contextos diferentes:
  - Programas MS-DOS: rodam em computadores 32bits;
    - As chamadas feitas pelo MS-DOS ao Sistema Operacional eram realizadas e monitoradas pelo monitor da máquina virtual (VMM);

Estrutura dos Sistemas Operacionais – Máquina Virtual

- A ideia de máquina virtual foi posteriormente utilizada em contextos diferentes:
  - 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;



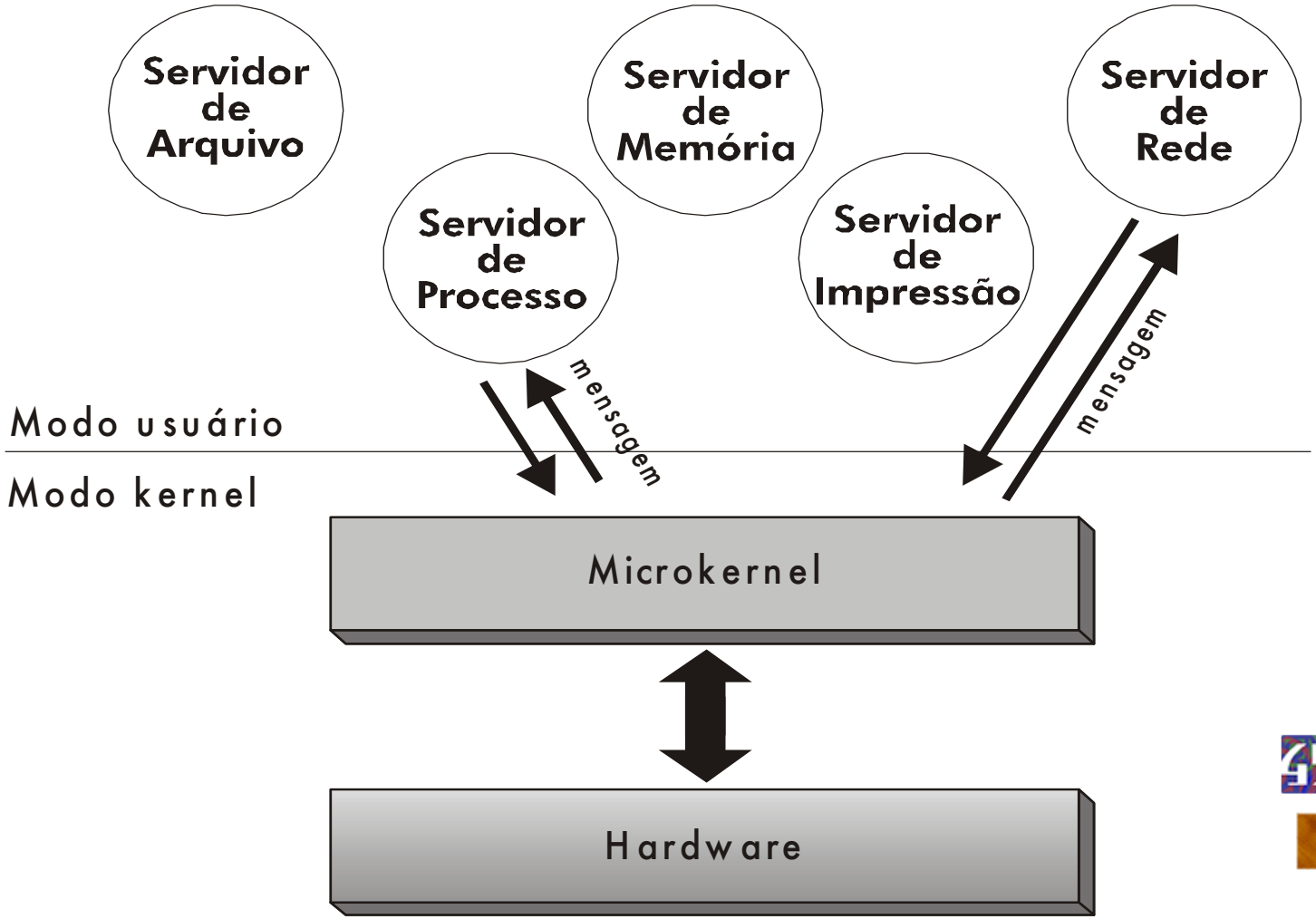
Estrutura dos Sistemas Operacionais – Máquina Virtual

- A ideia de máquina virtual foi posteriormente utilizada em contextos diferentes:
  - Computação em nuvem
    - Virtualização dos servidores simula diferentes ambientes em servidores físicos

Estrutura dos Sistemas Operacionais – Máquina Virtual

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

Estrutura dos Sistemas Operacionais – *Micro-Kernel*

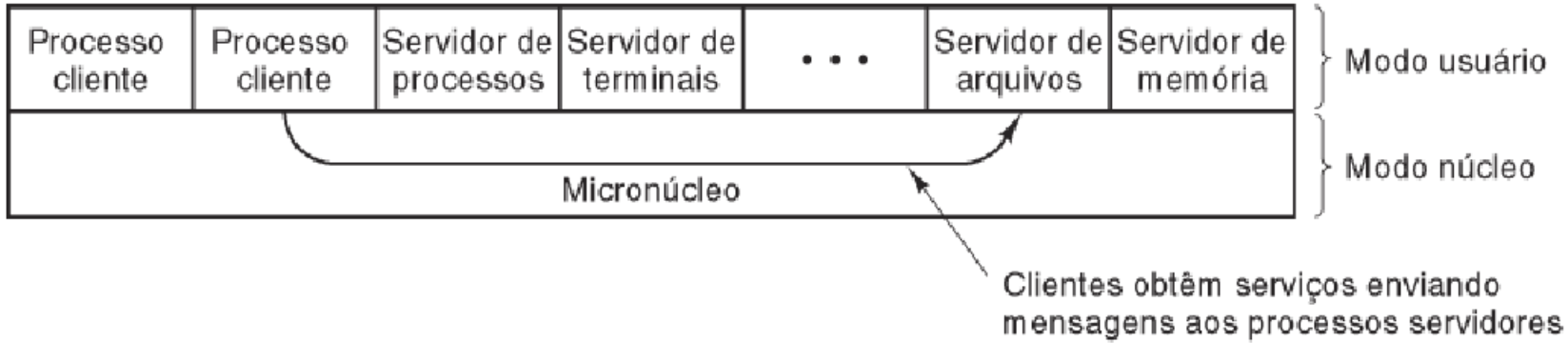


Estrutura dos Sistemas Operacionais – Cliente/Servidor

- 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

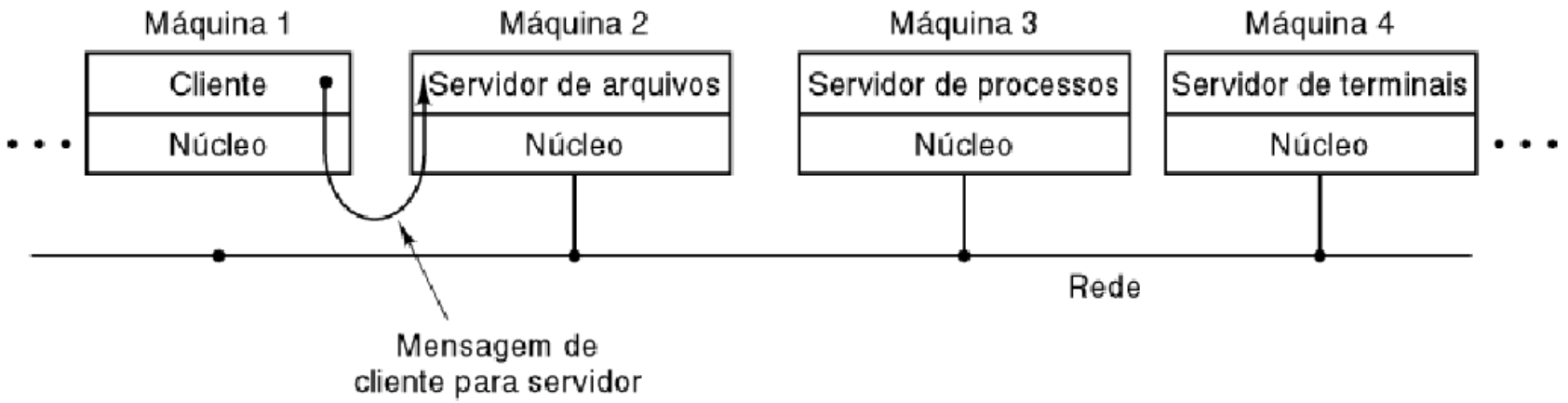
Cada processo servidor trata de uma tarefa



- Os processos servidores não têm acesso direto ao hardware. Assim, se algum problema ocorrer com algum desses servidores, o hardware não é afetado;
- 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;



Universidade Federal de Mato Grosso
Campus Rondonópolis

Estrutura dos Sistemas Operacionais – Cliente/Servidor

- Linux
  - Monolítico +Módulos
- Windows
  - Microkernel + Camadas + Módulos