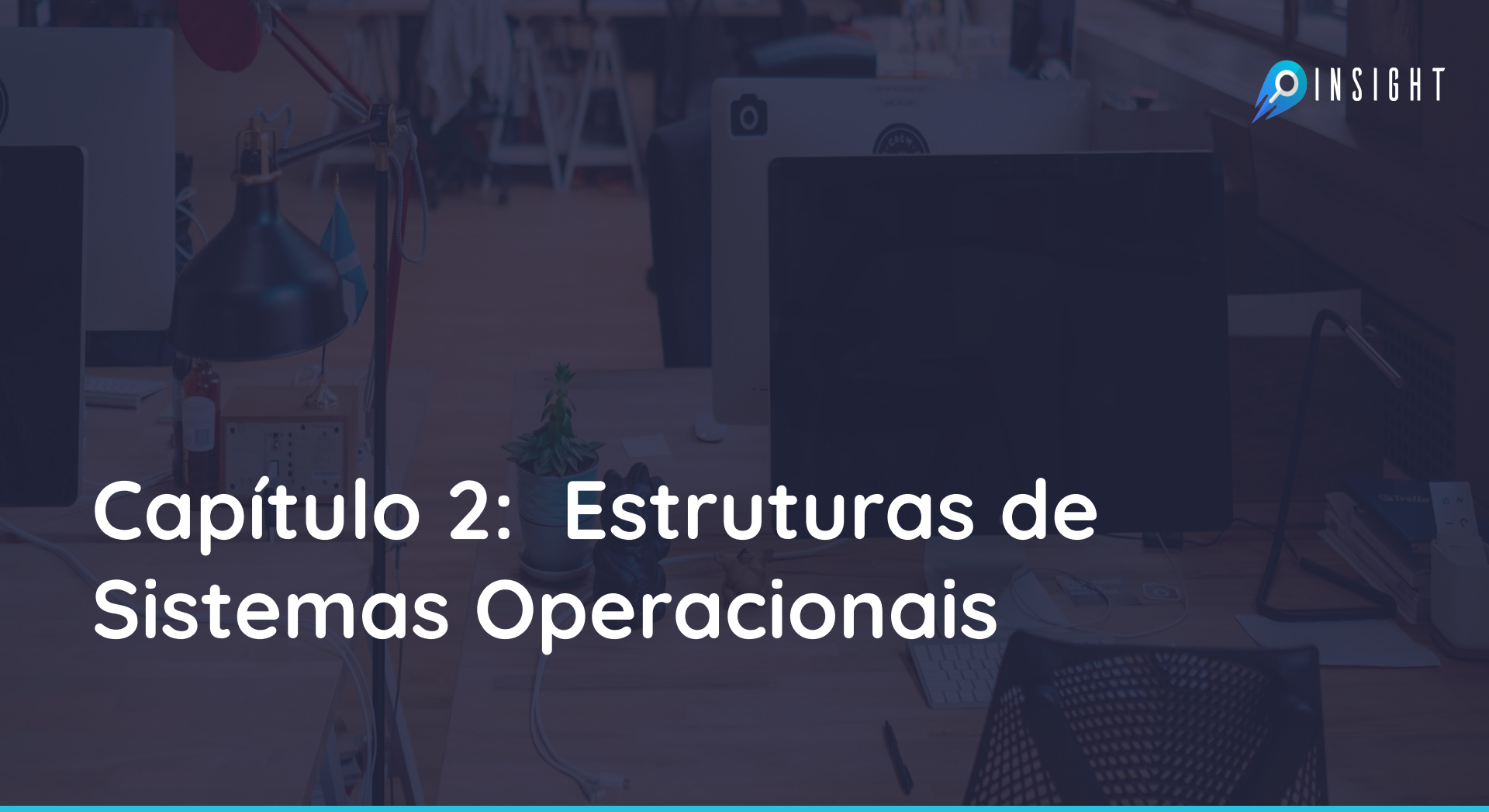


Capítulo 2: Estruturas de Sistemas Operacionais

Cap. 2 : Estruturas de SOs

- Servicos de Sistemas Operacionais
- Interface do SO com Usuario
- Chamadas ao Sistema
- Tipos de Chamadas ao Sistema
- Programas do Sistema
- Projeto e Implementação do SO
- Estrutura do SO
- Maquinas Virtuais
- Geração do SO
- Inicio (boot) do Sistema

Objetivos

- Descrever os serviços que um S.O. prove para os usuários, processos e outros sistemas
- Discutir as várias formas de estruturar um S.O.
- Explicar como sistema operacionais são instalados e como eles iniciam (boot)

Serviços do Sistema Operacional

- Um conjunto de serviços do S.O. provê funções que são úteis para o usuário:
  - **Interface com o usuário** – A maioria dos sistemas operacionais tem uma interface com o usuário (UI)
    - Variam entre linha de comando (CLI), Interface Gráfica com Usuário (GUI) e Processamento em lote de comandos (Batch)
  - **Execução de programas** – O sistema deve permitir a carga de um programa em memória, executar este programa, e finalizar sua execução, tanto normalmente como de forma anormal (indicando erro)
  - **Operações de E/S** - Um programa em execução pode requisitar E/S, a qual pode envolver um arquivo ou um dispositivo de E/S.
  - **Manipulação do sistema de arquivos** - O sistema de arquivo é de particular interesse. Obviamente, programas precisam ler e gravar arquivos e diretórios, criando e eliminando-os, procurando eles, apresentando informações sobre os mesmos e gerenciando suas permissões.

Serviços do Sistema Operacional (Cont.)

- Um conjunto de serviços do S.O. provê funções que são úteis para o usuário (Cont):
  - **Comunicações** – Processos podem trocar informações, no mesmo computador ou através da rede
    - Comunicação pode ser via memória compartilhada ou através passagem de mensagens (pacotes movidos pelo S.O.)
  - **Detecção de erro** – S.O. precisa estar consciente dos possíveis erros
    - Podem ocorrer na CPU e memoria, nos dispositivos de E/S, no programa do usuario
    - Para cada tipo de erro, o S.O. deve escolher a ação apropriada para garantir a computação correta e consistente
    - Facilidades de depuração podem ajudar bastante em enriquecer as habilidades dos usuários e programadores no uso eficiente do sistema

Serviços do Sistema Operacional (Cont.)

- Funções do S.O. que existem para garantir a operação eficiente do próprio sistema via recurso compartilhado
  - **Alocação de Recurso –** Quando múltiplos usuários ou múltiplos jobs executam concorrentemente, recursos devem ser alocados para cada um deles
    - Vários tipos de recursos – Alguns (do tipo ciclo de CPU, memória principal, e armazenamento de arquivo) podem ter código de alocação especial, outros (tais como dispositivos de E/S) podem ter código de requisição e liberação gerais.
  - **Contabilidade –** Para manter o rastro de quais usuarios usam o quanto e quais tipos de recursos do computador
  - **Protecao e seguranca -** Os donos da informação armazenada em um computador multiusuário ou de rede, podem desejar controlar uso desta informação, processos concorrentes não podem interferir uns nos outros
    - **Proteção –** envolve a garantia de que todos os acessos aos recursos do sistema são controlados
    - **Segurança** do sistema requer autenticação do usuário, estendido para defesa dos dispositivos de E/S das tentativas de acesso invalido
    - Se um sistema deve ser protegido e seguro, precauções devem ser instituídas.
    - ***Uma corrente é tão forte quanto seu elo mais fraco.***

Interface do SO com o usuario - CLI

CLI (command line interface) permite a entrada direta do usuário

- Algumas vezes implementada no kernel, outras vezes pelos programas do sistema
- Algumas vezes múltiplos sabores são implementados – conhecidos como **shells**
- Primariamente recupera um comando do usuário e executa-o
  - Algumas vezes comandos são embutidos, outras vezes são apenas nomes de programas
    - No último caso, a adição de novas funcionalidades não requer modificação do shell

Interface do SO com o usuario - GUI

- Interface que usa a metafora da area de trabalho (desktop)
  - Usualmente mouse, teclado, e monitor
  - **Icones** representam arquivos, programas, ações, etc
  - Vários botões do mouse sobre objetos da interface causam várias ações (provê informações, opções, executa funções, abre arquivos (conhecido como pasta)
  - Inventado pela Xerox PARC
- Vários sistemas agora incluem tanto interfaces CLI como interfaces GUI
  - Microsoft Windows é GUI com CLI usando shell
  - Apple Mac OS X usa interface GUI com kernel UNIX e shells disponiveis
  - Solaris é CLI com uso opcional de interfaces GUI (Java Desktop, KDE)

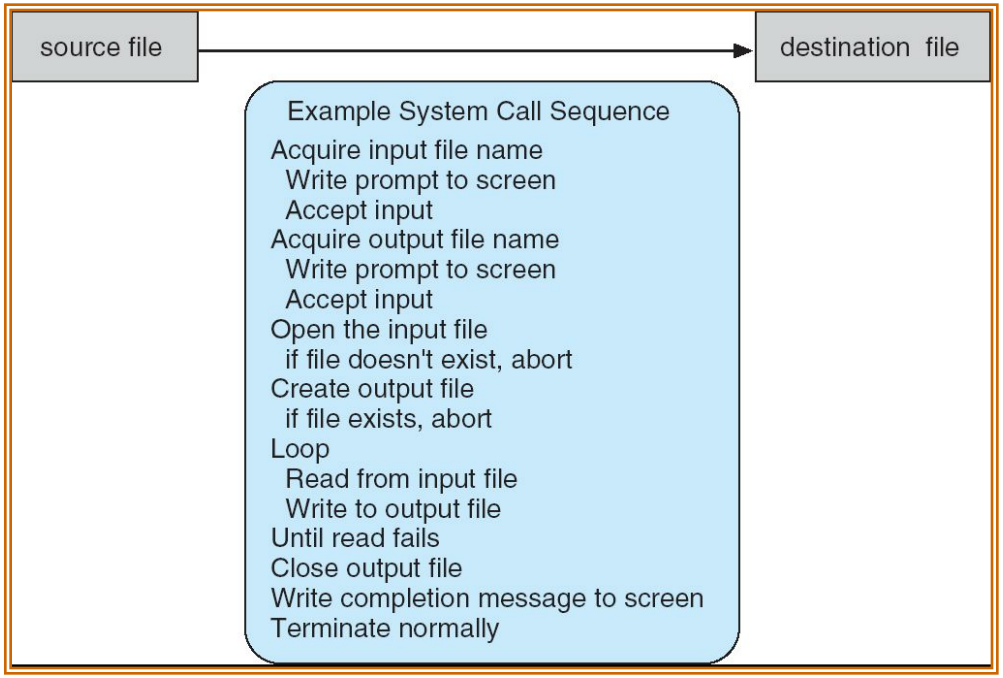
CHAMADAS AO SISTEMA

Chamadas ao sistema

- Interface de programação para serviços providos pelo S.O.
- Tipicamente escritas em uma linguagem de alto-nível (C ou C++)
- Em sua maioria acessadas por programas via uma interface de aplicação (API) de alto-nível do que chamadas diretas ao sistema
- As 3 mais conhecidas APIs são: **Win32 API** para Windows, **POSIX API** para sistemas baseados em POSIX (incluindo todas as versões do UNIX, Linux, e Mac OS X), e **Java API** para máquina virtual Java (JVM)
- Por que usar APIs do que chamadas diretas ao sistema ?

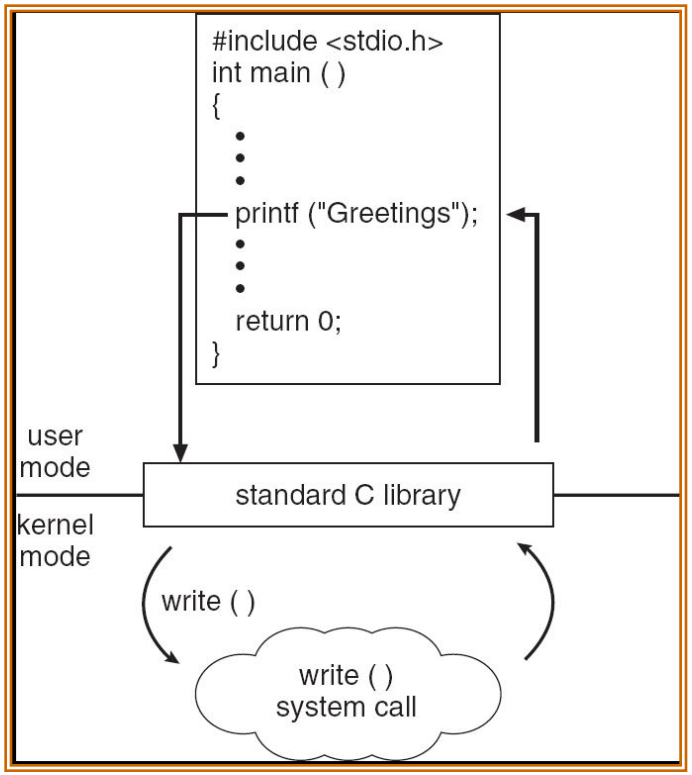
Exemplos de Chamadas ao Sistema

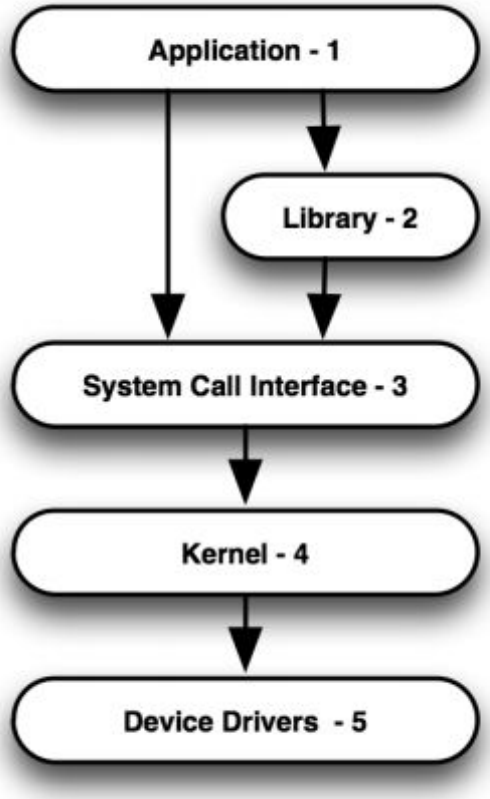
- Sequência de chamada ao sistema para copiar o conteúdo de um arquivo para outro



Exemplo da biblioteca padrão C

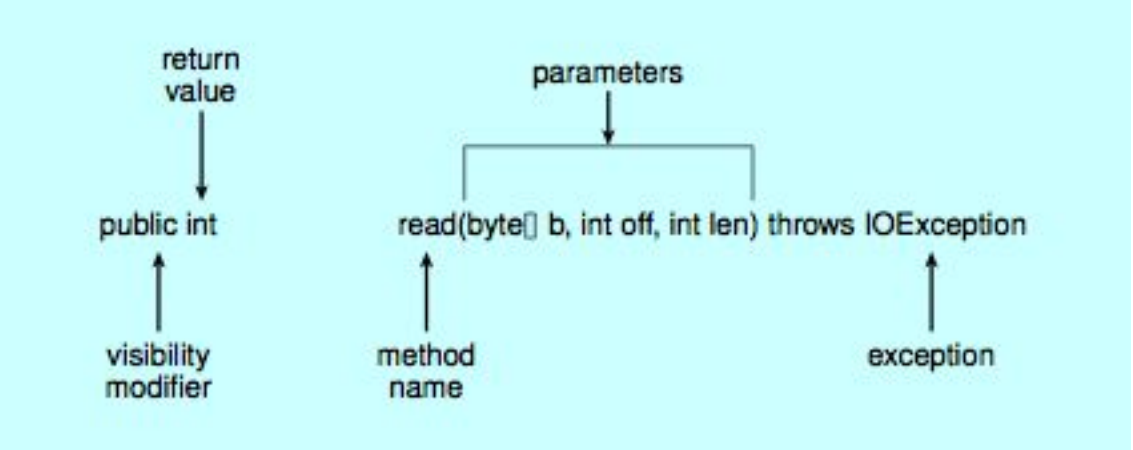
- Programa C invocando printf(), o qual chama a system call write()





Exemplo de uma API padrão

- Considere a funcao read() do Java



byte[] b – o buffer no qual o dado é lido
int off – o offset inicial em b onde o dado é lido
int len – o número máximo de bytes para ler

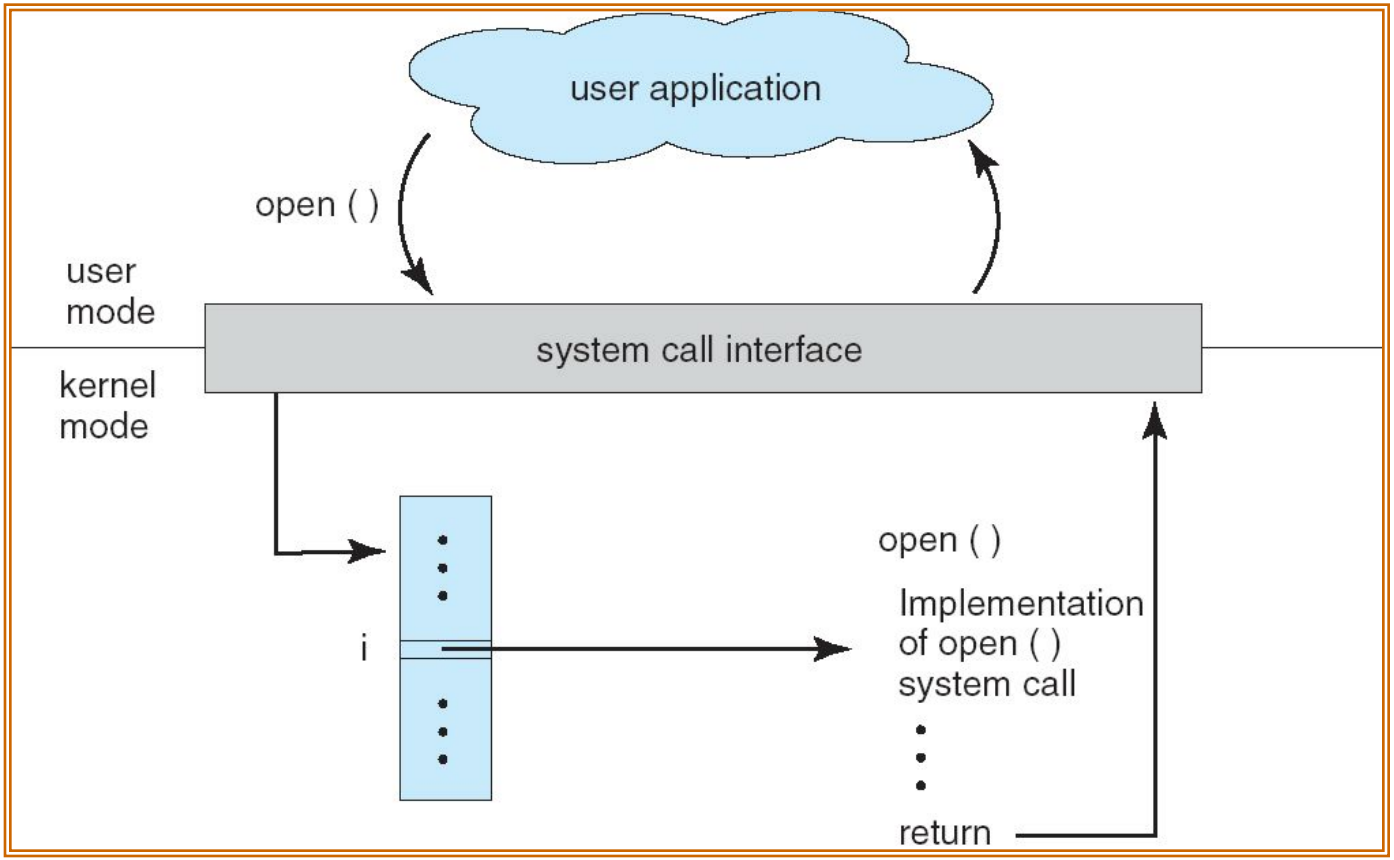
Solaris 10 dtrace rastreando System Call

```
# ./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
```

Implementação de Chamada ao Sistema

- Tipicamente, um número associado com cada chamada ao sistema
  - Interface de chamada ao sistema mantém uma tabela indexada de acordo com esses números
- Uma interface de chamada ao sistema invoca chamada do sistema no kernel do S.O. e retorna o estado da chamada e alguns valores de retorno
- O chamador não precisa saber sobre como a chamada ao sistema é implementada
  - Só precisa obedecer a API e entender o que o S.O. retornará como resultado a chamada
  - A maioria dos detalhes da interface do SO são escondidas do programador pela API
    - Gerenciada pela biblioteca de suporte em tempo de execução (conjunto de funções incorporadas nas bibliotecas incluídas no compilador)

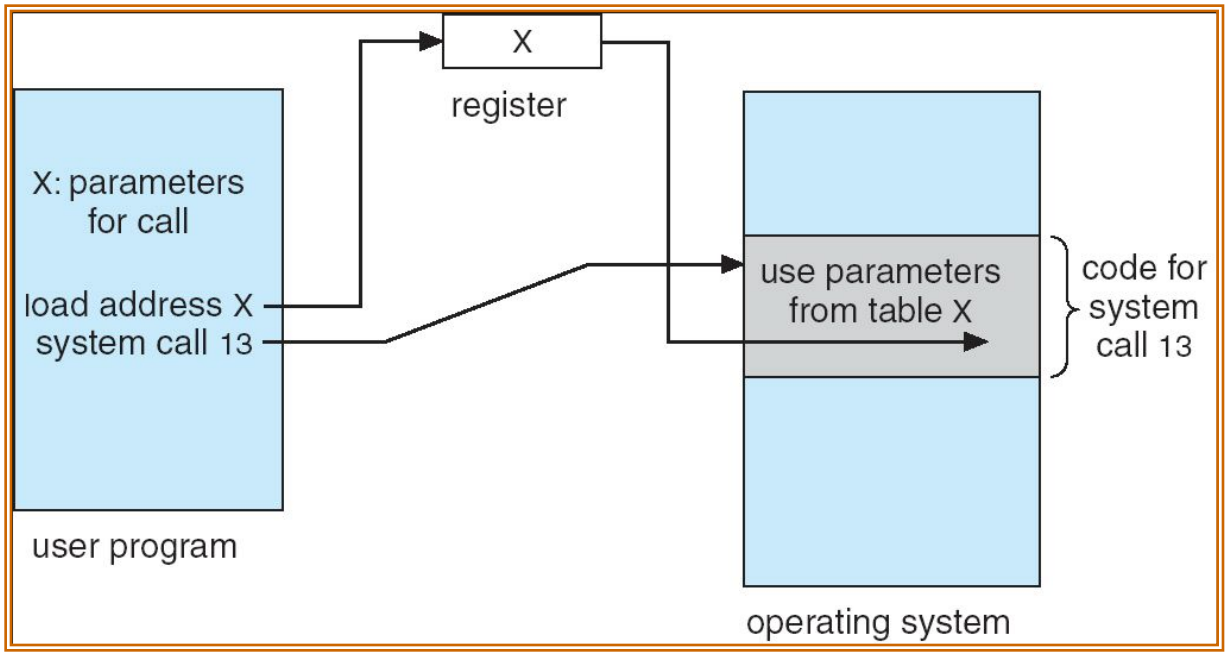
Relacionamento da API de chamadas ao sistema com S.O.



Passagem de parametro a System Call

- Em geral, mais informações são necessárias do que simplesmente a identidade da system call
  - Tipo exato e quantidade de informação variam de acordo com o SO e a chamada
- Tres metodos usados para passar parametros para o SO
  - Mais simples: passando os parâmetros em registradores
    - Em alguns casos, pode existir mais parâmetros do que registradores
  - Parâmetros armazenados em um bloco, ou tabela, em memória, e o endereço do bloco passado como parâmetro em um registrador
    - Esta abordagem é usada pelo Linux e Solaris
  - Parâmetros colocado em uma pilha pelo programa e recuperadas da pilha pelo SO
  - Métodos de bloco e pilha não limitam o número e tamanho dos parâmetros que estão sendo passados

Passagem de Parametro via Registrador

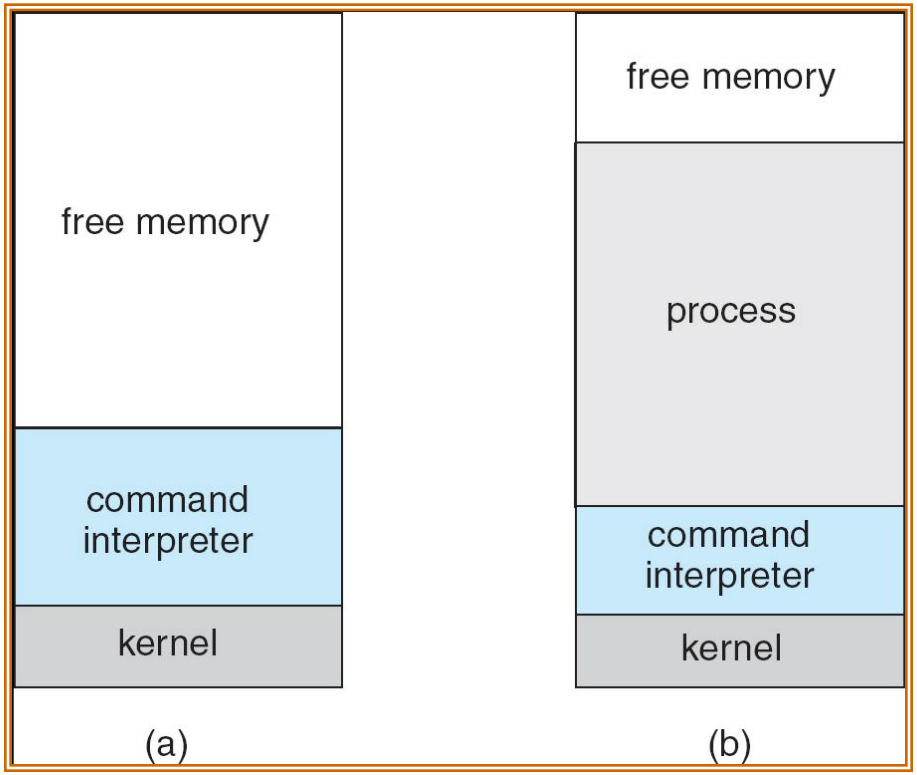


Tipos de System Calls

- Controle de Processo
- Gerenciamento de Arquivo
- Gerenciamento de Dispositivo
- Manutencao de Informação
- Comunicações
- etc

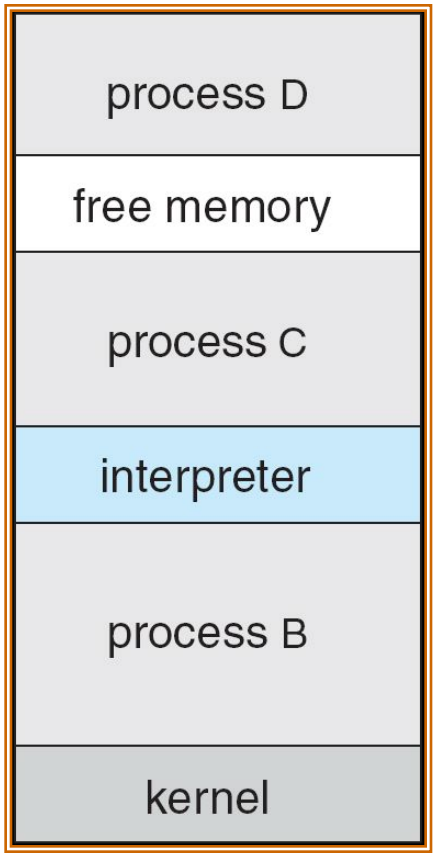
PROGRAMAS DO SISTEMA

Execucao MS-DOS



(a) At system startup (b) running a program

FreeBSD Executando Multiplos Programas



Programas do Sistema

- Provem um ambiente conveniente para desenvolvimento e execução de programas. Eles podem ser divididos em:
  - Manipulacao de Arquivo
  - Informacao sobre estado
  - Modificacao de Arquivo
  - Suporte a linguagem de programação
  - Carga e execucao de programa
  - Comunicacoes
  - Programas de Aplicacao
- A maior parte da visão dos usuários sobre o SO é definida pelos programas do sistema, não sobre as system calls

Programas do Sistema

- Provem um ambiente conveniente para desenvolvimento e execução de programas.
  - Alguns deles são simples interfaces com usuário para as system calls; outros são consideravelmente mais complexos
- Gerenciamento de arquivos – criar, eliminar, copiar, renomear, imprimir, dump, listar, e geralmente manipular arquivos e diretórios
- Informacao de estado
  - Alguém pergunta sobre uma informação – data, hora, memória disponível, espaço em disco, número de usuários
  - Outros provem informações detalhadas sobre performance, logging, e depuracao
  - Tipicamente, esses programas formatam e imprimem a saída para terminal ou outros dispositivos de saída
  - Alguns sistemas implementam um registry – usado para armazenar e recuperar informação sobre configuração

Programas do Sistema (cont)

- Modificacao de Arquivo
  - Editores de texto para criar e modificar arquivos
  - Comandos especiais para pesquisar conteúdo dos arquivos ou para executar transformações no texto
- Suporte a linguagem de programação – Compiladores, assemblers, debuggers, e interpretadores algumas vezes são providos
- Carga e execução de programas – Carregadores absolutos, carreadores realocáveis, editores de linkage, e carregadores de overlay, sistemas de depuração para alto-nível e linguagem de máquina
- Comunicações – Prover os mecanismos para criação virtual de conexoes entre processos, usuarios, e sistemas de computador
  - Permite usuários enviar mensagens para outra telas, navega em paginas web, enviar email eletronico, logar remotamente, transferir dados de uma máquina para outra

PROJETO DE SISTEMA OPERACIONAL

Projeto e Implementação de Sistema Operacional

- Projeto e Implementação de SO não foi solucionada, mas algumas abordagens foram aprovadas com sucesso
- Estrutura Interna de diferentes Sistemas Operacionais podem variar enormemente
- Iniciar definindo objetivos e especificações
- Afetada pela escolha do hardware, tipo de sistema
- *Objetivos dos usuarios e objetivos do sistema*
  - Objetivos dos Usuarios – SO deve ser conveniente para usar, facil para aprender, confiavel, seguro, e rapidos
  - Objetivos do Sistema – SO deve ser fácil de projetar, implementar, e manter, assim como flexivel, confiável, sem erros e eficiente

Projeto e Implementação de Sistema Operacional (Cont.)

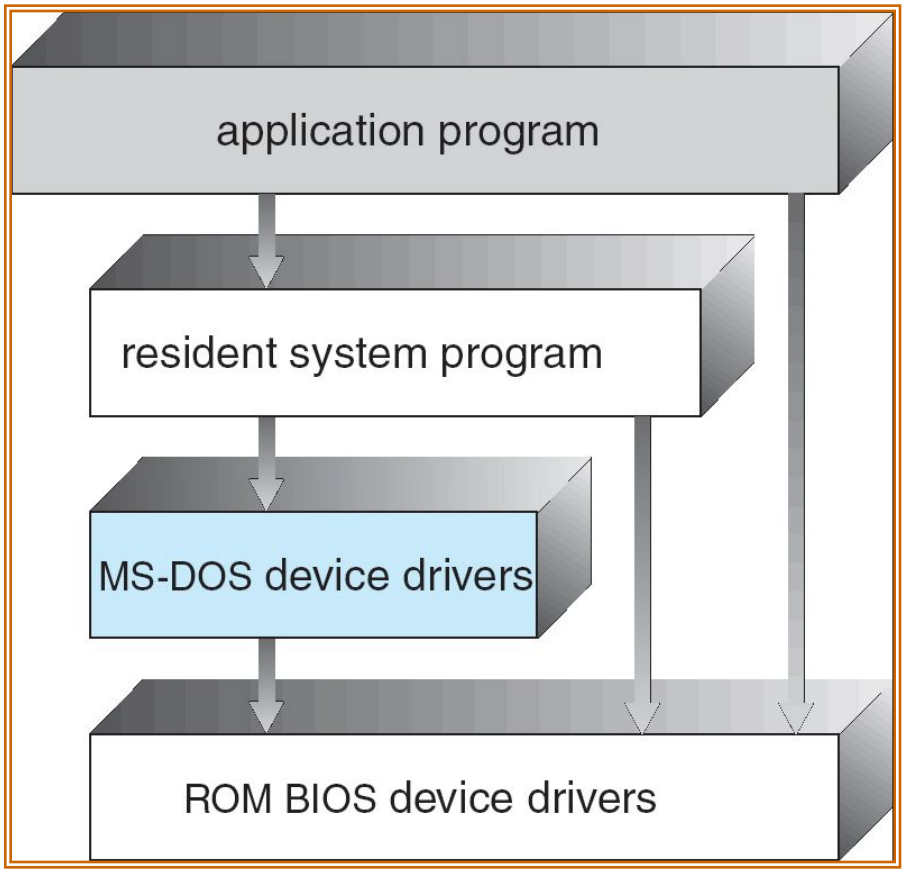
- Principio importante para separar

  Politica: O que deverá ser feito?
  Mecanismo: Como fazer?
- Mecanismos determinam como fazer alguma coisa, políticas decidem o que será feito
  - A separação da política do mecanismo é um princípio muito importante, isto permite a flexibilidade máxima se as decisões da política podem ser alteradas mais tarde.

Estrutura simples

- MS-DOS – escrito para prover o maximo de funcionalidade no menor espaco possivel
  - Não é dividido em módulos
  - Embora MS-DOS tenha alguma estrutura, suas interfaces e níveis de funcionalidade não são bem separadas

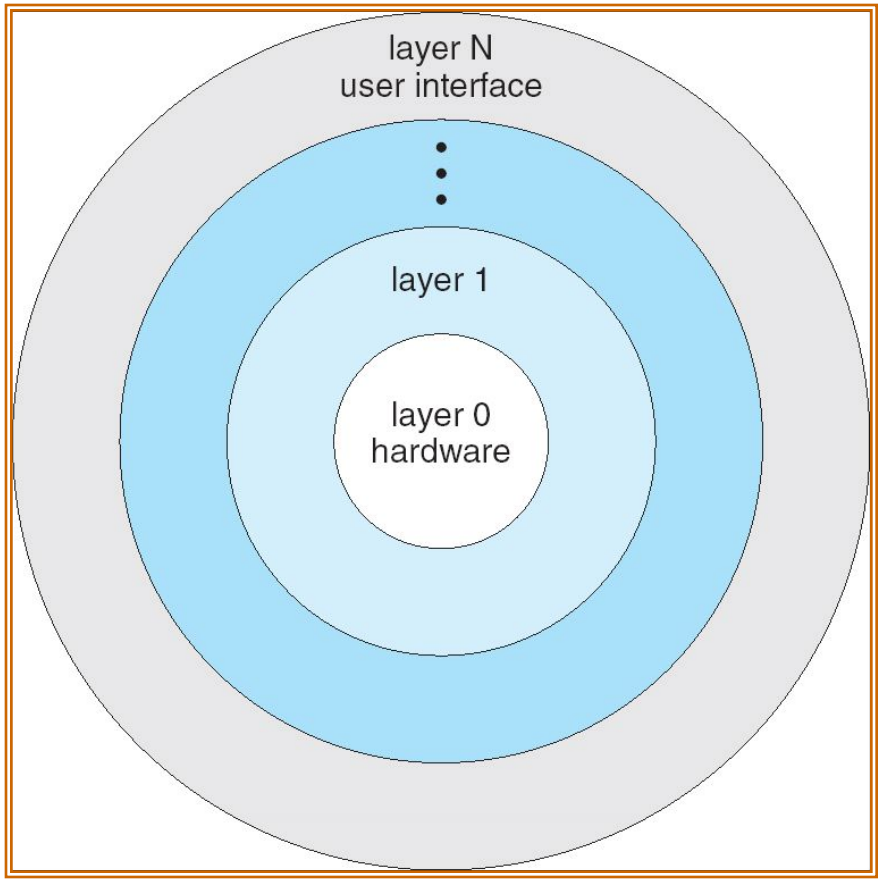
Estrutura de Camada do MS-DOS



Abordagem em camadas

- O SO é dividido em um número de camadas (niveis), cada uma construído sobre as camadas mais abaixo. A camada inferior (camada 0), é o hardware; a camada mais alta é a interface do usuário.
- Com modularidade, camadas são selecionadas de tal forma que cada uma usa funções (operações) e serviços somente dos níveis inferiores.

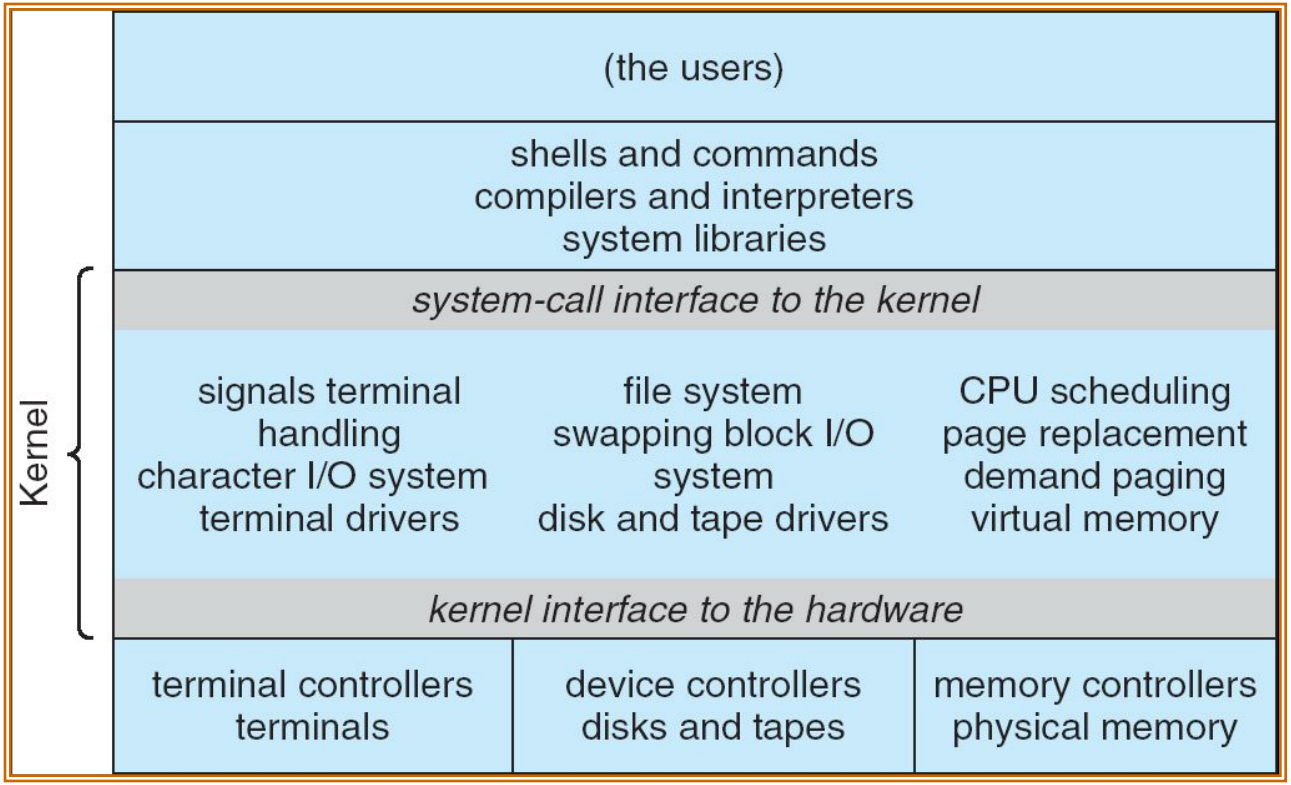
Sistema Operacional em Camadas



UNIX

- O UNIX consiste de duas partes separadas:
  - Programas do Sistema
  - O kernel
    - Consiste de tudo abaixo da interface de system-call e acima do hardware fisico
    - Prove o sistema de arquivo, escalonamento de CPU, gerenciamento de memória, e outras funções do sistema; um grande número de funções para um nível

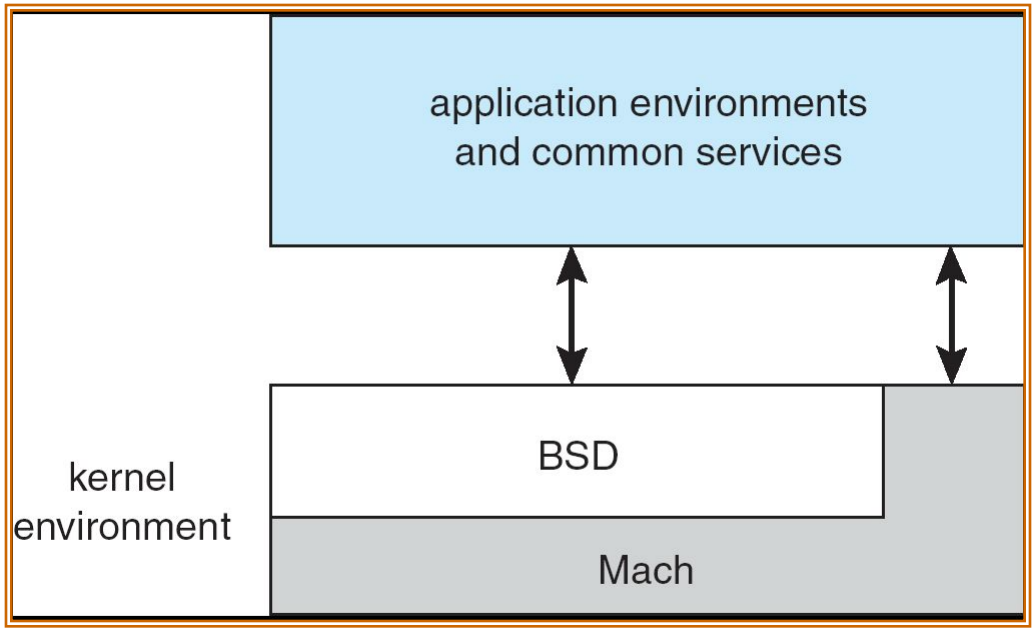
Estrutura do Sistema UNIX



Estrutura do Sistema de Microkernel

- Move o maximo do kernel para o espaco do usuario
- Comunicacao toma lugar entre modulos do usuario usando troca de mensagem
- Beneficios:
  - Facil para estender o microkernel
  - Facil de portar o SO para novas arquiteturas
  - Mais confiavel (menos codigo executando no modo kernel)
  - Mais seguro
- Detrimentos:
  - Sobrecarga de comunicacao do espaco do usuario para o espaco do kernel

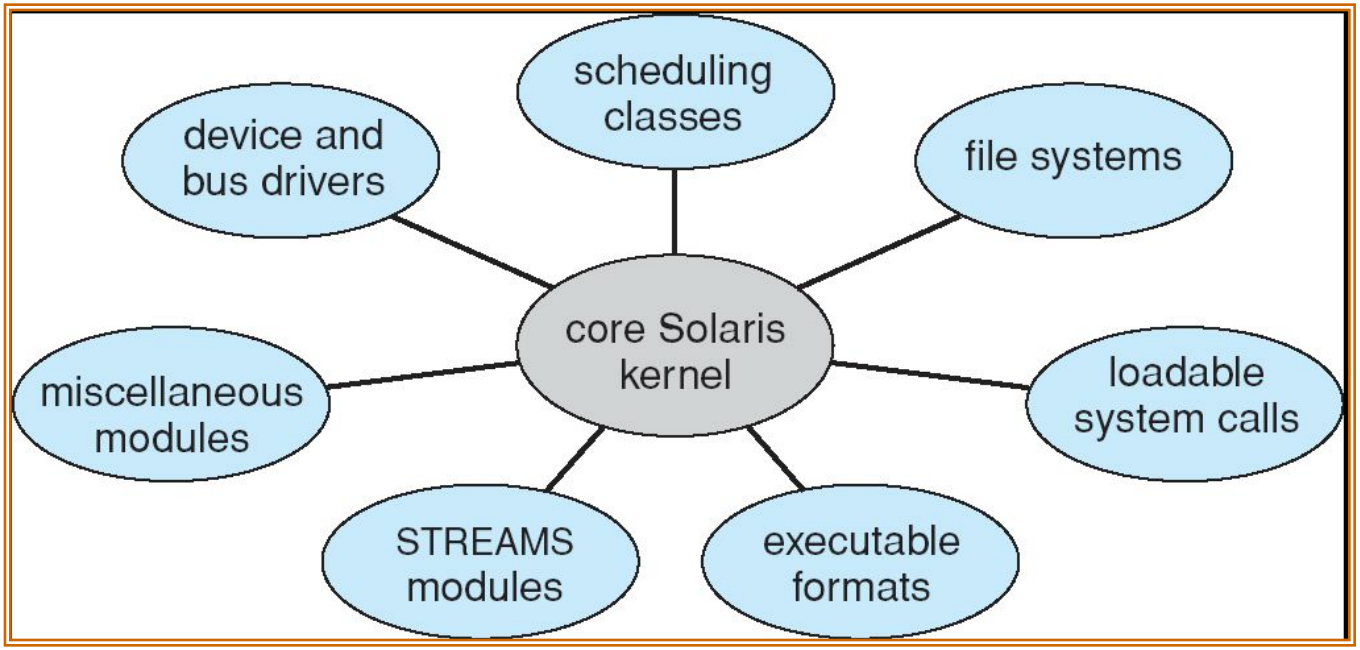
Estrutura do Mac OS X



Modulos

- A maioria dos SOs implementam modulos de kernel
  - Usam abordagem orientada a objetos
  - Cada componente é separado
  - Cada um fala com os outros atraves de interfaces conhecidas
  - Cada um é alocado conforme necessario dentro do kernel
- De forma geral, similar as camadas porem mais flexivel

Abordagem Modular Solaris



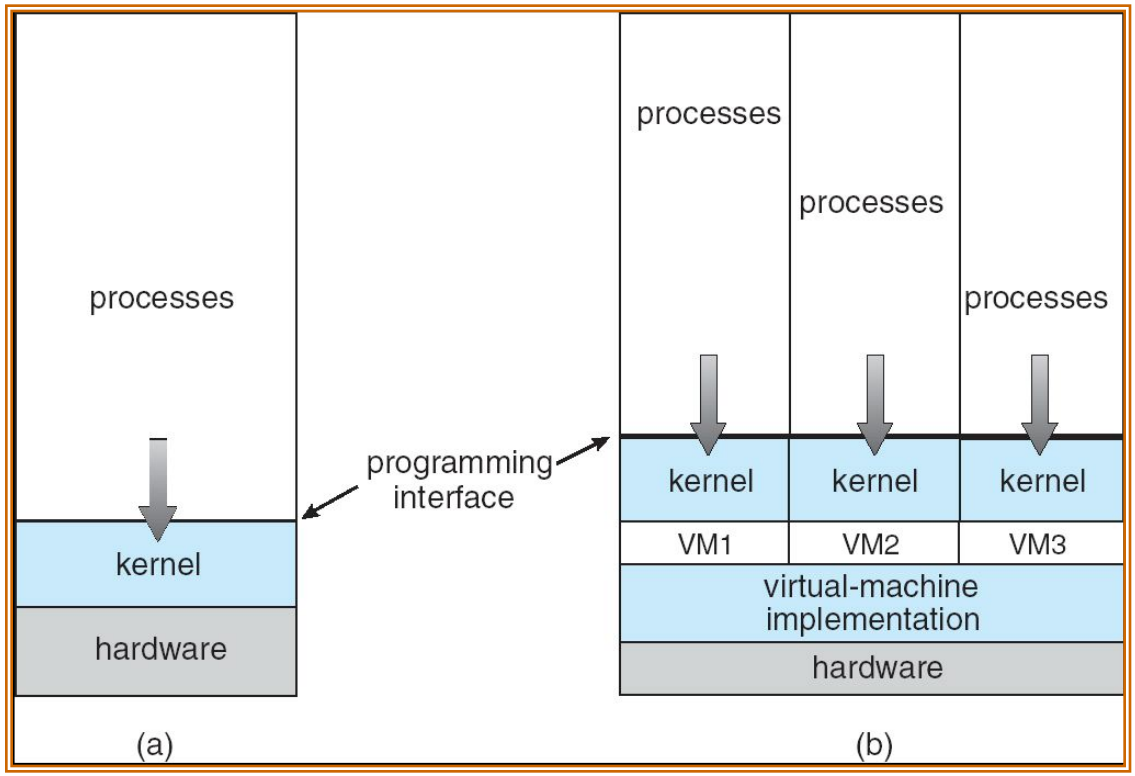
Maquinas Virtuais

- Uma máquina virtual adota uma abordagem em camadas na sua conclusão lógica. Ela trata o hardware e o kernel como se fossem todos softwares.
- Uma máquina virtual provê uma interface idêntica ao hardware subjacente desprotegido
- O SO cria a ilusão de múltiplos processos, cada um executando sobre seu próprio processador com sua própria memória virtual

Maquinas Virtuais (Cont.)

- Os recursos do computador fisico sao compartilhados para criar as maquinas virtuais
  - Escalonamento de CPU pode dar a aparencia de que usuarios tem seu proprio processador
  - Spooling e um sistema de arquivo pode prover leitores e impressoras virtuais
  - Um terminal de usuario de tempo-compartilhado normal serve como uma console do operador da maquina virtual

Maquinas Virtuais (Cont.)



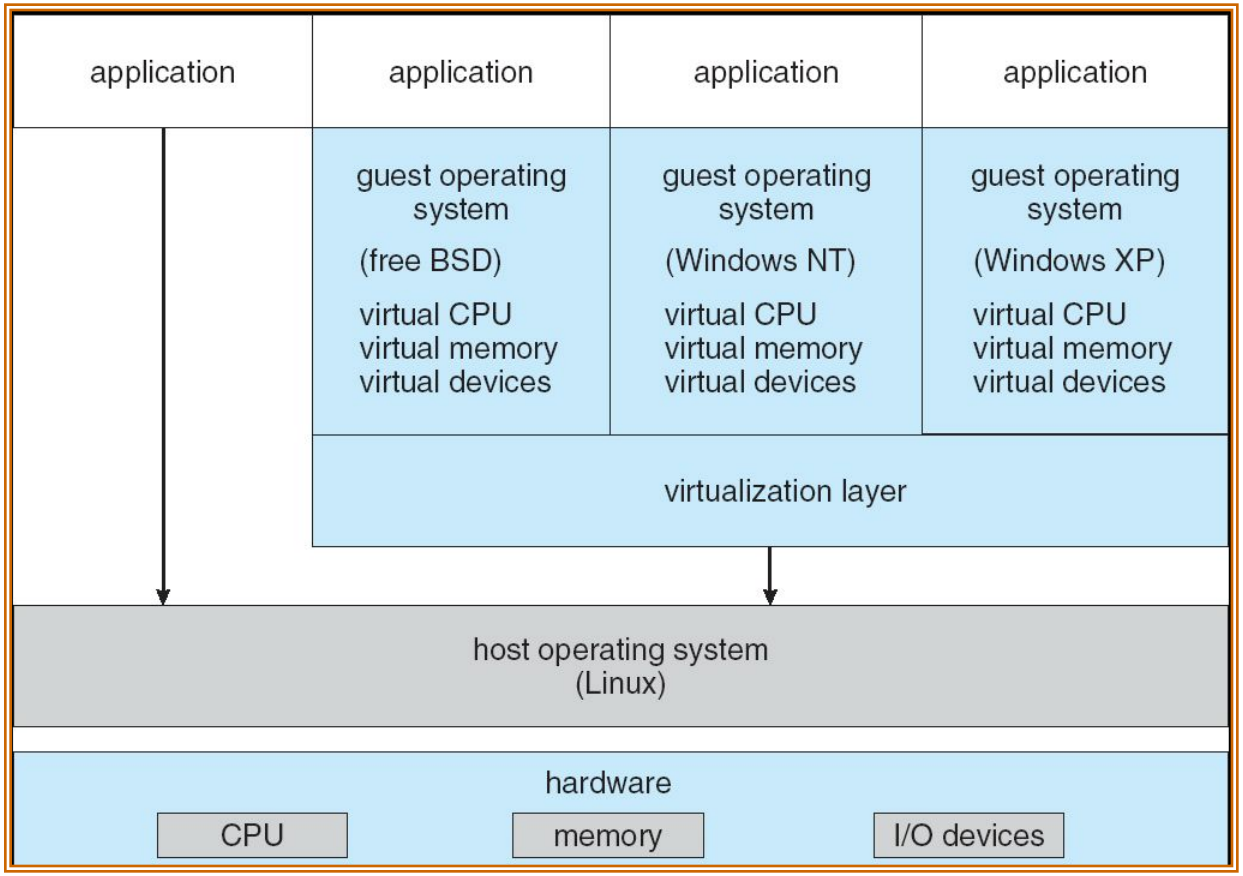
Non-virtual Machine

Virtual Machine

Virtual Machines (Cont.)

- The virtual-machine concept provides complete protection of system resources since each virtual machine is isolated from all other virtual machines. This isolation, however, permits no direct sharing of resources.
- A virtual-machine system is a perfect vehicle for operating-systems research and development. System development is done on the virtual machine, instead of on a physical machine and so does not disrupt normal system operation.
- The virtual machine concept is difficult to implement due to the effort required to provide an *exact* duplicate to the underlying machine

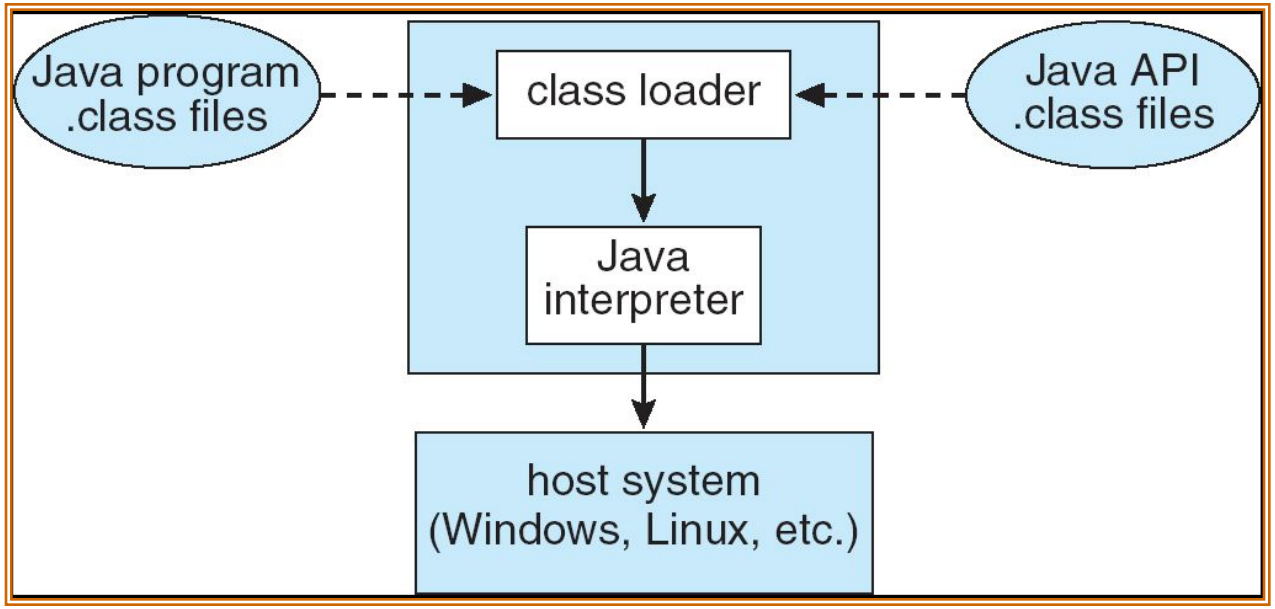
VMware Architecture



Java

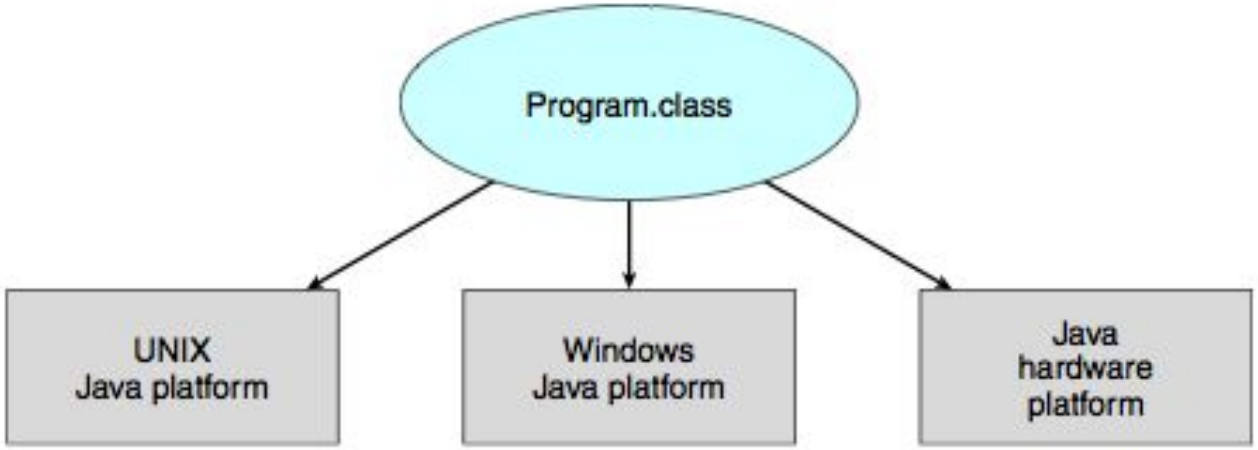
- Java consiste em
  1. Especificação de Linguagem de Programação
  2. Application programming interface (API)
  3. Especificacao de uma maquina virtual

The Java Virtual Machine

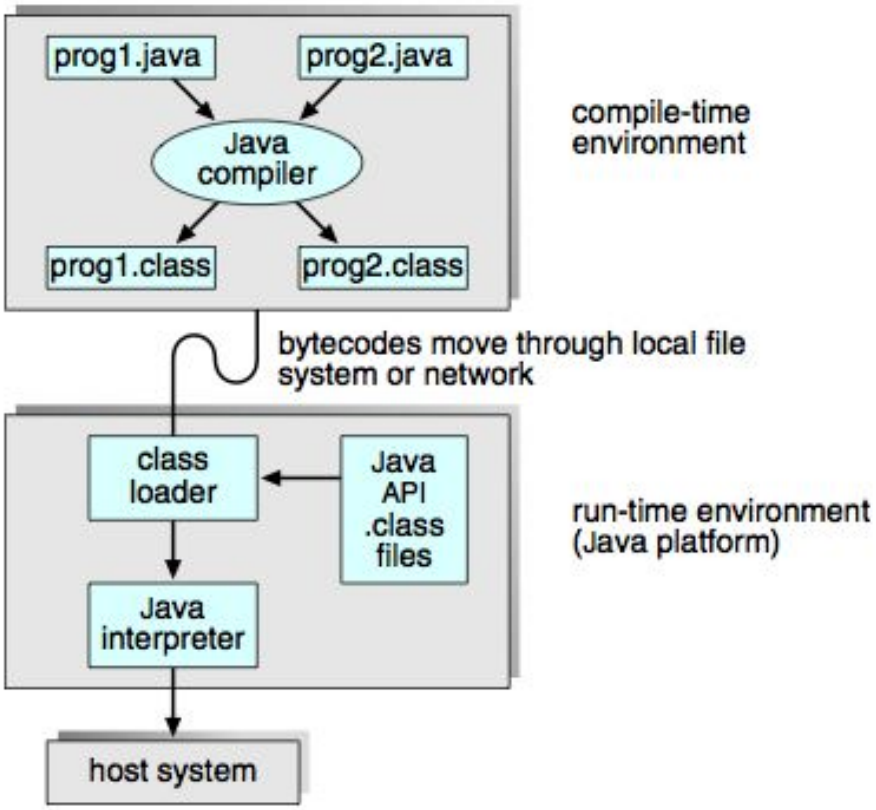


A maquina virtual Java

Portabilidade Java entre plataformas



O Ambiente de desenvolvimento Java



GERAÇÃO DE SISTEMAS OPERACIONAIS

Geracao de Sistema Operacional

- Sistemas operacionais são projetados para executar qualquer classe de máquinas; o sistema deve ser configurado para cada computador específico
- Programa SYSGEN obtem informacao que concerne a configuracao especifica do sistema de hardware
- *Booting* – iniciar um computador carregando o kernel
- *Bootstrap program* – código armazenado em ROM que é habilitado a localizar o kernel, carrega-lo na memória, e iniciar sua execução

System Boot

- SO deve estar disponivel para o hardware de tal forma que o hardware possa inicia-lo
  - Pequeno pedaco de codigo – **bootstrap loader**, localiza o kernel, carrega ele em memoria, e o inicia
  - Algumas vezes um processo em dois passos onde um **boot block** em uma localização fixa carrega o **bootstrap loader**
  - Quando o sistema é ligado, a execução inicia em uma localização fixa de memória
    - Firmware usado para guardar o codigo de boot inicial

Final do Capitulo 2