BC1518 - Sistemas Operacionais

Aula 3: Estrutura de Sistemas Operacionais

Tópicos desta aula

- Breve revisão
- Componentes do Sistema Operacional
- Serviços de Sistemas Operacionais
- Chamadas de Sistema (*System Calls*)
- Estrutura do Sistema Operacional
- Máquinas Virtuais

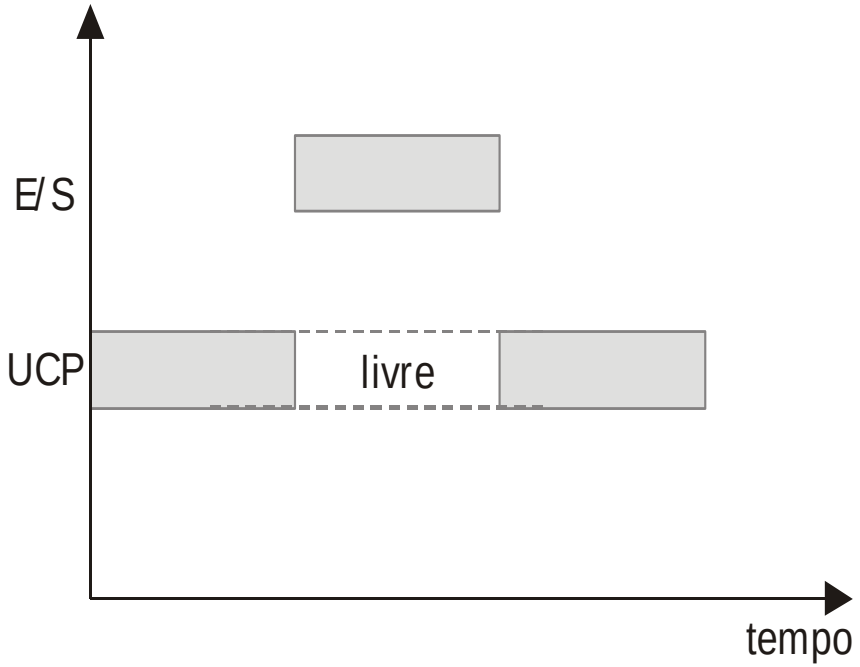
Sistemas monoprogramáveis x multiprogramáveis

- Em sistemas **monoprogramáveis**, apenas um programa pode estar em execução de cada vez
  - Os recursos como processador, memória, dispositivos de E/S ficam dedicados à execução de uma única tarefa
  - Enquanto o programa aguarda por algum evento (leitura de um dado do disco) o processador fica ocioso sem realizar nenhum processamento
  - A memória é subutilizada, caso o programa não ocupe todo o espaço disponível
  - A implementação é simples, não requerendo o suporte ao compartilhamento de recursos como memória, processador e dispositivos de E/S

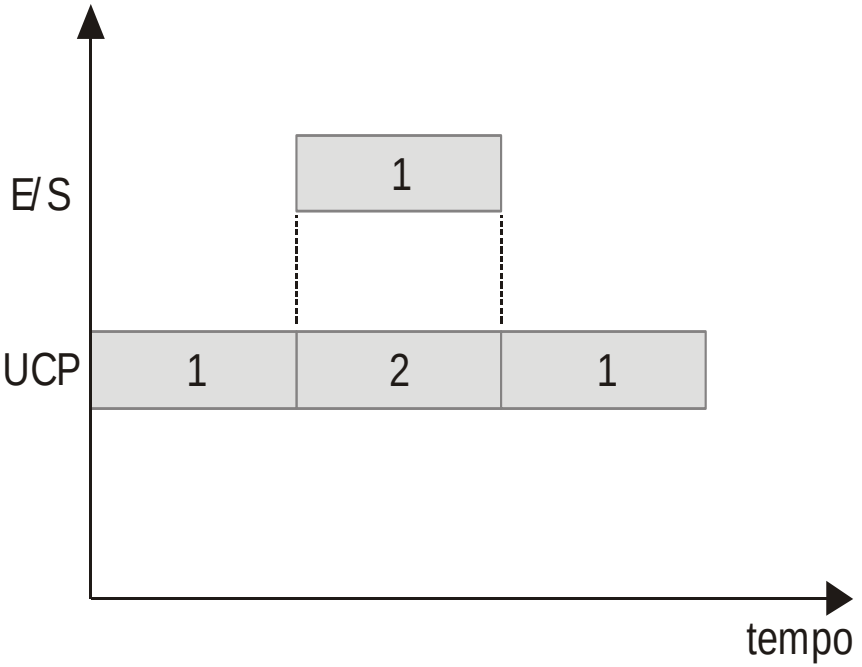
Sistemas monoprogramáveis x multiprogramáveis

- Em sistemas **multiprogramáveis**, vários programas podem estar em execução ao mesmo tempo
  - Os recursos computacionais são compartilhados entre vários usuários e aplicações
  - Diversas tarefas são mantidas na memória principal ao mesmo tempo, e a CPU é multiplexada entre elas
- Enquanto um programa aguarda por uma operação de E/S, o processador pode executar algum outro programa no mesmo intervalo de tempo
  - Proporciona um aumento na utilização do processador (organizado de modo que sempre tenha uma tarefa a ser executada) e melhor utilização dos outros recursos também
  - Além da redução do tempo médio de processamento (em contrapartida ao processamento sequencial) e possibilita a redução de custos com o compartilhamento de recursos
  - Implementação mais complexa: é preciso lidar com a concorrência no acesso aos recursos

Sistemas monoprogramáveis x multiprogramáveis



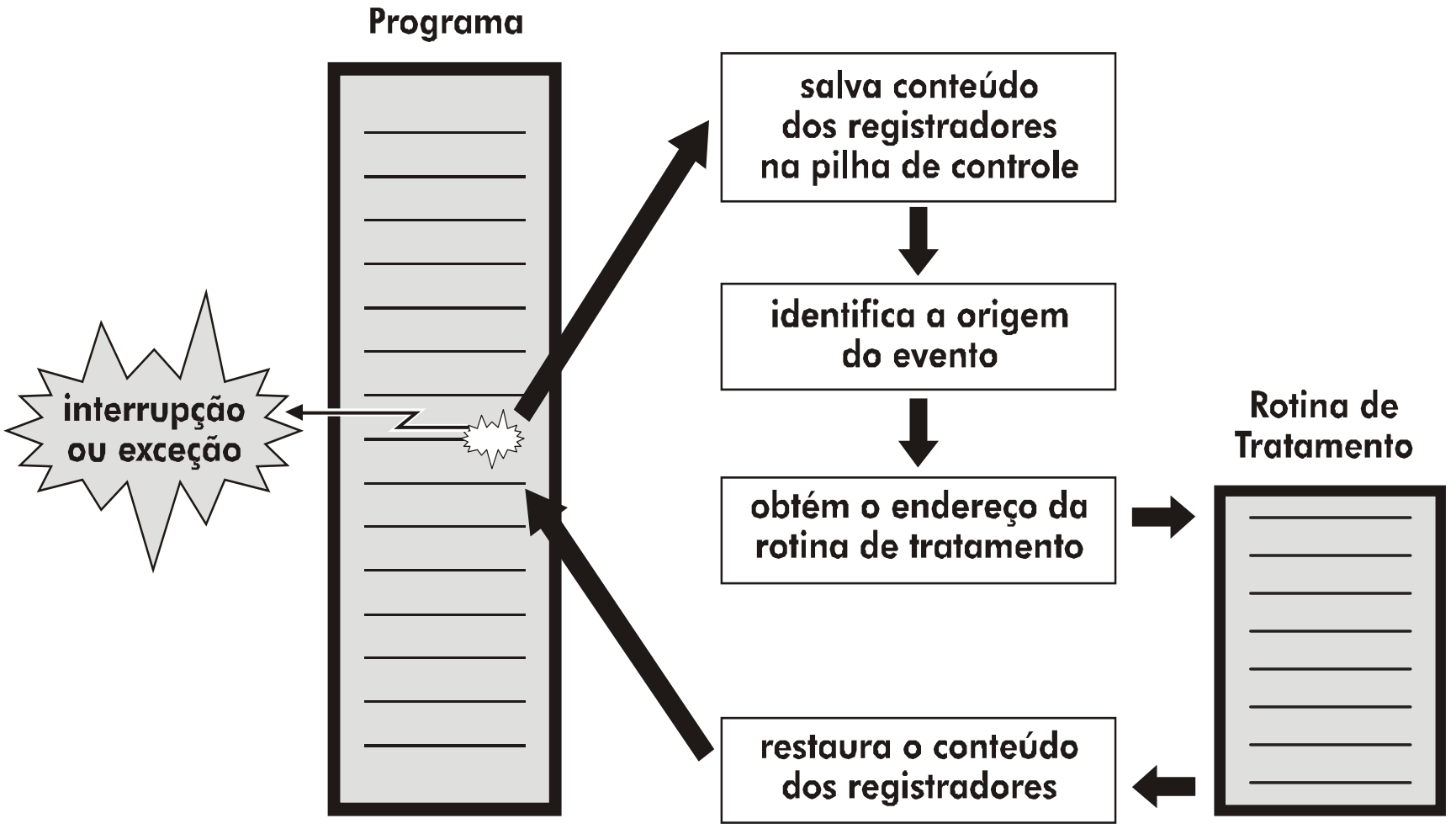
(a) Sistema Monoprogramável



(b) Sistema Multiprogramável

[Machado]

Mecanismo de interrupção



Mecanismo de interrupção/exceção [Machado]

Componentes comuns do SO

- Gerência de Processos
- Gerência da Memória Principal
- Gerência de Arquivos
- Gerência do Sistema de E/S
- Gerência de Armazenamento Secundário
- Sistema de Proteção
- Redes (Sistemas Distribuídos)

Gerência de Processos

- Um ***processo*** é um **programa em execução**
  - É a unidade de trabalho em um sistema
  - Um sistema é uma coleção de processos (processos do sistema operacional e processos do usuário)
  - Um programa por si só não é um processo (entidade *passiva*)
  - Um processo é uma entidade *ativa*, tem associado a ele várias informações: contador de programa, ponteiro para pilha, etc. (é como um contâiner que armazena todas as informações necessárias para a execução do mesmo)
  - Ao iniciar a execução de um programa, é criado um processo correspondente (por exemplo, ao abrir o navegador web ou editor de texto)
  - Um processo precisa de certos recursos, incluindo tempo de CPU, memória, arquivos e dispositivos de E/S para realizar sua tarefa. Esses recursos são atribuídos ao processo quando ele é criado ou são alocados enquanto ele está sendo executado

Gerência de Processos

➢Havendo vários processos na memória, é preciso definir qual deles será escolhido para ser executado quando a CPU estiver livre

➢Quando um processo é suspenso temporariamente, ele deverá ser reiniciado mais tarde a partir do ponto onde havia interrompido

➢O **Sistema Operacional** é responsável pelas seguintes atividades em conjunto com o gerenciamento de processos:

- Criar e excluir processos
- Suspender e retomar processos
- Oferecer mecanismos para:
  - sincronismo de processos
  - comunicação entre processos
  - tratamento de *deadlock*

Gerência de Memória Principal

- ***Memória*** é um grande vetor de palavras ou *bytes*, cada um com seu próprio endereço
  - É um repositório de dados rapidamente acessíveis, compartilhados pela CPU e pelos dispositivos de E/S
- A ***memória principal*** é um dispositivo de armazenamento volátil; ela perde seu conteúdo no caso de interrupção de energia
- O objetivo é utilizar de maneira eficiente o espaço da memória principal, alocando a cada programa uma área de memória independente - isso evita que programas com erros ou maliciosos interfiram na execução dos demais
- O ***Sistema Operacional*** é responsável pelas seguintes atividades em conjunto com o gerenciamento de memória:
  - Controlar que partes da memória estão sendo usadas atualmente e por quem
  - Decidir quais processos devem ser carregados quando o espaço de memória se tornar disponível

Gerência de Arquivos

➢Um ***arquivo*** é uma **coleção de informações relacionadas**, definidas por seu criador

- Normalmente, os arquivos representam programas (nos formatos fonte e objeto) e dados

➢O **Sistema Operacional** é responsável pelas seguintes atividades relacionadas ao gerenciamento de arquivos:

- Criação e exclusão de **arquivos**
- Criação e exclusão de **diretórios** (ou **pastas**)
- Suporte a primitivas para **manipulação de arquivos e diretórios**
- Mapeamento de arquivos em ***armazenamento secundário***
- Backup de arquivos em ***meios de armazenamento estáveis*** (não voláteis)

Gerência de Sistema de E/S

➢Um dos objetivos de um sistema operacional é ocultar as peculiaridades de dispositivos de hardware específicos

➢E oferecer maior simplicidade para as operações de E/S para os usuários e aplicações (subsistema de E/S)

➢A interação com cada dispositivo é implementada por meio de drivers (que recebem comandos para acesso ao dispositivo e os traduz em comandos específicos que serão executados pelo controlador de dispositivo

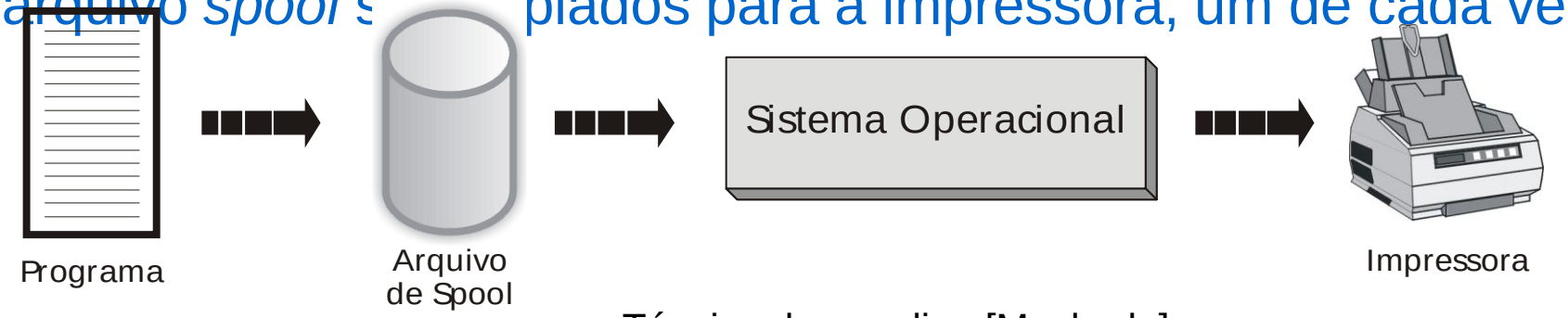
➢O subsistema de E/S consiste em:

- Um componente de gerência de memória que inclui ***buffering***, armazenamento em ***cache*** e ***spooling***
- Uma interface geral de *driver* de dispositivo
- *Drivers* para dispositivos de *hardware* específicos

• Apenas o *driver* do dispositivo conhece as peculiaridades do dispositivo específico ao qual foi atribuído

Spooling

- A técnica *spooling* (*Simultaneous Peripheral Operations On-line*) utiliza um *buffer* para controlar a saída concorrente para um dispositivo
- Está presente na maioria dos SOs e é utilizada para o gerenciamento de impressão
  - A impressora pode imprimir um arquivo de cada vez
  - Várias aplicações podem querer imprimir suas saídas ao mesmo tempo (sem que elas fiquem intercaladas)
  - O SO intercepta toda saída para a impressora, armazenando-a em arquivo de disco (arquivo de *spool*)
  - Após o término de uma impressão, os arquivos de saída no arquivo *spool* são copiados para a impressora, um de cada vez



Técnica de spooling [Machado]

Gerência de Armazenamento Secundário

➢Como a memória principal (***armazenamento primário***) é volátil e muito pequena para acomodar todos os dados e programas permanentemente, o sistema de computador precisa fornecer ***armazenamento secundário*** para apoiar a memória principal

➢A maioria dos sistemas de computador modernos utiliza discos como o principal meio de armazenamento para programas e dados

- Compiladores, processadores de texto, etc. são armazenados em disco e quando são utilizados são carregados na memória
- Utilizam o disco como origem e destino de seu processamento
-

➢O **Sistema Operacional** é responsável pelas seguintes atividades relacionadas ao **gerenciamento de disco**:

- Gerenciamento do espaço livre
- Alocação do armazenamento
- Escalonamento do disco

Sistema de Proteção

➢***Proteção*** se refere a qualquer **mecanismo para controlar o acesso dos programas**, processos ou usuários **aos recursos do sistema de computação**

➢Os mecanismos de proteção provêem acesso controlado limitando os tipos de acesso permitidos aos usuários

➢Além disso, a proteção garante que somente os processos com autorização apropriada possam acessar memória, CPU, dentre outros recursos

➢O mecanismo de proteção precisa:

- distinguir entre **uso autorizado e não autorizado**
- especificar os **controles a serem impostos**
- fornecer um **meio para a imposição**

➢É tarefa da **segurança** defender um sistema contra ataques externos e internos (quebra de confidencialidade, integridade,

Sistemas Distribuídos / Redes

➢Um ***sistema distribuído*** é um conjunto de processadores que não compartilham memória ou um relógio

- cada processador possui sua própria memória local

➢Os processadores são conectados por uma ***rede de comunicação***

➢A comunicação ocorre através do uso de um ***protocolo*** (por ex., TCP/IP, ATM etc.)

➢Um protocolo de rede requer um dispositivo de interface (**adaptador de rede** e um *driver* para gerenciá-lo) e um software para tratar os dados

➢Um Sistema Operacional de Rede possibilita o compartilhamento de arquivos pela rede e inclui um esquema de comunicação que permite que diferentes processos em diferentes máquinas se comuniquem (troquem mensagens)

Serviços de Sistemas Operacionais

O SO oferece um ambiente para a execução de programas

Fornece um conjunto de serviços com funções úteis para o usuário:

- **Interface do usuário**
  - Interface de linha de comando utiliza comandos em modo texto
  - Interface gráfica (*Graphical User Interface - GUI*)

- **Execução de programas:** capacidade do sistema para carregar um programa na memória e executá-lo

- **Operações de E/S:** como os programas do usuário não podem executar operações de E/S diretamente, o Sistema Operacional precisa prover meios para realizar essas operações

Serviços de Sistemas Operacionais (cont.)

- **Manipulação do sistema de arquivos:** oferecer meios para ler, gravar, criar, excluir arquivos e gerenciar permissões de acesso a arquivos e diretórios

- **Comunicações:** troca de informações entre processos sendo executados no mesmo computador ou em sistemas diferentes ligados por uma rede. Implementadas através de ***memória compartilhada*** ou pela ***passagem de mensagens***

- **Detecção de erro:** assegura computação correta detectando erros no *hardware* da CPU e da memória, nos dispositivos de E/S ou nos programas do usuário (estouro aritmético, tentativa de acessar um local de memória ilegal, etc.)

Funções adicionais do SO

As funções adicionais existem, não para auxiliar o usuário, mas para garantir operações eficientes do sistema

- **Alocação de recursos:** os recursos são alocados a diversos usuários ou a múltiplas tarefas sendo executadas ao mesmo tempo

- **Contabilidade:** controla e registra que usuários usam, quanto, e que tipo de recursos do computador para contabilizar cobrança ou para acumular estatísticas de uso (estatísticas podem ser utilizadas para a melhoria dos serviços do sistema)

- **Proteção:** assegura que todo o acesso aos recursos do sistema seja controlado. A **segurança** do sistema contra acesso por pessoas estranhas (influências externas) também é importante

Interpretador de Comandos

- Uma das maneiras de os usuários realizarem a interface com o Sistema Operacional é através da interface de linha de comando (terminal)
  - Usuários entram diretamente com comandos para serem executados pelo SO
- O programa que lê e interpreta instruções de controle é chamado alternadamente de:
  - ***Interpretador de linha de comando***
  - Interpretadores são conhecidos como ***shell*** *(Bourne shell, C shell, Korn shell*, nos sistemas UNIX e Linux)
  - A função principal do **interpretador de comandos** é buscar e executar a próxima instrução de comando especificada pelo usuário
- Os comandos podem ser implementados de duas formas:
  - No próprio código do interpretador de comandos (cada comando possui uma seção de código que define os parâmetros e faz a chamada de sistema apropriada)
  - Tamanho do arquivo é proporcional à quantidade de comandos

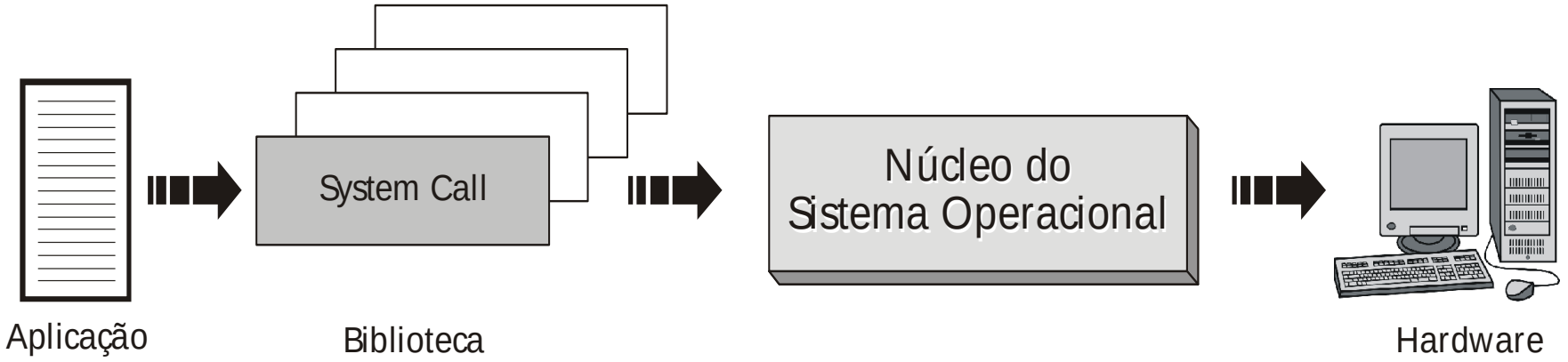
Interpretador de Comandos (cont.)

- A segunda forma, empregada pelo UNIX dentre outros SOs, implementa os comandos por meio de **programas do sistema**, i.e., ele usa o comando para identificar um arquivo a ser carregado na memória e executado
- Por exemplo, o comando **`rm arquivo.txt`** procura um arquivo chamado **`rm`**, carrega o arquivo na memória e o executa com o parâmetro **`arquivo.txt`**
  - O programa interpretador de comandos é pequeno e não requer alterações na inclusão de novos comandos (basta acrescentar novos arquivos)

Chamadas de Sistema (*System Calls*)

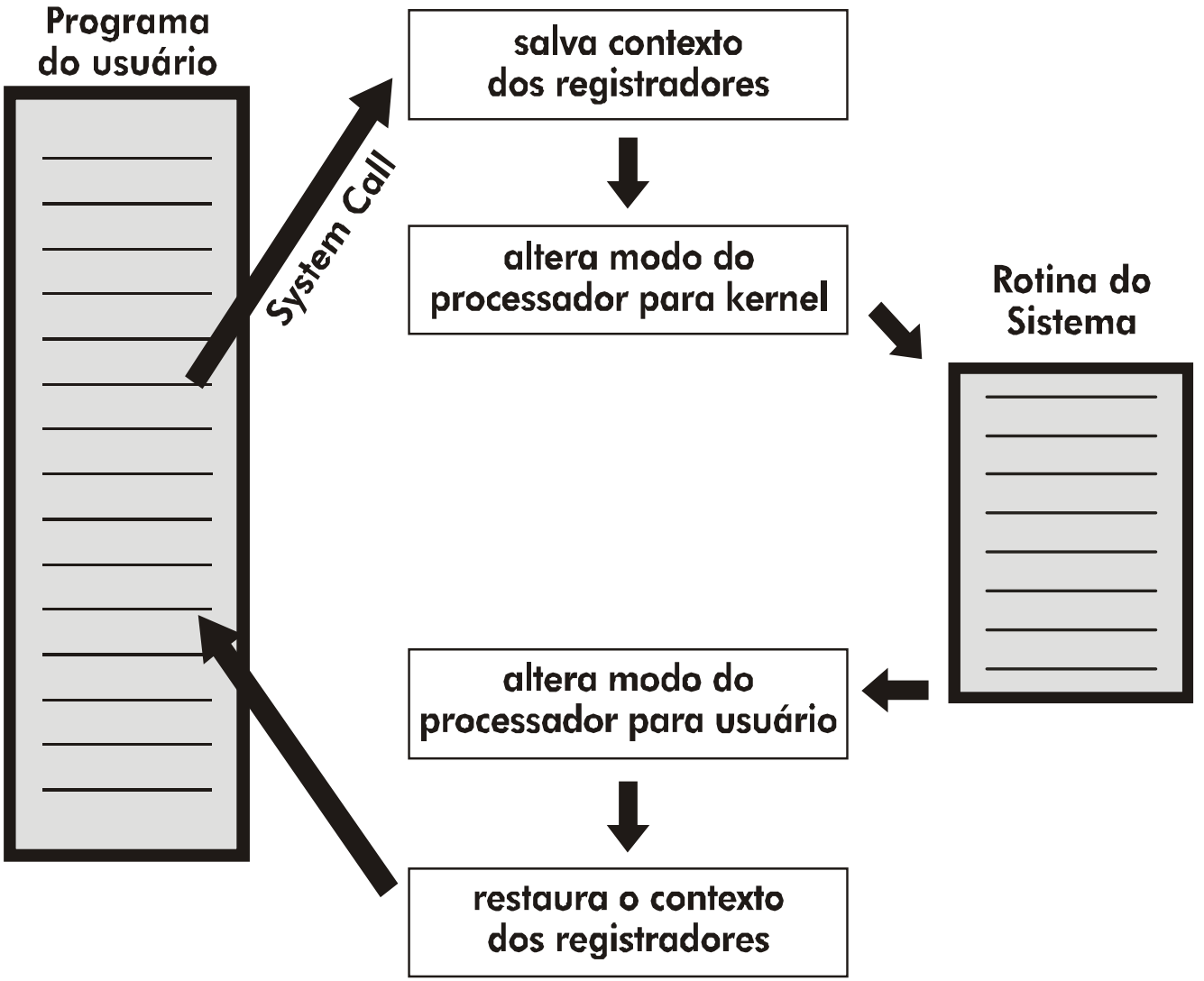
- O Sistema Operacional é formado por um conjunto de rotinas que oferece serviços aos usuários e às aplicações
  - Esse conjunto de rotinas é chamado de **núcleo do sistema** ou **kernel**
  - As operações de E/S por exemplo são privilegiadas e dessa forma necessitam de um mecanismo para serem acessadas pelos programas
  - Quando um programa deseja chamar uma rotina do SO é utilizado o mecanismo de **Chamada de Sistema** (*System Call*)
  - Os sistemas operacionais definem chamadas de sistema para todas as operações envolvendo o acesso a recursos de baixo nível (periféricos, arquivos, alocação de memória, etc.)
  - Geralmente as chamadas de sistema são oferecidas para as aplicações em modo usuário através de uma biblioteca do sistema (*system library*), que prepara os parâmetros, invoca a interrupção de software e retorna à aplicação os resultados obtidos (no Windows: API Win32, Unix/Linux: POSIX)

Chamada de sistema



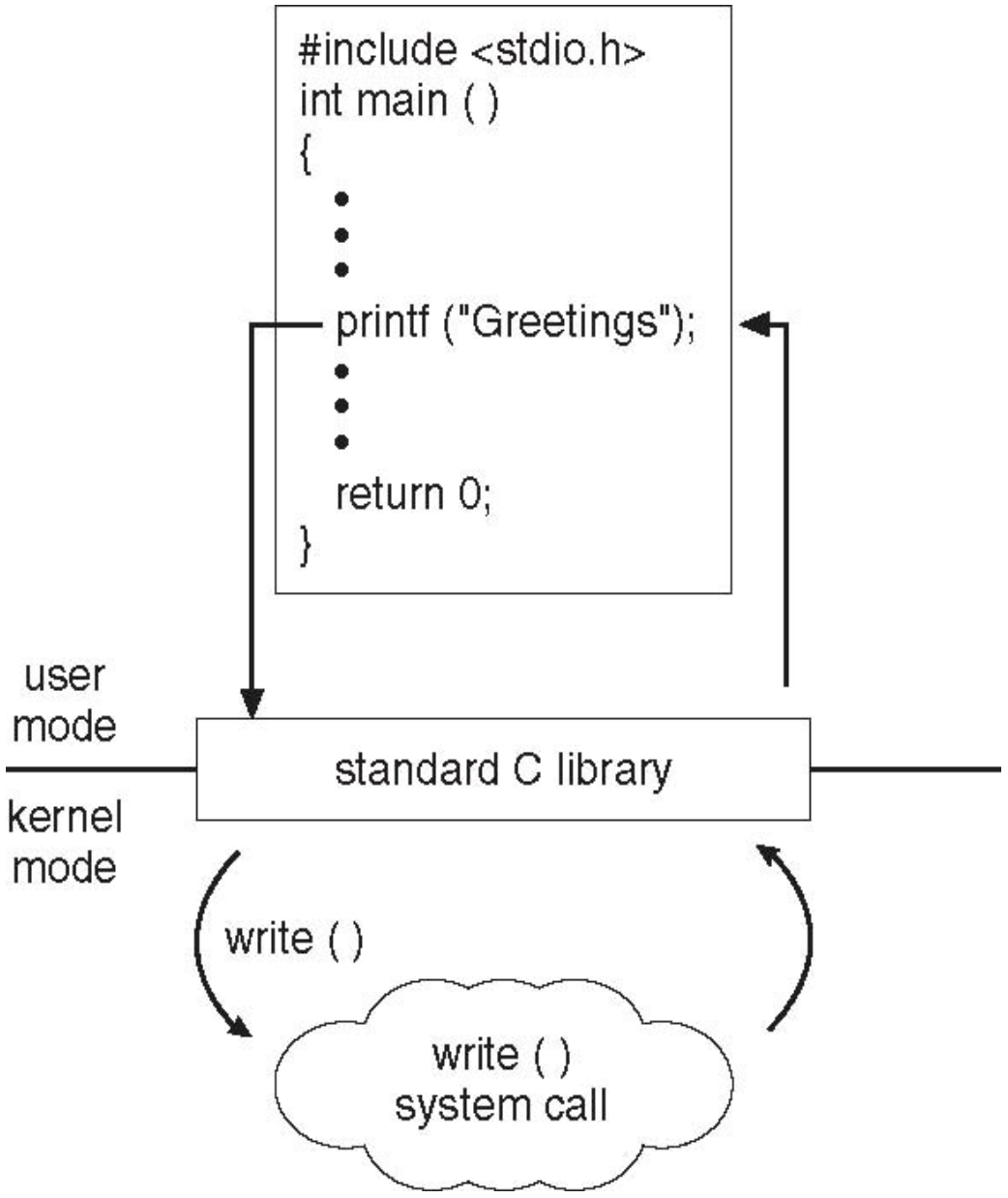
Chamada a uma rotina de sistema [Machado]

Chamada de sistema



Chamada a uma rotina de sistema [Machado]

Chamada a uma rotina do sistema



[Tanenbaum]

- A biblioteca C padrão oferece uma parte da interface de chamada de sistema para muitas versões Unix/Linux
- Como exempo, um programa C invoca a instrução printf, essa chamada é interceptada pela biblioteca C, que por sua vez invoca a chamada de sistema no SO (chamada de sistema write)
- A biblioteca C apanha o valor retornado por write e o passa de volta ao programa do usuário

Chamadas de Sistema (*System Calls*)

➢As **chamadas de sistema (*system calls*)** fornecem a **interface** entre um **programa em execução** e o **sistema operacional**

- Geralmente disponíveis como instruções em linguagem *assembly*
- Linguagens definidas para substituir a linguagem *assembly* para a programação de sistemas permitem que as *system calls* sejam feitas diretamente (por exemplo, C, C++)

➢Quando se invoca uma **Chamada de Sistema**, três métodos são usados para passar parâmetros entre o programa em execução e o Sistema Operacional

- **Passar parâmetros via *registradores***
- **Armazenar os parâmetros em uma tabela na memória**, e o endereço da tabela é passado como um parâmetro em um registrador (abordagem do Linux)
- O programa coloca (***push***) os parâmetros na pilha e o Sistema Operacional retira (***pop***) os parâmetros da pilha

Tipos de chamadas de sistema

- **Controle de processos**
- -> end, load, execute, create process, wait for time, etc.
- **Gerência de arquivos**
- -> create file, open, read, write, get files attributes, etc.
- **Gerência de dispositivos**
- -> request device, read, write, get devices attributes, etc.
- **Manutenção de informações**
- -> set time, get system data, set system data, etc.
- **Comunicações**
- -> create communication connection, send messages, etc.

Chamadas de sistema (gerenc. arquivos)

➢Exemplos de chamadas de sistemas no POSIX (Unix/Linux)

Gerenciamento de arquivos

Chamada	Descrição
Fd = open(file, how, ...)	Abre um arquivo para leitura, escrita ou ambos
s = close(fd)	Fecha um arquivo aberto
n = read(fd, buffer, nbytes)	Lê dados a partir de um arquivo em um buffer
n = write(fd, buffer, nbytes)	Escreve dados a partir de um buffer em um arquivo
position = lseek(fd, offset, whence)	Move o ponteiro do arquivo
s = stat(name, &buf)	Obtém informações sobre um arquivo

Tabela 1.1 Algumas das principais chamadas de sistema do POSIX. O código de retorno *s* é –1 se um erro tiver ocorrido. Os códigos de retorno são os seguintes: *pid* é um processo id, *fd* é um descritor de arquivo, *n* é um contador de bytes, *position* é uma compensação no interior do arquivo e *seconds* é o tempo decorrido. Os parâmetros são explicados no texto. [Tanenbaum]

Chamadas de sistema no Unix e Windows

➢Exemplos de chamadas de sistemas no POSIX (Unix/Linux)

UNIX	Win32	Descrição
fork	CreateProcess	Cria um novo processo
waitpid	WaitForSingleObject	Pode esperar que um processo saia
execve	(nenhuma)	CreateProcess = fork + execve
exit	ExitProcess	Conclui a execução
open	CreateFile	Cria um arquivo ou abre um arquivo existente
close	CloseHandle	Fecha um arquivo
read	ReadFile	Lê dados a partir de um arquivo
write	WriteFile	Escreve dados em um arquivo
lseek	SetFilePointer	Move o ponteiro do arquivo
stat	GetFileAttributesEx	Obtém vários atributos do arquivo
mkdir	CreateDirectory	Cria um novo diretório
rmdir	RemoveDirectory	Remove um diretório vazio
link	(nenhuma)	Win32 não dá suporte a links
unlink	DeleteFile	Destrói um arquivo existente
mount	(nenhuma)	Win32 não dá suporte a mount
umount	(nenhuma)	Win32 não dá suporte a mount
chdir	SetCurrentDirectory	Altera o diretório de trabalho atual
chmod	(nenhuma)	Win32 não dá suporte a segurança (embora o NT suporte)
kill	(nenhuma)	Win32 não dá suporte a sinais
time	GetLocalTime	Obtém o tempo atual

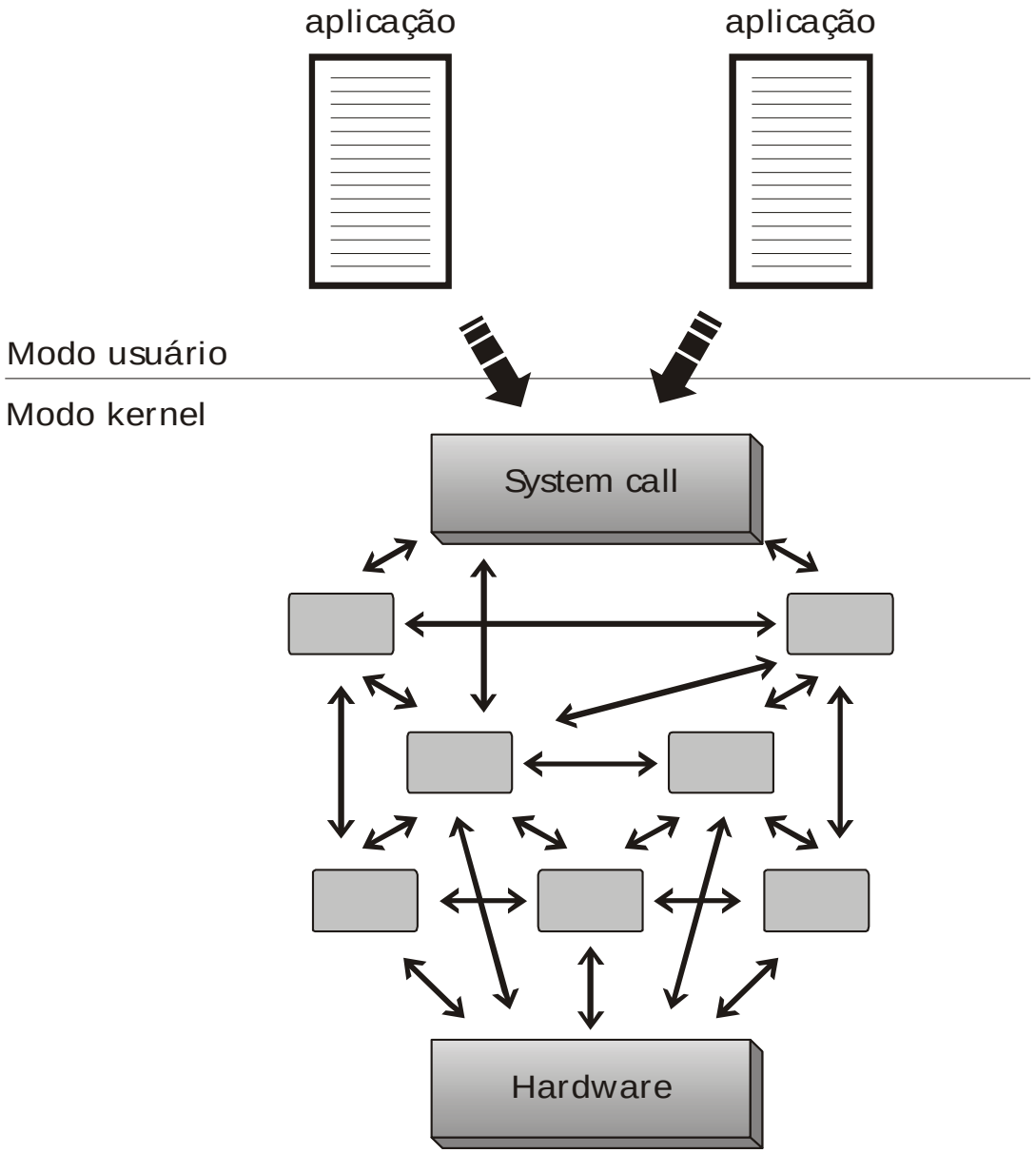
Tabela 1.2 As chamadas da API Win32 que correspondem aproximadamente às chamadas do UNIX da Tabela 1.1.

[Tanenbaum]

UFABC

Estrutura do Sistema Operacional

Monolítica



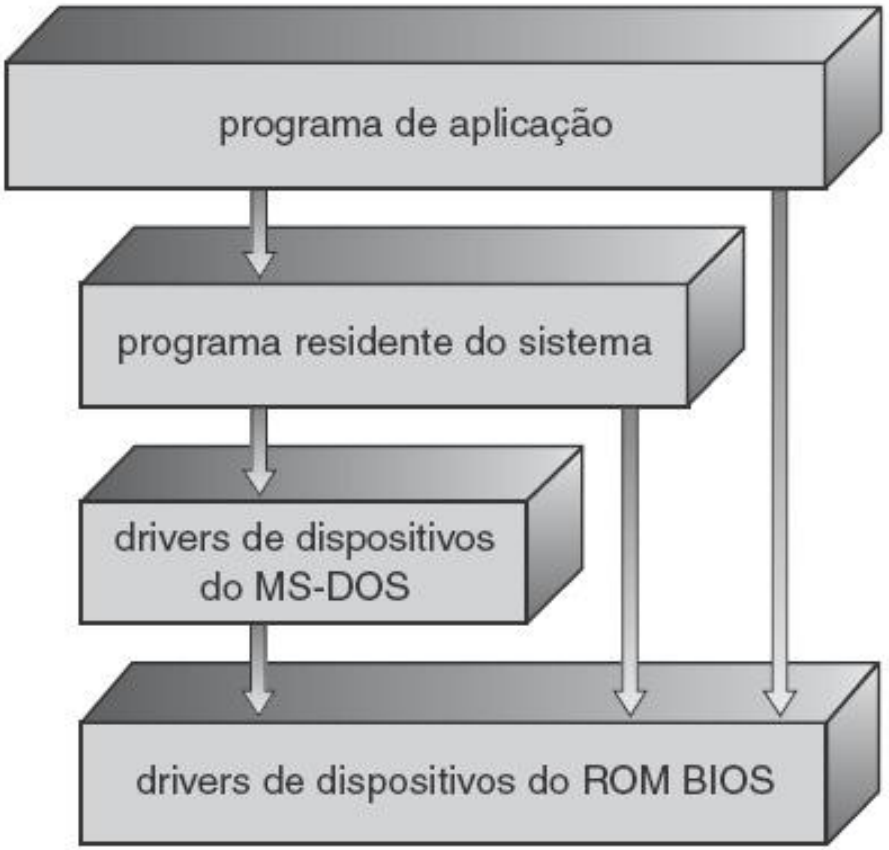
Arquitetura Monolítica [Silberschatz]

- Monolítico: que se comporta como um conjunto rígido, indivisível (dic. Aurélio)
- Todos os componentes do núcleo do sistema operam em modo núcleo e interagem diretamente (um grande e único programa executável)
- Simplicidade e desempenho
- Dificuldade de manuntenção e evolução do núcleo
- Foi utilizada nos primeiros sistemas operacionais
- E também no MS-DOS e primeiros sistemas Unix

Estrutura do Sistema MS-DOS

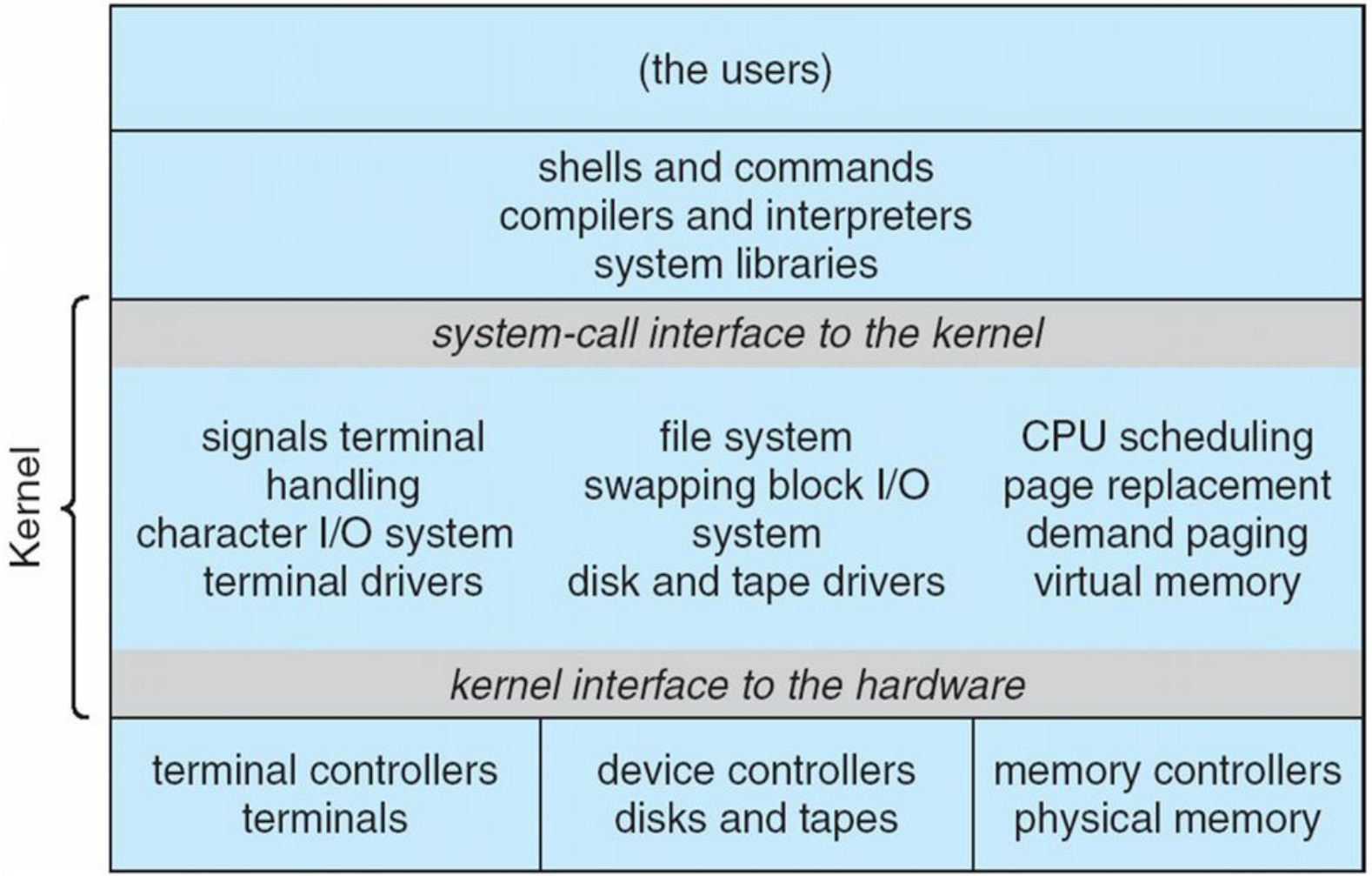
- MS-DOS – escrito para fornecer o **máximo de funcionalidade no menor espaço possível** (por causa do *hardware* limitado no qual ele era operado)
  - **Não foi dividido cuidadosamente em módulos** (começou como um sistema pequeno, simples e limitado, e depois cresceu além do seu escopo original)
  - Embora o MS-DOS tenha alguma estrutura, suas interfaces e níveis de funcionalidade não são bem separados
  - Os programas aplicativos podem acessar as rotinas básicas de E/S para escrever diretamente na tela e nas unidades de disco (podendo causar problemas de segurança)
  - Foi escrito para o processador Intel 8088 que não provia proteção de hardware, modo dual (sistema ficou limitado ao hardware)

Estrutura em Camadas do MS-DOS



Estrutura do MS-DOS [Silberschatz]

Estrutura do UNIX tradicional



Estrutura do Sistema Unix [Silberschatz]

Enfoque em camadas

➢O Sistema Operacional é dividido em uma **série de camadas** (níveis), cada uma construída sobre as camadas inferiores. A camada mais baixa (camada 0) é o *hardware*; a mais alta (camada N) é a interface do usuário

➢A principal vantagem da abordagem em camadas é a **modularidade**

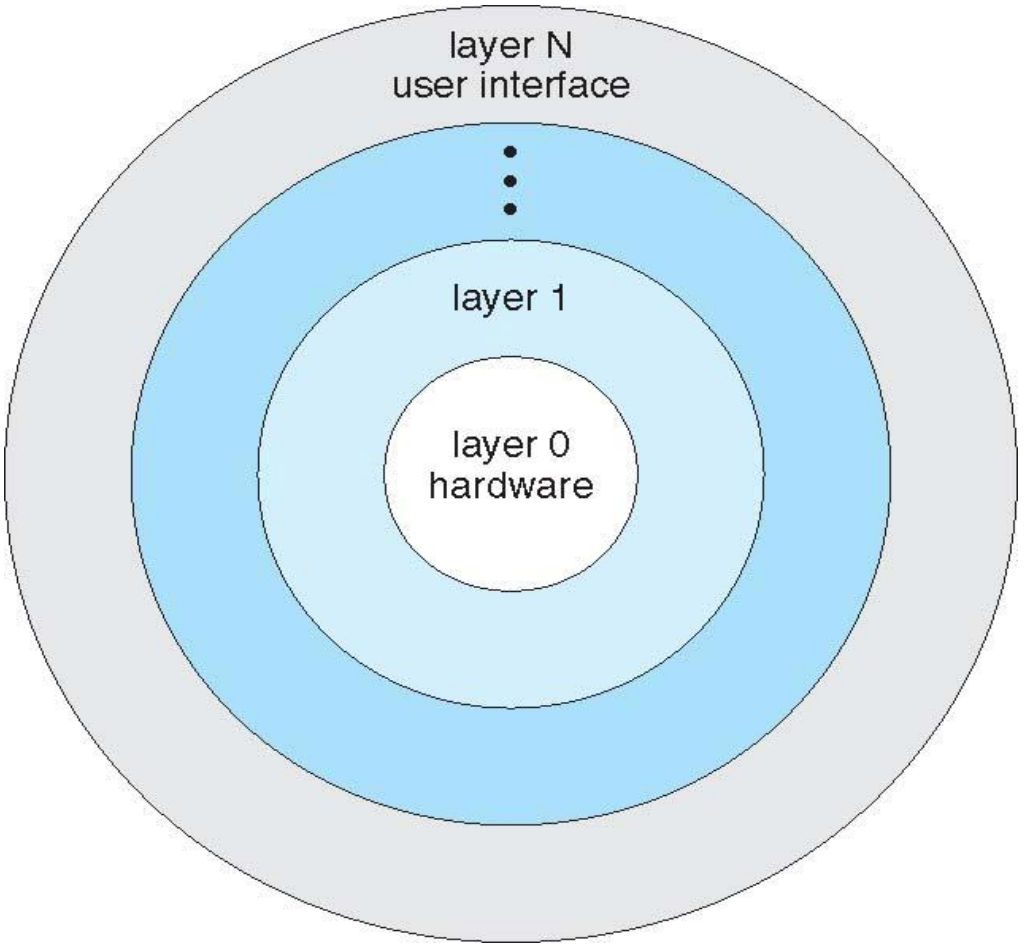
➢Cada camada utiliza funções (operações) e serviços apenas de camadas de nível mais baixo

➢A principal dificuldade com esse enfoque em camadas envolve a definição apropriada das diversas camadas (e a forma de interação entre elas)

➢Esse enfoque é menos eficiente, cada camada acrescenta um custo adicional (overhead) à chamada de sistema

➢Exemplos: IBM OS/2, MULTICS

Enfoque em Camadas



Um Sistema Operacional em camadas [Silberschatz]

Microkernel

➢A maioria dos Sistemas Operacionais modernos implementa ***módulos*** ou ***microkernels***

- Usa técnicas orientadas a objeto
- Cada componente básico é separado
- Cada um se comunica com os outros através de interfaces conhecidas (troca de mensagens)

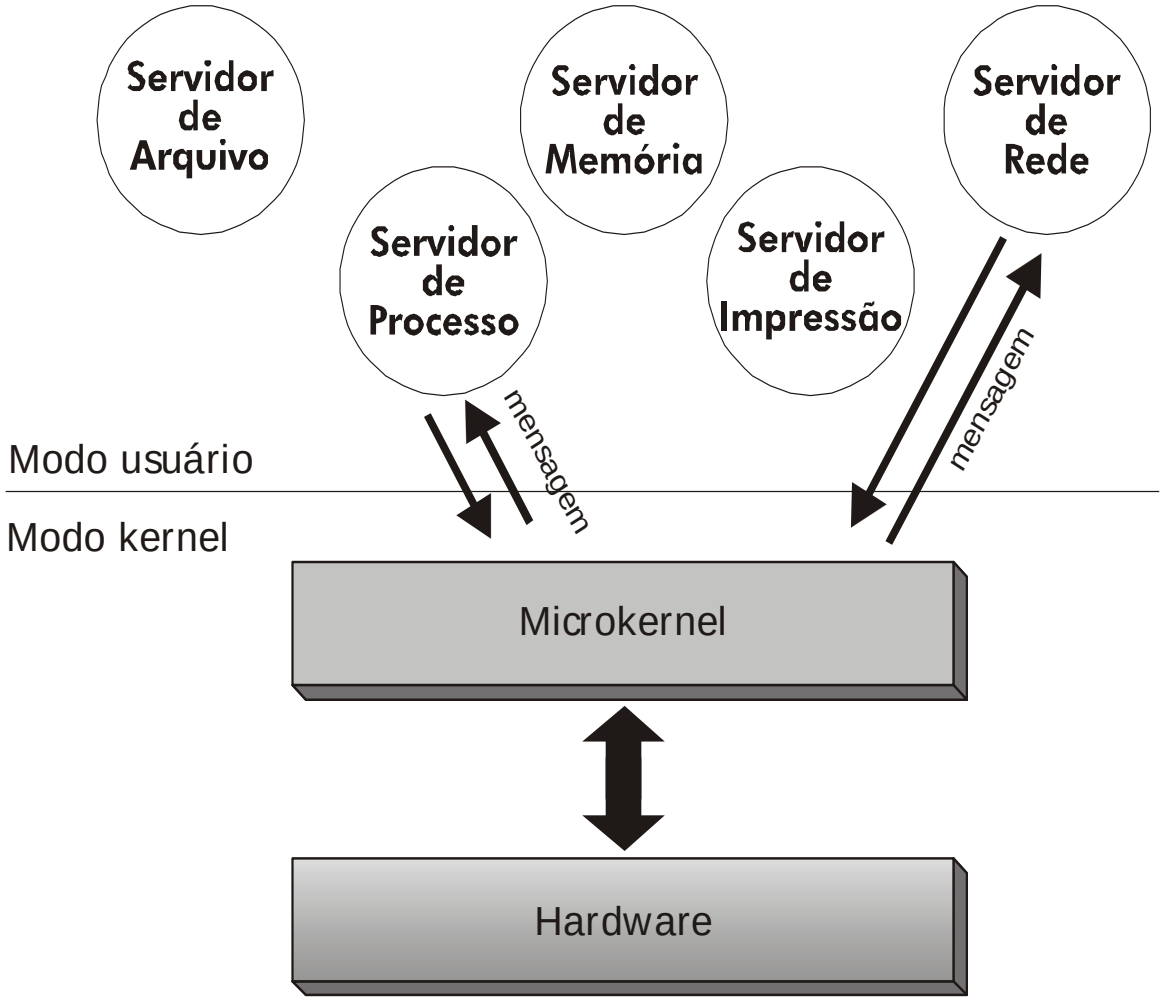
➢Somente os componentes essenciais são mantidos no kernel do SO, enquanto que componentes não essenciais são implementados como programas em nível de sistema ou usuário => resulta em um kernel menor

➢Em geral, *microkernels* oferece uma gerência mínima de processo e memória e facilidade de comunicação

➢Dentre algumas vantagens:

- A expansão do sistema torna-se mais fácil
- Todos os novos serviços são adicionados ao espaço do usuário e, consequentemente, não exigem a modificação do *kernel*
- Ex.: MacOS X que tem suas raízes no sistema Mach, Digital UNIX

Microkernel



Arquitetura microkernel [Machado]

- Os serviços do sistema são implementados através de processos, onde cada um é responsável por oferecer um conjunto de funções
- A principal função do microkernel é possibilitar a comunicação entre o programa cliente e os serviços oferecidos no espaço do usuário (troca de mensagens)
- Se um programa quer acessar um arquivo, ele precisa interagir com o Servidor de Arquivo, mas essa comunicação é indireta, através da troca de mensagens com o microkernel
- O custo associado às trocas de mensagens entre componentes pode ser bastante elevado, o que

Máquinas virtuais

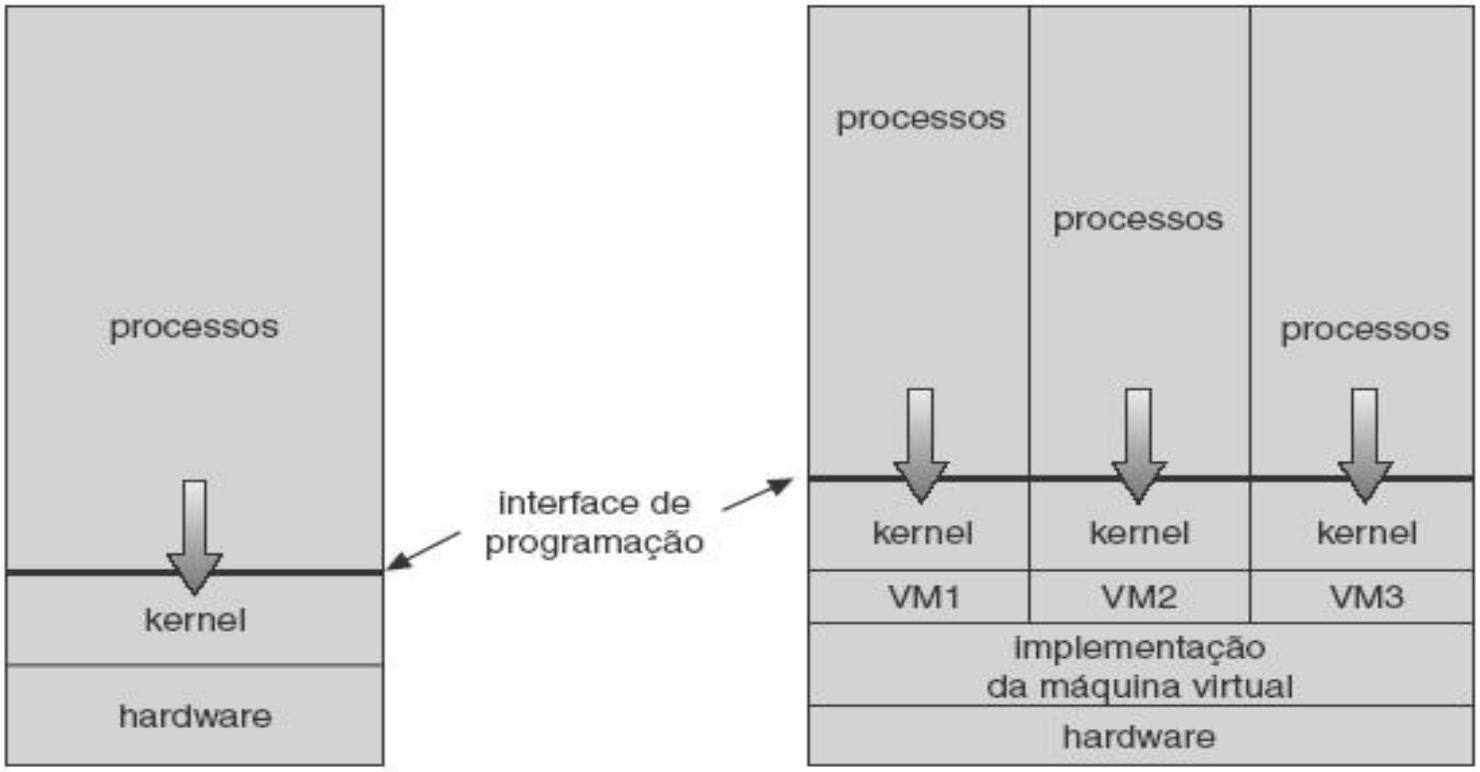
- A principal ideia das máquinas virtuais é utilizar um mesmo hardware em vários ambientes de execução diferentes
  - Cria-se a ilusão de que cada ambiente de execução separado está executando em um computador particular
  - E, a principal motivação é a capacidade de compartilhamento do mesmo hardware entre diferentes ambientes de execução (i.e., diferentes sistemas operacionais) ao mesmo tempo
- A virtualização pode ser utilizada em muitas situações:
  - Os vários servidores de uma companhia (servidor web, servidor de banco de dados, servidor de e-mail, etc.) que utilizam diferentes SOs, podem ser executados em uma mesma máquina, sem que uma falha afete o restante
  - Hospedagem de sites Web: clientes hospedam suas páginas em máquinas virtuais (flexibilidade quanto à configuração de software, menor custo em relação a um servidor dedicado)
  - Usuário final que deseja utilizar dois ou mais sistemas operacionais na mesma máquina

Máquinas virtuais (cont.)

- Uma **máquina virtual** oferece uma **interface** que é **idêntica** à do **hardware básico**
- O Sistema Operacional cria a **ilusão de múltiplos processos**, cada um sendo executado em **seu próprio processador**, com **sua própria memória** (virtual)

- Os **recursos** do computador físico são **compartilhados** para criar as máquinas virtuais
  - O **escalonamento de CPU** pode criar a aparência de que os usuários possuem seus próprios processadores
  - Um **terminal de usuário** normal de tempo compartilhado age como o console do operador da máquina virtual

Máquinas virtuais (cont.)



Máquina não-virtual

Máquina virtual [Silberschatz]

Com as Máquinas Virtuais, os usuários podem executar diferentes Sistemas Operacionais ou pacotes de software disponíveis

Vantagens e desvantagens das máquinas virtuais

➢O conceito de **máquina virtual** provê **proteção completa dos recursos do sistema**, uma vez que **cada máquina virtual é completamente isolada de todas as outras máquinas virtuais**. Esse isolamento, entretanto, **não permite** qualquer **compartilhamento direto dos recursos**

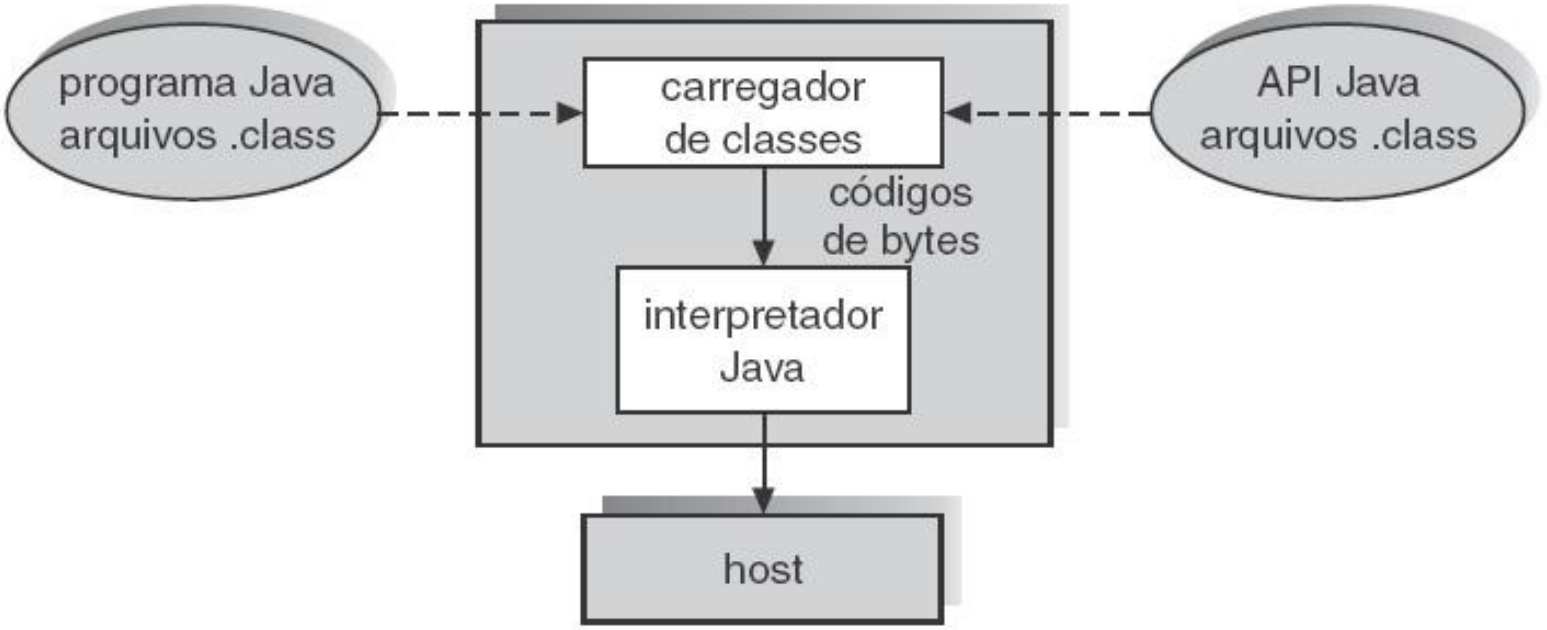
➢Um sistema de máquina virtual é um **veículo perfeito para pesquisa e desenvolvimento de Sistemas Operacionais**. O desenvolvimento do sistema é feito na máquina virtual, e não diretamente em uma máquina física, e não interrompe a operação normal do sistema

➢O conceito de máquina virtual é **difícil de implementar** devido ao esforço necessário para oferecer uma **cópia exata da máquina utilizada** (a máquina básica tem dois modos: **modo usuário** e **modo monitor**)

Máquina virtual Java

- Os programas Java compilados são *bytecodes* para plataforma neutra executados por uma **Máquina Virtual Java** (**JVM**)
- A **JVM** consiste em:
  - Carregador de classes
  - Interpretador em tempo de execução
- Há basicamente dois tipos de interpretadores:
- **Interpretador de *software:*** interpreta um *bytecode* de cada vez
- **Compilador *Just-In-Time:*** transforma os *bytecodes* independentes de arquitetura em linguagem de máquina nativa para o computador *host*
- A maioria das implementações da JVM utiliza um **compilador JIT para aumentar o desempenho**
- O interpretador pode também ser implementado em um *chip* de *hardware* que executa os *bytecodes* Java

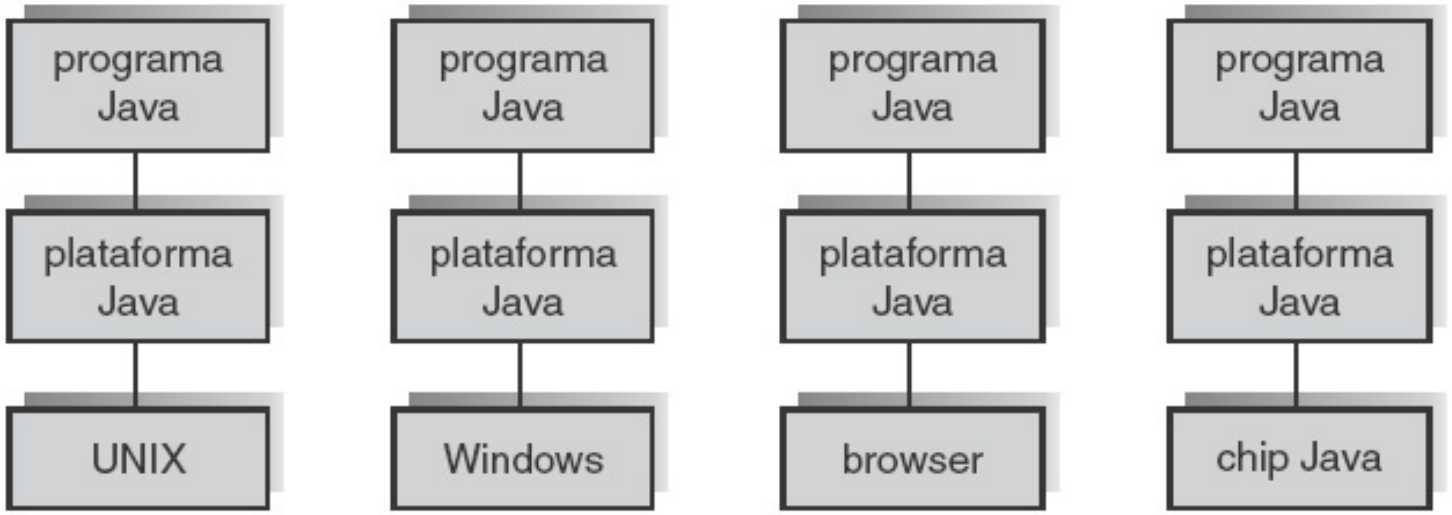
A Máquina Virtual Java



A máquina vitual Java [Silberschatz]

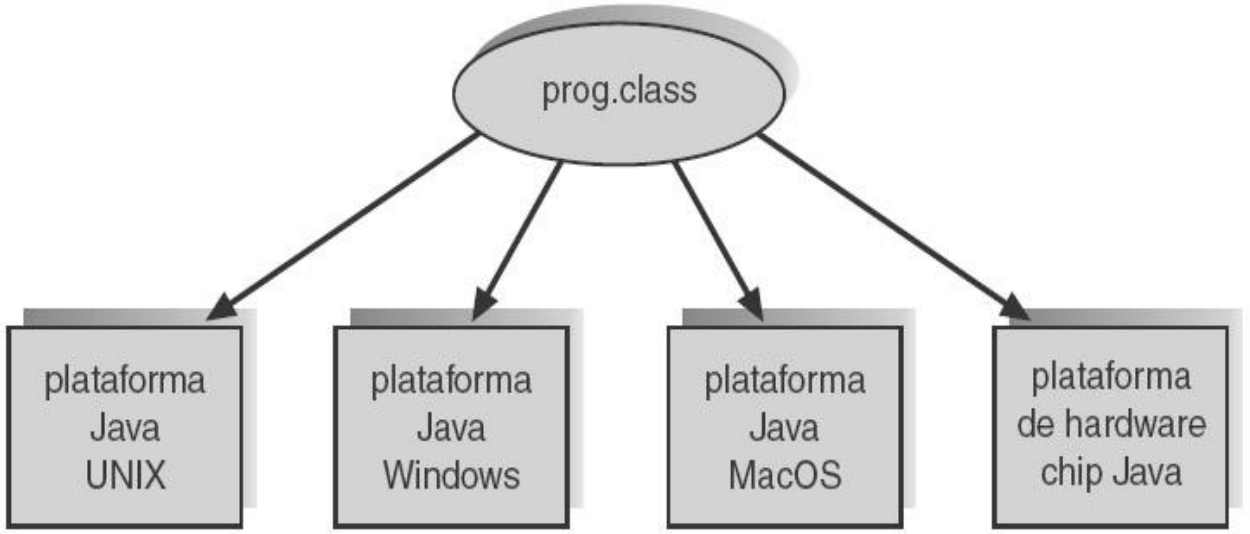
Uma instância de JVM é criada sempre que uma aplicação Java é executada. Se executarmos simultaneamente dois programas Java no mesmo computador, teremos três instâncias JVM

A Plataforma Java



A plataforma Java pode ser implementada sobre um Sistema Operacional *host* (UNIX, Windows etc.), como parte de um navegador *Web* ou em *hardware*

Arquivo java.class em diferentes plataformas

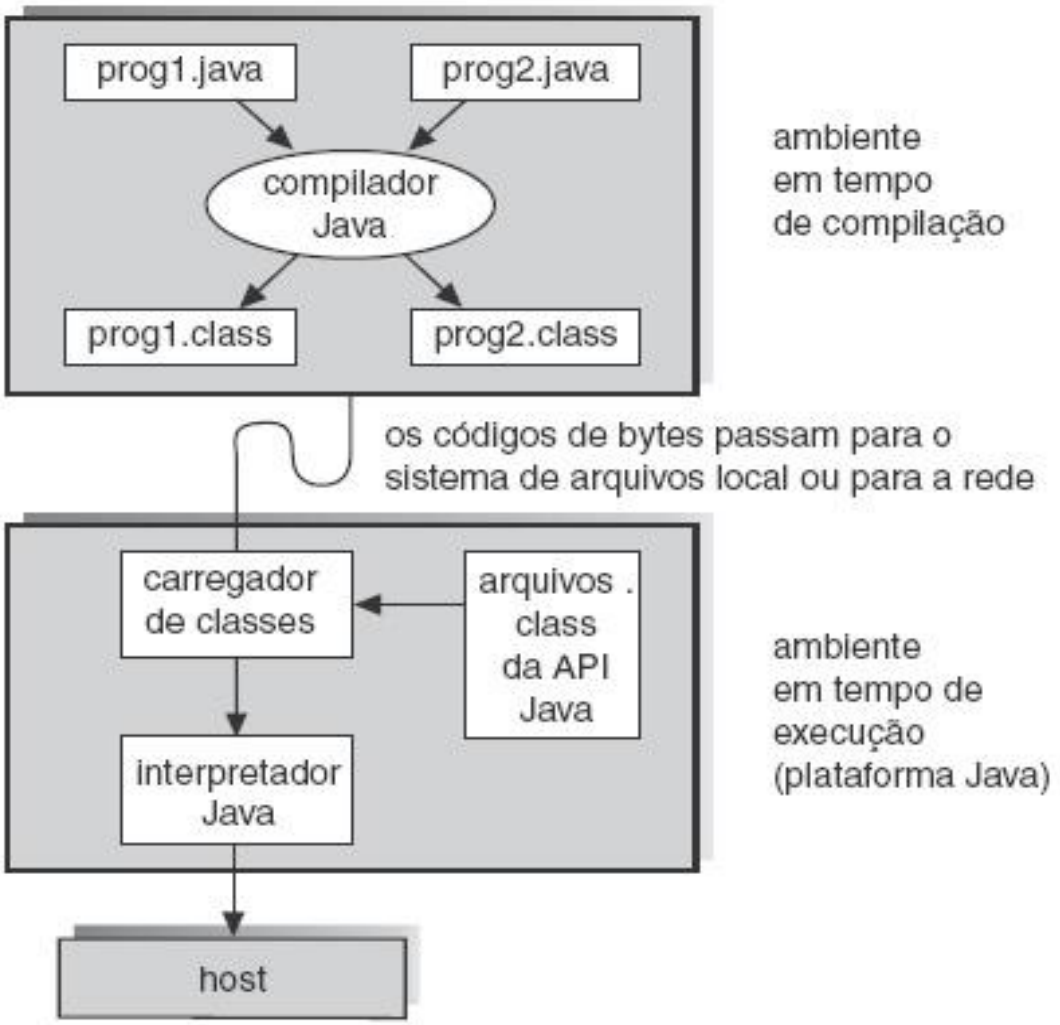


Arquivo .class em várias plataformas [Silberschatz]

➢A plataforma Java torna possível desenvolver programas independentes de arquitetura e portáteis.

➢Um arquivo .class executa em qualquer sistema que tenha implementada a JVM

Ambiente de desenvolvimento Java (Java Development Environment - JDE)



Ambiente de desenvolvimento Java [Silberschatz]

Referências

- [Silberschatz] SILBERCHATZ, A., GALVIN, P. B. e GAGNE, G. **Sistemas Operacionais com Java**. 7ª ed., Rio de Janeiro: Elsevier, 2008.
- [Tanenbaum] TANENBAUM, A. **Sistemas Operacionais Modernos**. 3ª ed. São Paulo: Prentice Hall, 2009.
- [MACHADO] MACHADO, F. B. e MAIA, L. P. **Arquitetura de Sistemas Operacionais**. 4ª ed., Rio de Janeiro: LTC, 2007.