Sistemas Operacionais

Sistemas de Informação – EACH / USP – 2° Semestre de 2020

Prof. Dr. Alexandre da Silva Freire

Aula 2: Estruturas do sistema operacional

Silberschatz, Galvin e Gagne ©2007

Cap. 2: Estruturas do sistema operacional

- Serviços
- Interface com o usuário
- Chamadas de sistema
- Tipos de chamadas de sistema
- Programas do sistema
- Projeto e implementação
- Estrutura
- Máquinas virtuais
- Geração
- Boot do sistema

Objetivos

- Descrever os serviços que um SO oferece aos usuários e outros sistemas
- Discutir as várias maneiras de estruturar um SO
- Explicar como os SO's são instalados e personalizados e como eles realizam o boot

Serviços do sistema operacional

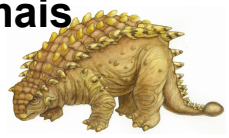
- Um conjunto de serviços do sistema operacional oferece funções que são úteis ao usuário:
  - **Interface com o usuário (UI)**
    - Varia entre linha de comando (CLI), interface gráfica com o usuário (GUI), batch
  - **Execução do programa** – carregar na memória, executar e terminar a execução, normal ou anormalmente (indicando erro)
  - **Operações de E/S** – Um programa em execução pode exigir E/S, que pode envolver um arquivo ou um dispositivo de E/S.
  - **Manipulação do sistema de arquivos** – os programas precisam ler e gravar arquivos e diretórios, criá-los e excluí-los, pesquisá-los, listar informação do arquivo, gerenciamento de permissão.

Serviços do sistema operacional (cont.)

- Um conjunto de serviços do sistema operacional oferece funções que são úteis ao usuário (cont.):
  - Comunicações – Processos podem trocar informações, no mesmo computador ou entre computadores de uma rede
    - As comunicações podem ser via memória compartilhada ou por passagem de mensagens (pacotes movidos pelo SO)
  - Detecção de erro – O SO precisa estar continuamente ciente dos erros possíveis
    - Pode ocorrer na CPU e no hardware de memória, em dispositivos de E/S, no programa do usuário
    - Para cada tipo de erro, o SO deve tomar a ação apropriada para garantir a computação correta e coerente
    - Facilidades de depuração podem melhorar bastante as capacidades do usuário e do programador de usar o sistema de modo eficiente

Serviços do sistema operacional (cont.)

- Outro conjunto de funções do SO existe para garantir a operação eficiente do próprio sistema por meio do compartilhamento de recursos
  - **Alocação de recursos** – Quando vários usuários ou várias tarefas estão executando simultaneamente, os recursos devem ser alocados a cada um deles
    - Muitos tipos de recursos – Alguns (como ciclos de CPU, memória principal e armazenamento de arquivo) podem ter código de alocação especial, outros (como dispositivos de E/S) podem ter código geral de solicitação e liberação.
  - **Contabilidade** – Registrar quais usuários usam quantos e que tipos de recursos do computador
  - **Proteção e segurança** – Os proprietários da informação armazenada em um sistema de computador multiusuário ou em rede podem querer controlar o uso dessa informação; processos concorrentes não deverão interferir uns nos outros
    - A proteção envolve garantir que todo o acesso aos recursos do sistema é controlado
    - Segurança do sistema contra estranhos requer autenticação do usuário, se estende para defender dispositivos de E/S externos contra tentativas de acesso inválidas
    - Se um sistema tiver que ser protegido e seguro, nele devem ser instituídas precauções. **Uma cadeia é tão forte quanto seu elo mais fraco**.



Interface de comando do sistema operacional - CLI

CLI permite entrada direta de comando

- Às vezes implementado no kernel, às vezes pelo sistema operacional
- Às vezes, múltiplos tipos implementados - shells
- Principalmente, busca um comando do usuário e o executa
  - Às vezes, comandos internos, às vezes apenas nomes de programas
    - No segundo caso, a inclusão de novos recursos não exige modificação do shell

Interface com o usuário do sistema operacional - GUI

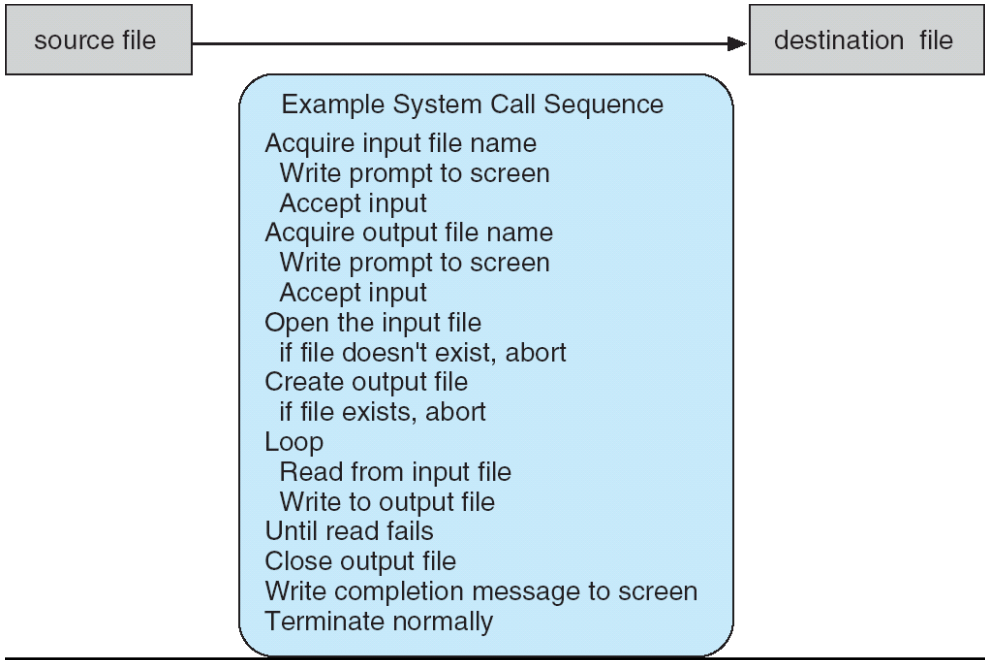
- Interface amigável da metáfora do desktop
  - Normalmente mouse, teclado e monitor
  - **Ícones** representam arquivos, programas, ações etc.
  - Diversos botões do mouse sobre objetos na interface causam diversas ações: fornecer informações, opções, executar função, abrir diretório (conhecido como pasta)
  - Inventado na Xerox PARC
- Muitos sistemas agora incluem CLI e GUI
  - Microsoft Windows é GUI com shell de “comando” CLI
  - Apple Mac OS X como interface GUI “Aqua” com kernel do UNIX por baixo e shells disponíveis
  - Solaris é CLI com interfaces GUI opcionais (Java Desktop, KDE)

Chamadas do sistema

- Interface de programação para os serviços fornecidos pelo SO.
- Normalmente, escritas em uma linguagem de alto nível (C ou C++).
- Acessadas principalmente pelos programas por meio de uma **Application Program Interface (API)** de alto nível, ao invés do uso da chamada direta do sistema
- Três APIs mais comuns são Win32 API para Windows, POSIX API para sistemas baseados em POSIX (incluindo praticamente todas as versões do UNIX, Linux e Mac OS X) e Java API para a Java Virtual Machine (JVM)

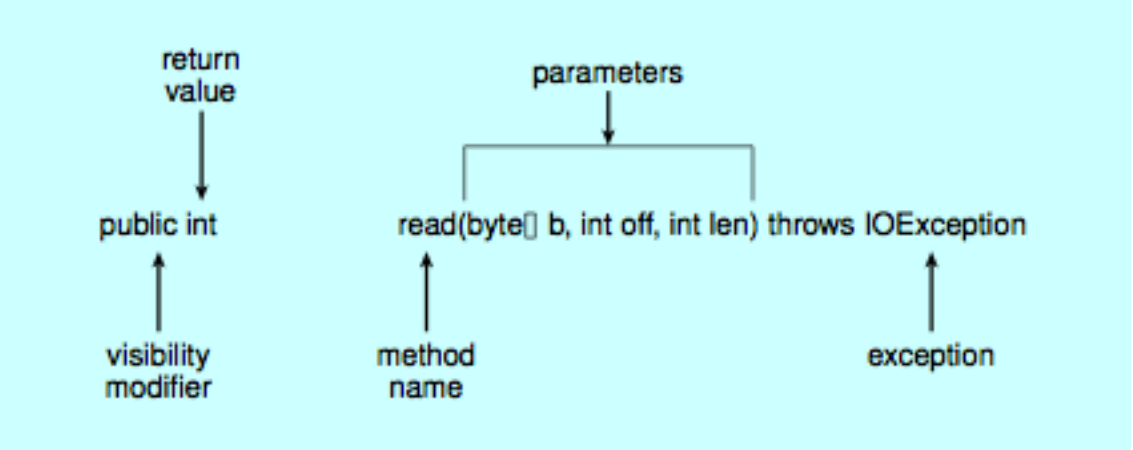
Exemplo de chamadas do sistema

- Seqüência de chamada do sistema para copiar o conteúdo de um arquivo para outro



Exemplo da API padrão

- Considere o comando Java read()



byte[] b – o buffer no qual os dados são lidos

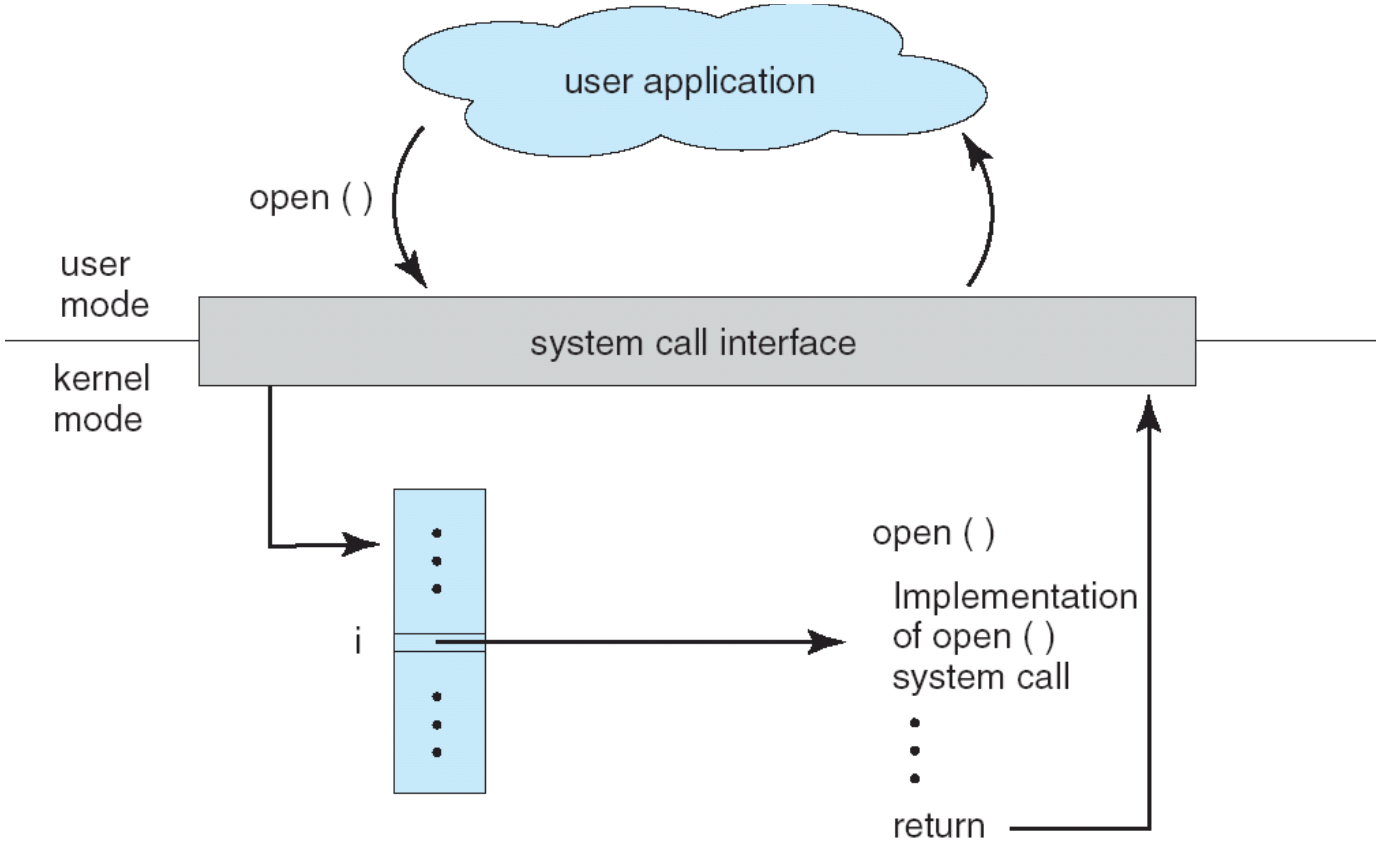
int off – o offset inicial em b onde os dados são lidos

int len – o número máximo de bytes a serem lidos

Implementação da chamada do sistema

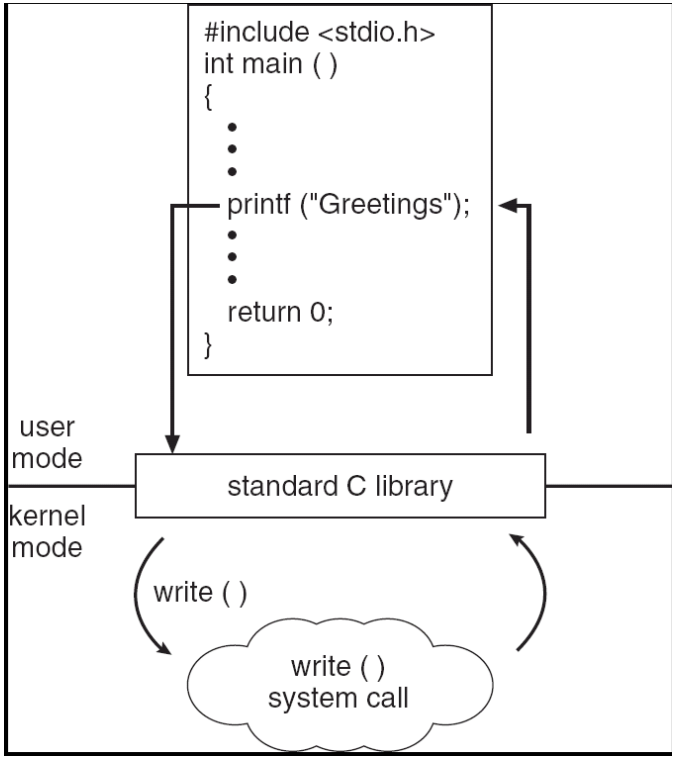
- Normalmente, um número associado a cada chamada do sistema
  - A interface de chamada do sistema mantém uma tabela indexada de acordo com esses números
- A interface de chamada do sistema invoca a chamada do sistema no kernel do SO e retorna o status e quaisquer valores de retorno
- Quem chama não precisa saber nada sobre como a chamada do sistema foi implementada
  - Só precisa obedecer a API e entender **o que** o SO fará
  - A maioria dos detalhes da interface do SO são ocultados do programador pela API

API – chamada do sistema – relacionamento do SO



Exemplo de biblioteca C padrão

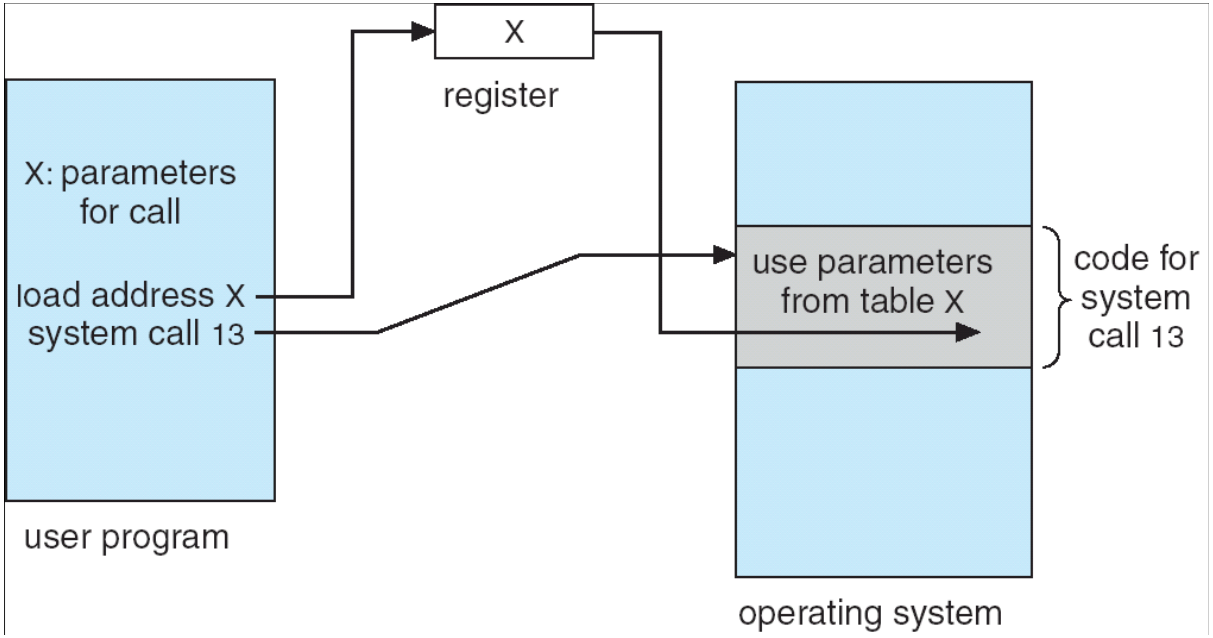
- Programa C invocando chamada de biblioteca printf(), que chama a chamada do sistema write()



Passagem de parâmetro na chamada do sistema

- Três métodos gerais usados para passar parâmetros ao SO
  - Mais simples: passar parâmetros nos *registradores*
    - Problema: em alguns casos, pode ser mais parâmetros do que registradores
  - Parâmetros armazenados em um *bloco,* ou tabela, na memória, e endereço do bloco passado como um parâmetro em um registrador
  - Parâmetros colocados, ou *empurrados,* na *pilha* pelo programa e *retirados* da pilha pelo sistema operacional
  - Métodos de bloco e pilha não limitam o número ou a extensão dos parâmetros sendo passados

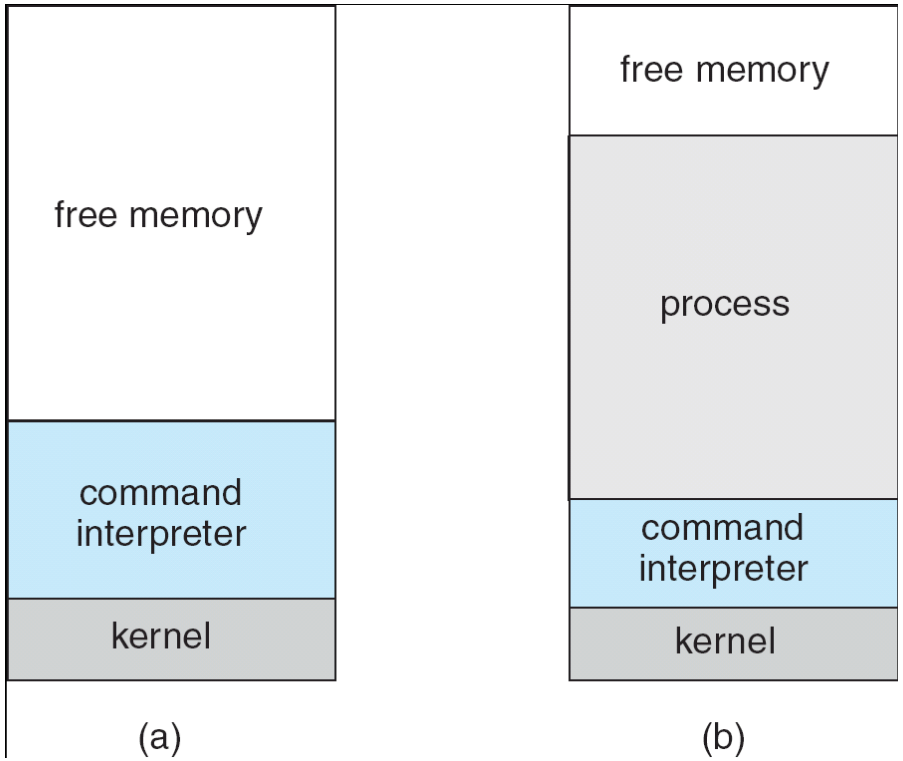
Passagem de parâmetros via tabela



Tipos de chamadas do sistema

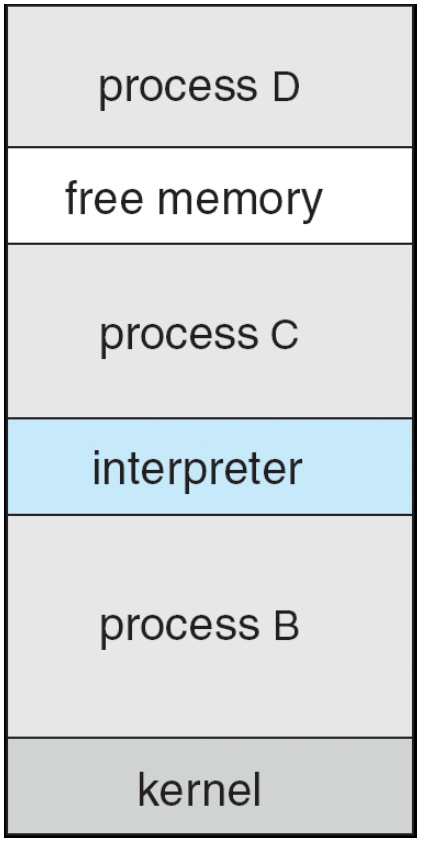
- Controle de processo
- Gerenciamento de arquivo
- Gerenciamento de dispositivo
- Manutenção de informações
- Comunicações

Execução no MS-DOS (monotarefa)



(a) Na partida do sistema (b) Executando um programa

Execução no FreeBSD (multitarefa)



Programas do sistema

- Programas do sistema oferecem um ambiente conveniente para desenvolvimento e execução do programa. Eles podem ser divididos em:
  - Manipulação de arquivo
  - Informação de status
  - Modificação de arquivo
  - Suporte à linguagem de programação
  - Carga e execução do programa
  - Comunicações
  - Programas de aplicação
- A visão do sistema operacional pela maioria dos usuários é definida por programas do sistema, e não pelas chamadas do sistema reais

dtrace do Solaris 10 seguindo chamada do sistema

```
# ./all.d `pgrep xclock` XEventsQueued
dtrace: script './all.d' matched 52377 probes
CPU FUNCTION
  0 -> XEventsQueued                        U
  0   -> _XEventsQueued                     U
  0     -> _X11TransBytesReadable           U
  0     <- _X11TransBytesReadable           U
  0     -> _X11TransSocketBytesReadable     U
  0     <- _X11TransSocketBytesreadable     U
  0     -> ioctl                            U
  0       -> ioctl                          K
  0         -> getf                         K
  0           -> set_active_fd              K
  0           <- set_active_fd              K
  0         <- getf                         K
  0         -> get_udatamodel               K
  0         <- get_udatamodel               K
...
  0         -> releasef                     K
  0           -> clear_active_fd            K
  0           <- clear_active_fd            K
  0           -> cv_broadcast               K
  0           <- cv_broadcast               K
  0         <- releasef                     K
  0       <- ioctl                          K
  0     <- ioctl                            U
  0   <- _XEventsQueued                     U
  0 <- XEventsQueued                        U
```

Programas do sistema

- Fornecem um ambiente conveniente para desenvolvimento e execução de programa
  - Alguns deles são simplesmente interfaces com o usuário para chamadas do sistema; outros são muito mais complexos
- Gerenciamento de arquivo – Criam, excluem, copiam, renomeiam, imprimem, listam e geralmente manipulam arquivos e diretórios
- Informação de status
  - Alguns pedem informações do sistema – data, hora, quantidade de memória disponível, espaço em disco, número de usuários
  - Outros oferecem informações detalhadas de desempenho, log e depuração
  - Normalmente, esses programas formatam e imprimem a saída no terminal ou outros dispositivos de saída
  - Alguns sistemas implementam um registro – usado para armazenar e recuperar informações de configuração

Programas do sistema (cont.)

- Modificação de arquivo
  - Editores de texto para criar e modificar arquivos
  - Comandos especiais para pesquisar conteúdo de arquivos ou realizar transformações do texto
- Suporte a linguagem de programação – Compiladores, assemblers, depuradores e interpretadores às vezes fornecidos
- Carga e execução de programa – Carregadores, editores de vínculo e sistemas de depuração para linguagem de alto nível e de máquina
- Comunicações – Oferecem mecanismo para criar conexões virtuais entre processos, usuários e sistemas de computação

Projeto e implementação do sistema operacional

- Algumas técnicas provaram ser bem sucedidas
- Estrutura interna de diferentes sistemas operacionais pode variar bastante
- Comece definindo objetivos e especificações
- Afetado pela escolha do hardware, tipo do sistema
- Objetivos do *usuário* e objetivos do *sistema*
  - Objetivos do usuário – o sistema operacional deve ser conveniente de usar, fácil de aprender, confiável, seguro e rápido
  - Objetivos do sistema – sistema operacional deve ser fácil de projetar, implementar e manter, além de ser flexível, confiável, livre de erro e eficiente

Projeto e implementação do sistema operacional

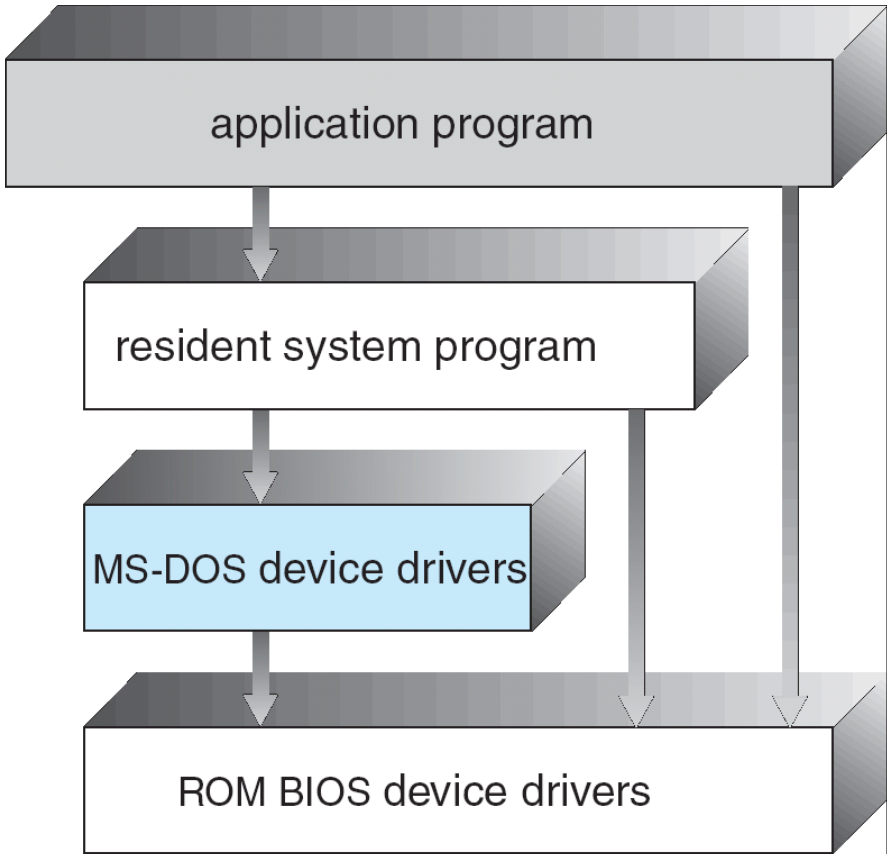
- Princípio importante para separar

  Política: O que será feito?
  Mecanismo: Como fazer isso?
- Mecanismos determinam *como* fazer algo, políticas decidem *o que* será feito
  - A separação entre política e mecanismo é um princípio muito importante, permite o máximo de flexibilidade se decisões políticas tiverem que ser alteradas mais tarde

Estrutura simples

- MS-DOS – escrito para oferecer o máximo de funcionalidade no menor espaço
  - Não dividido em módulos
  - Embora o MS-DOS tenha alguma estrutura, suas interfaces e níveis de funcionalidade não são bem separados

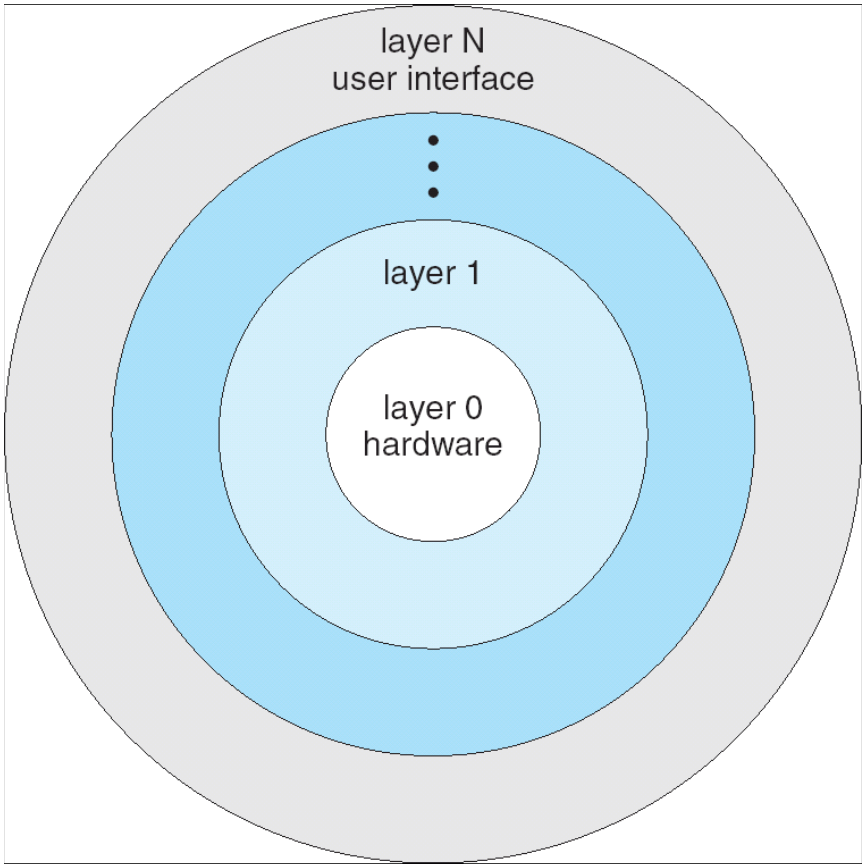
Estrutura em camadas do MS-DOS



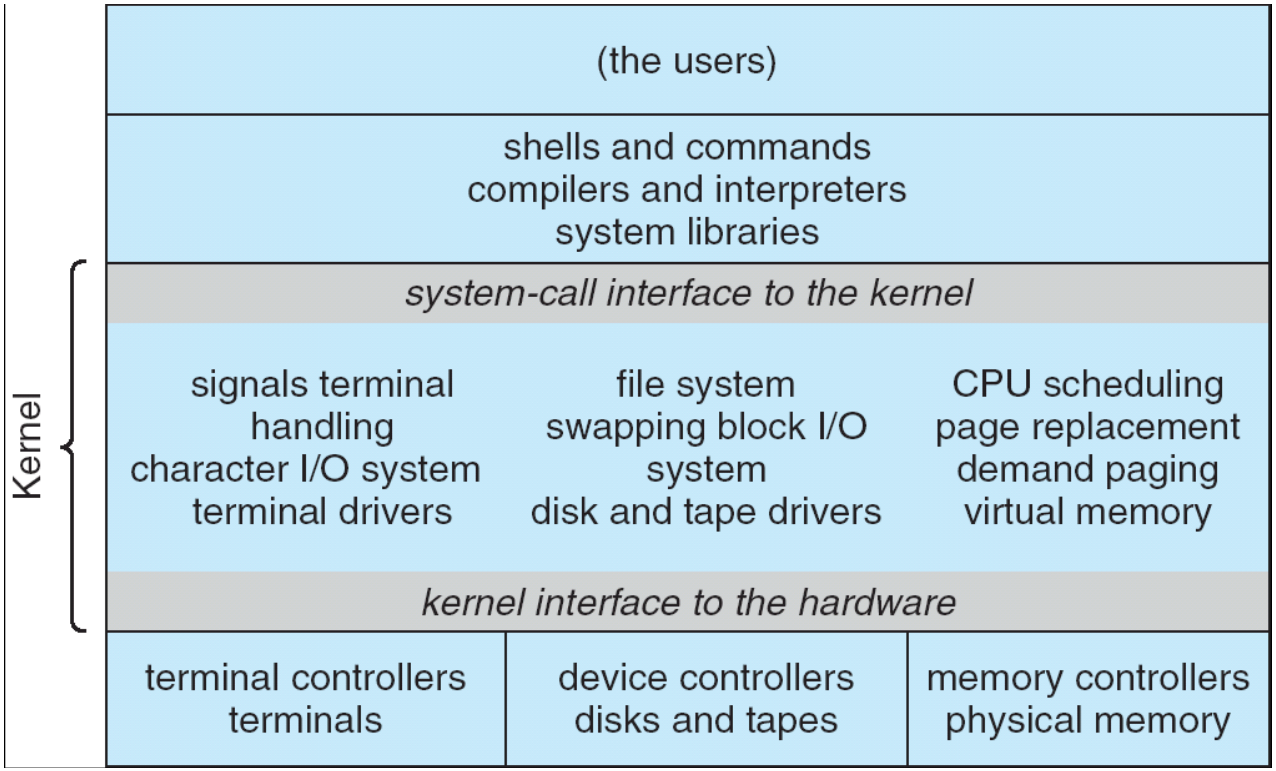
Técnica em camadas

- O sistema operacional é dividido em uma série de camadas (níveis), cada uma montada sobre camadas inferiores. A camada mais baixa (camada 0) é o hardware; a mais alta (camada N) é a interface com o usuário.
- Com a modularidade, as camadas são selecionadas de modo que cada uma use funções (operações) e serviços apenas de camadas de nível inferior

Sistema operacional em camadas



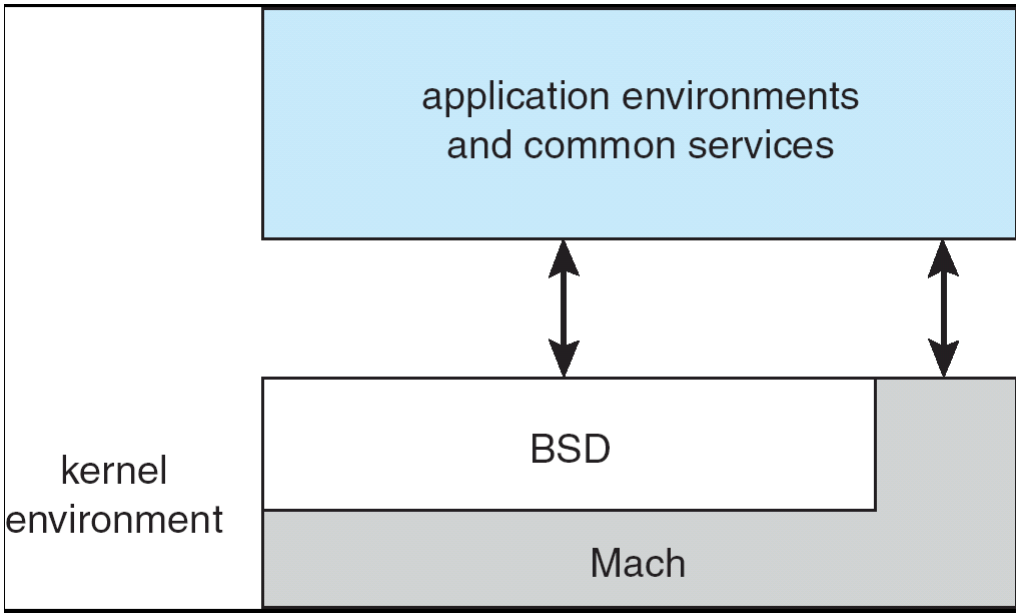
Estrutura do sistema UNIX



Estrutura do sistema de microkernel

- Move ao máximo do kernel para o espaço do "*usuário*"
- A comunicação ocorre entre os módulos do usuário usando a passagem de mensagens
- Benefícios:
  - Mais fácil de estender um microkernel
  - Mais fácil de portar o sistema operacional para novas arquiteturas
  - Mais confiável (menos código está executando no modo kernel)
  - Mais seguro
- Detrimentos:
  - Overhead de desempenho da comunicação entre espaço do usuário e espaço do kernel

Estrutura do Mac OS X



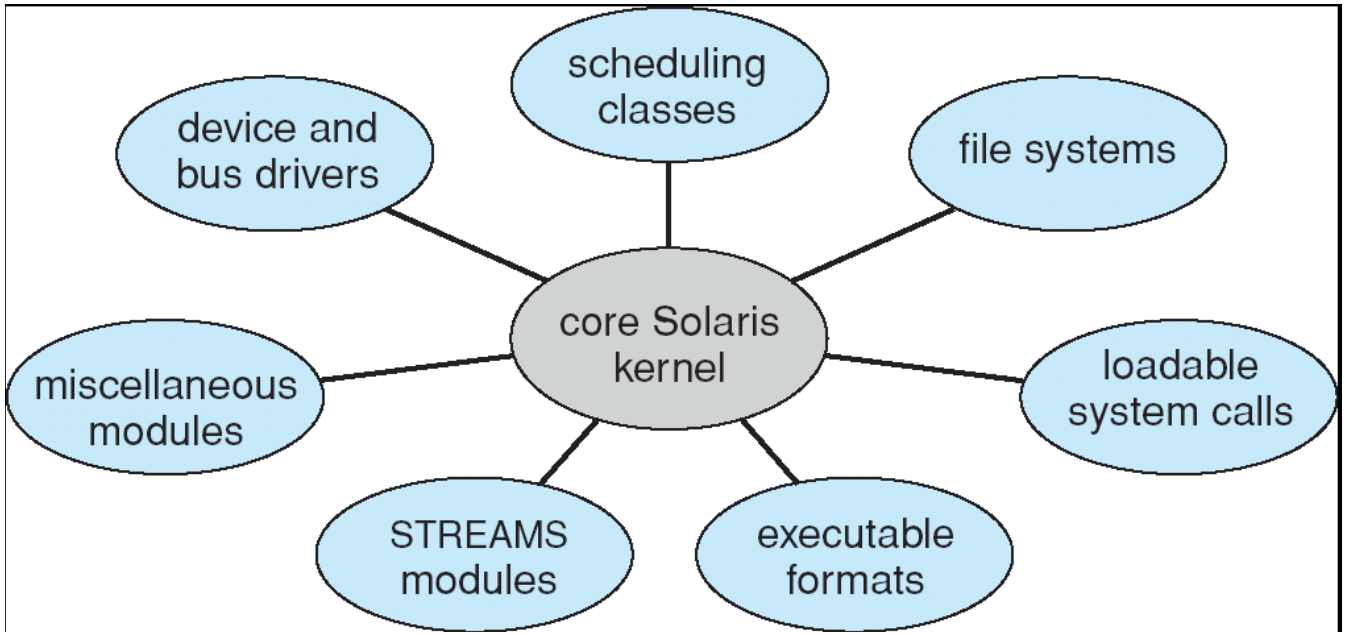
* Berkeley Software Distribution (**BSD**) series of UNIX

** **Mach** is an operating system kernel developed at Carnegie-Mellon University to support operating system research, primarily distributed and parallel computation.

Módulos

- A maioria dos sistemas operacionais modernos implementa módulos do kernel
  - Usa a técnica orientada a objeto
  - Cada componente do núcleo é separado
  - Cada um fala com os outros por interfaces conhecidas
  - Cada um é carregável conforme a necessidade dentro do kernel
- Em geral, semelhante a camadas, mas com mais flexibilidade

Técnica modular do Solaris



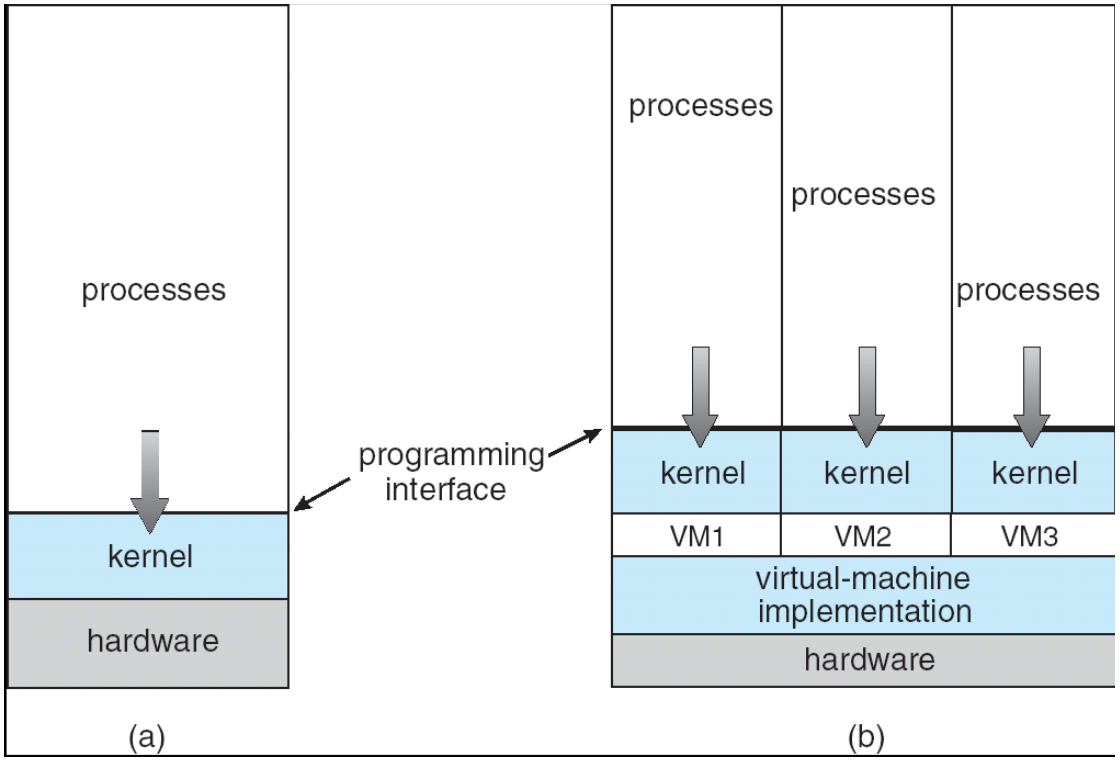
Máquinas virtuais

- Uma *máquina virtual* trata o hardware e o kernel do sistema operacional como se fossem tudo hardware
- Uma máquina virtual oferece uma interface *idêntica* ao hardware básico
- O sistema operacional cria a ilusão de múltiplos processos, cada um executando em seu próprio processador com sua própria memória (virtual)

Máquinas virtuais (cont.)

- Os recursos do computador físico são compartilhados para criar as máquinas virtuais
  - O escalonamento de CPU pode parecer que os usuários têm seu próprio processador
  - O *spooling* (permite aos programas entregar o trabalho a ser feito pelo periférico e, em seguida, avançar para outras tarefas) e um sistema de arquivos podem oferecer leitores de cartão virtual e impressoras de linha virtuais

Máquinas virtuais (cont.)



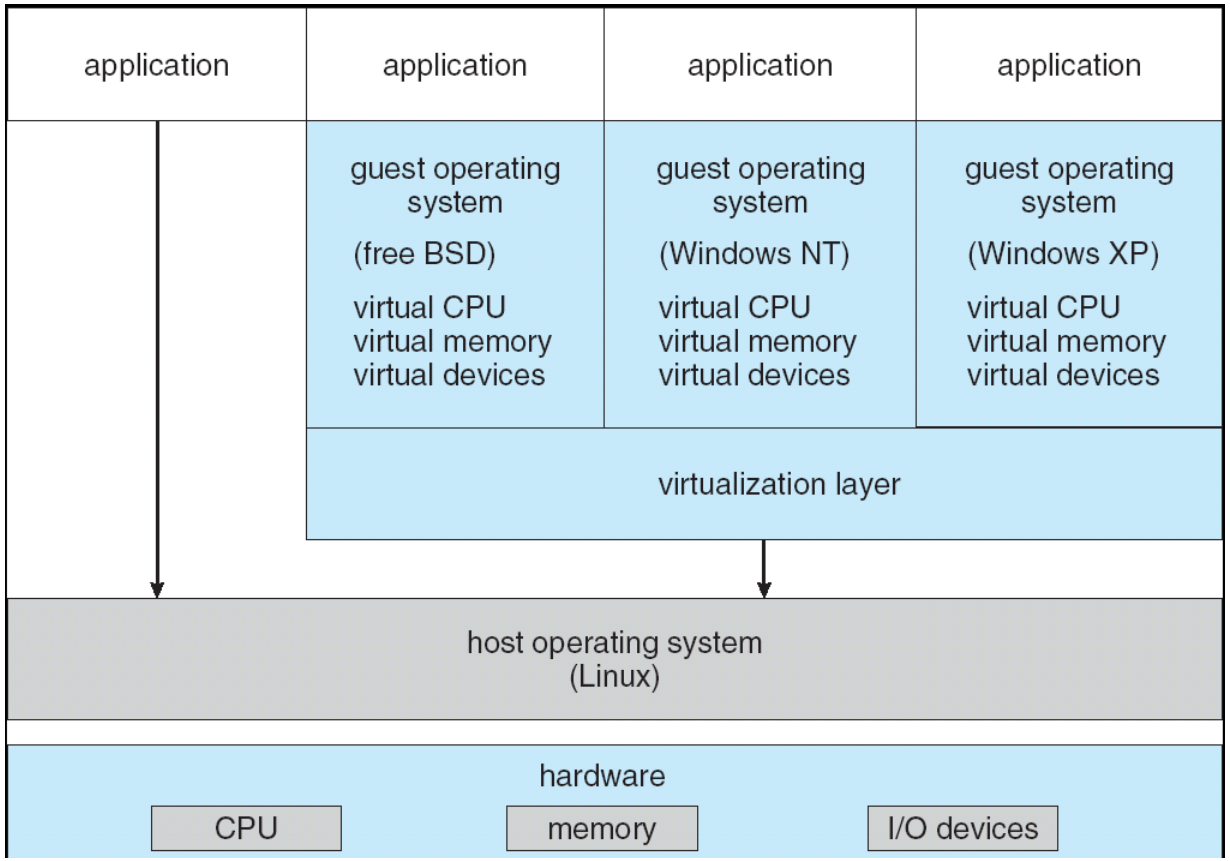
Máquina não virtual

Máquina virtual

Máquinas virtuais (cont.)

- O conceito de máquina virtual oferece proteção completa dos recursos do sistema, pois cada máquina virtual é isolada de todas as outras máquinas virtuais. Porém, esse isolamento não permite compartilhamento direto de 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, ao invés de uma máquina física, e por isso não atrapalha a operação normal do sistema.
- O conceito de máquina virtual é difícil de implementar, devido ao esforço exigido para fornecer uma duplicata *exata* à máquina básica.

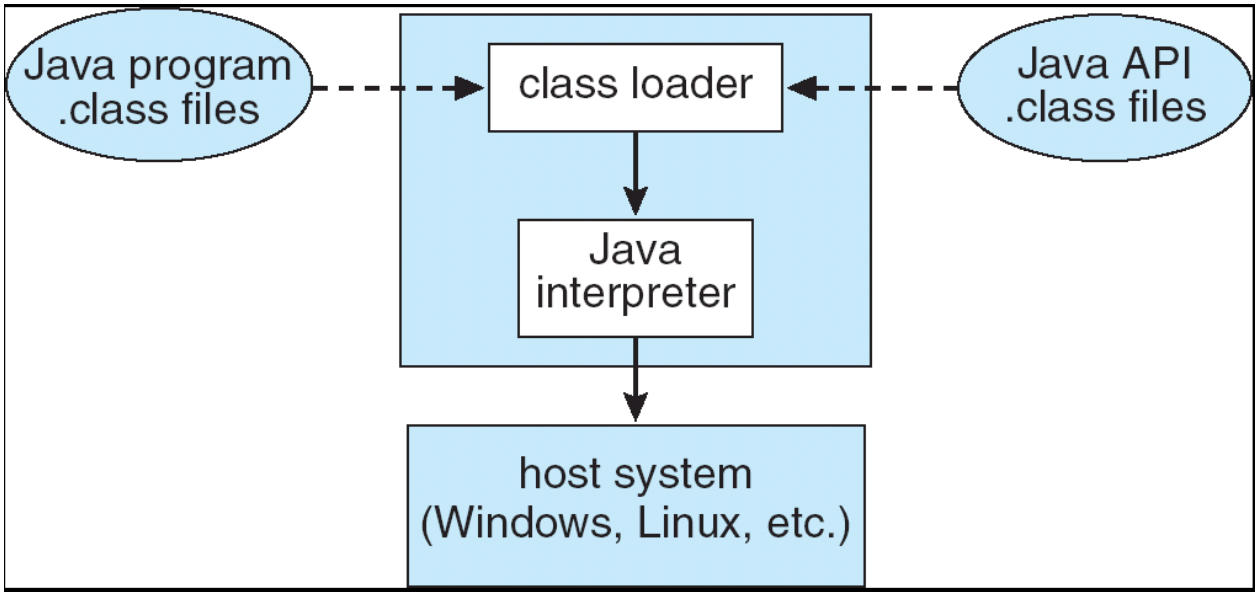
Arquitetura VMware



Java

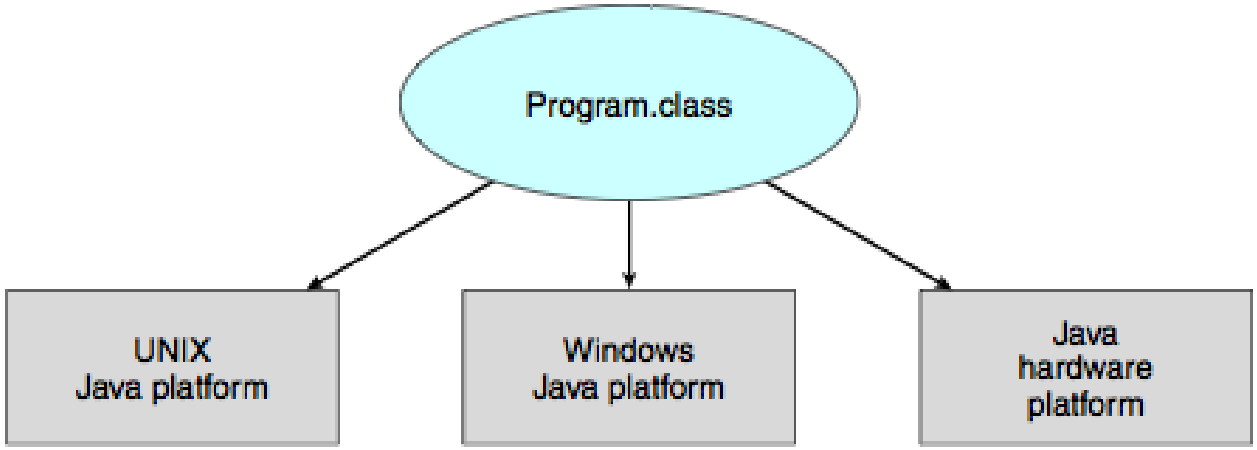
- Java consiste em
  1. Especificação de linguagem de programação
  2. Application Programming Interface (API)
  3. Especificação de máquina virtual

A Java Virtual Machine

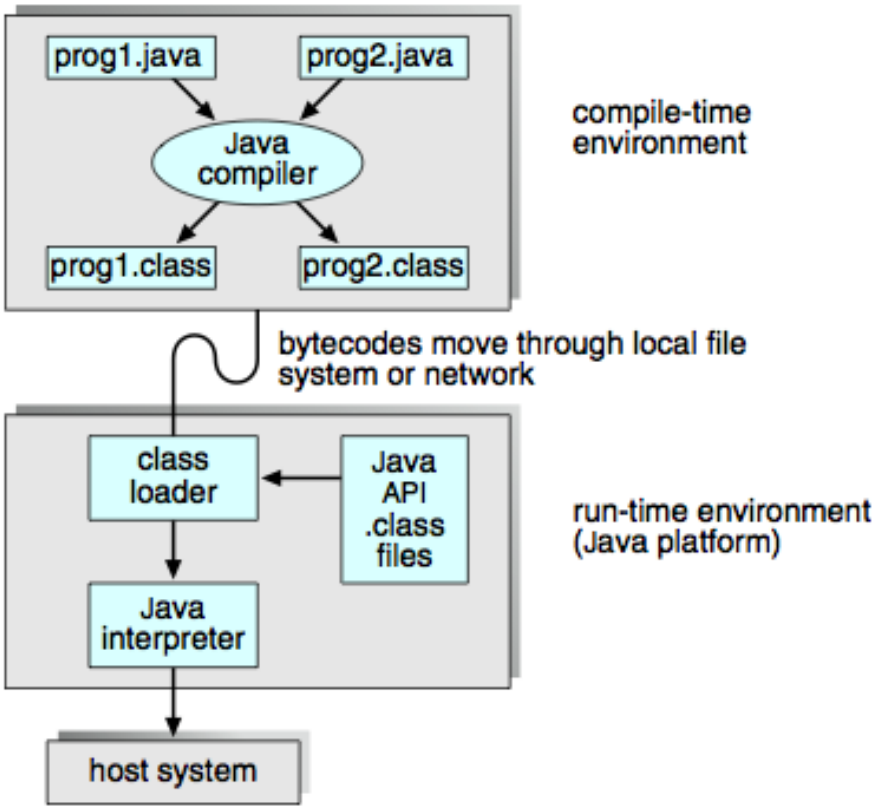


A Java Virtual Machine

Portabilidade da Java pelas plataformas.



O Java Development Environment

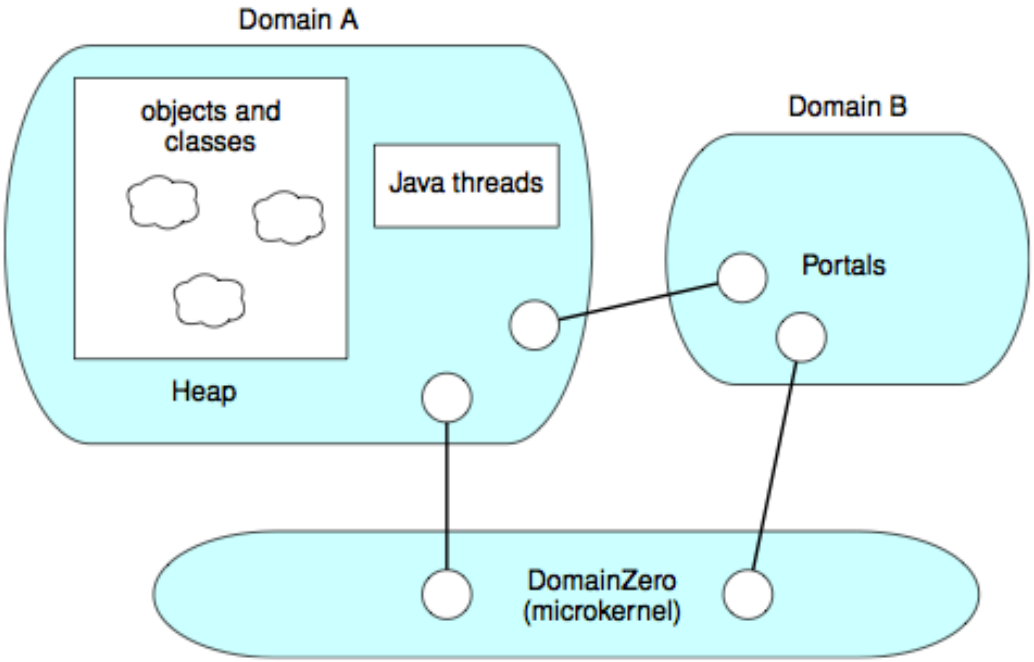


Sistemas operacionais Java

O sistema operacional JX

- 100% código java em todos os domínios, exceto no "zero"

- proteção pela linguagem e não pelo hardware (mais seguro, melhor desempenho, pois evita acionar mecanismos de controle do hardware)



Geração do sistema operacional

- Os sistemas operacionais são projetados para executar em qualquer uma de uma classe de máquinas; o sistema precisa ser configurado para cada computador específico
- O programa SYSGEN obtém informações referentes à configuração específica do sistema de hardware
- *Booting* – iniciar um computador carregando o kernel
- *Programa de bootstrap* – código armazenado na ROM que é capaz de localizar o kernel, carregá-lo na memória e iniciar sua execução

Boot do sistema

- O sistema operacional precisa estar disponível ao hardware, para que o hardware possa iniciá-lo
  - Pequeno trecho de código – carregador de bootstrap, localiza o kernel, carrega-o na memória e o inicia
  - Às vezes, um processo em duas etapas, onde o bloco de boot no local fixo carrega o carregador de bootstrap
  - Quando o sistema é inicializado, a execução começa em um local fixo da memória
    - Firmware usado para manter o código inicial de boot

Final do Capítulo 2

Silberschatz, Galvin e Gagne ©2007