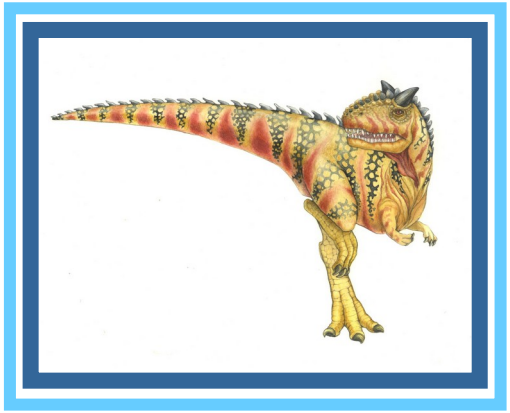
Capítulo 2: Estruturas de Sistema Operacional



Sobre a apresentação (About the slides)

Os slides e figuras dessa apresentação foram criados por Silberschatz, Galvin e Gagne em 2009. Esse apresentação foi modificada por Cristiano Costa (cac@unisinos.br). Basicamente, os slides originais foram traduzidos para o Português do Brasil.

É possível acessar os slides originais em http://www.os-book.com

Essa versão pode ser obtida em http://www.inf.unisinos.br/~cac

The slides and figures in this presentation are copyright Silberschatz, Galvin and Gagne, 2009. This presentation has been modified by Cristiano Costa (cac@unisinos.br). Basically it was translated to Brazilian Portuguese.

You can access the original slides at http://www.os-book.com

This version could be downloaded at http://www.inf.unisinos.br/~cac

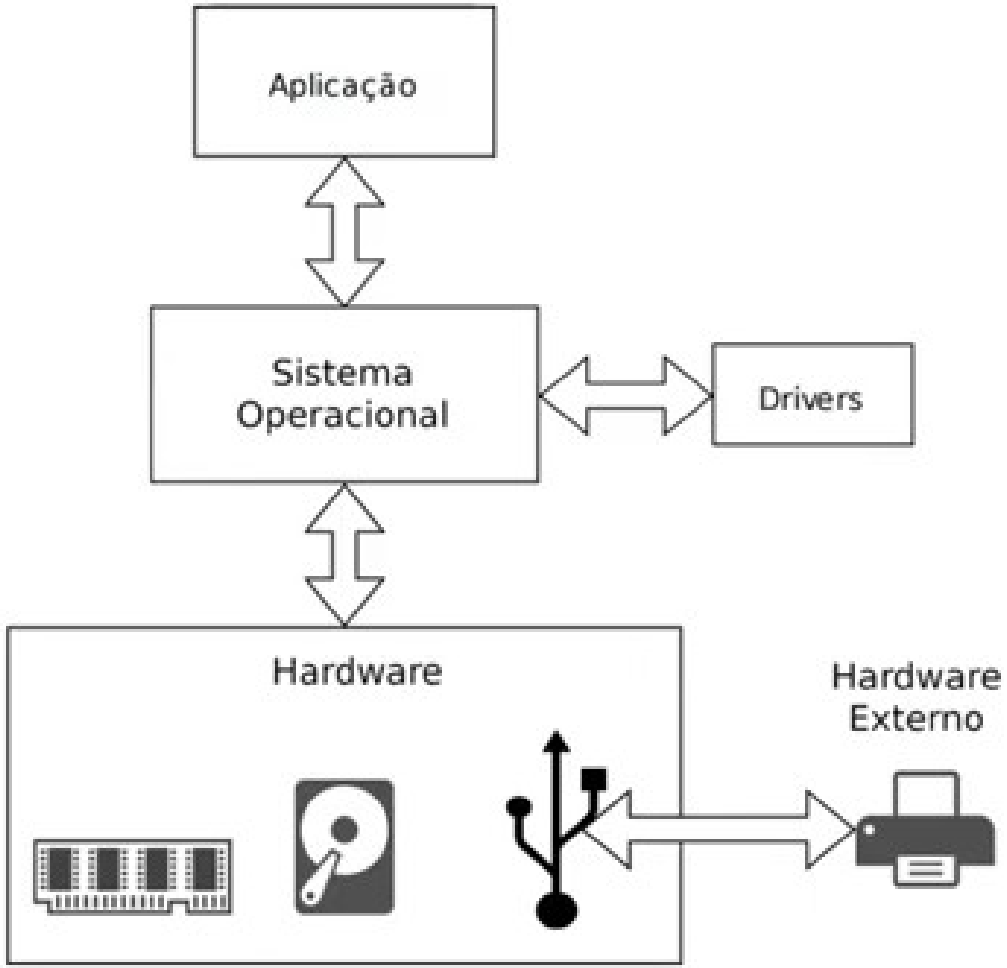
Capítulo 2: Estruturas de Sistema Operacional

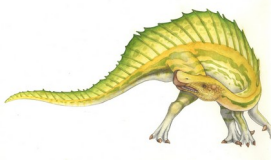
- Serviços do Sistema Operacional
- Interface com o Usuário de Sistema Operacional
- Chamadas de Sistema
- Tipos de Chamadas de Sistema
- Programas de Sistema
- Projeto e Implementação de Sistema Operacional
- Estruturas de Sistema Operacional
- Máquinas Virtuais
- Depuração de Sistema Operacional
- Geração de Sistema Operacional
- Inicialização do Sistema (Boot)

Objetivos

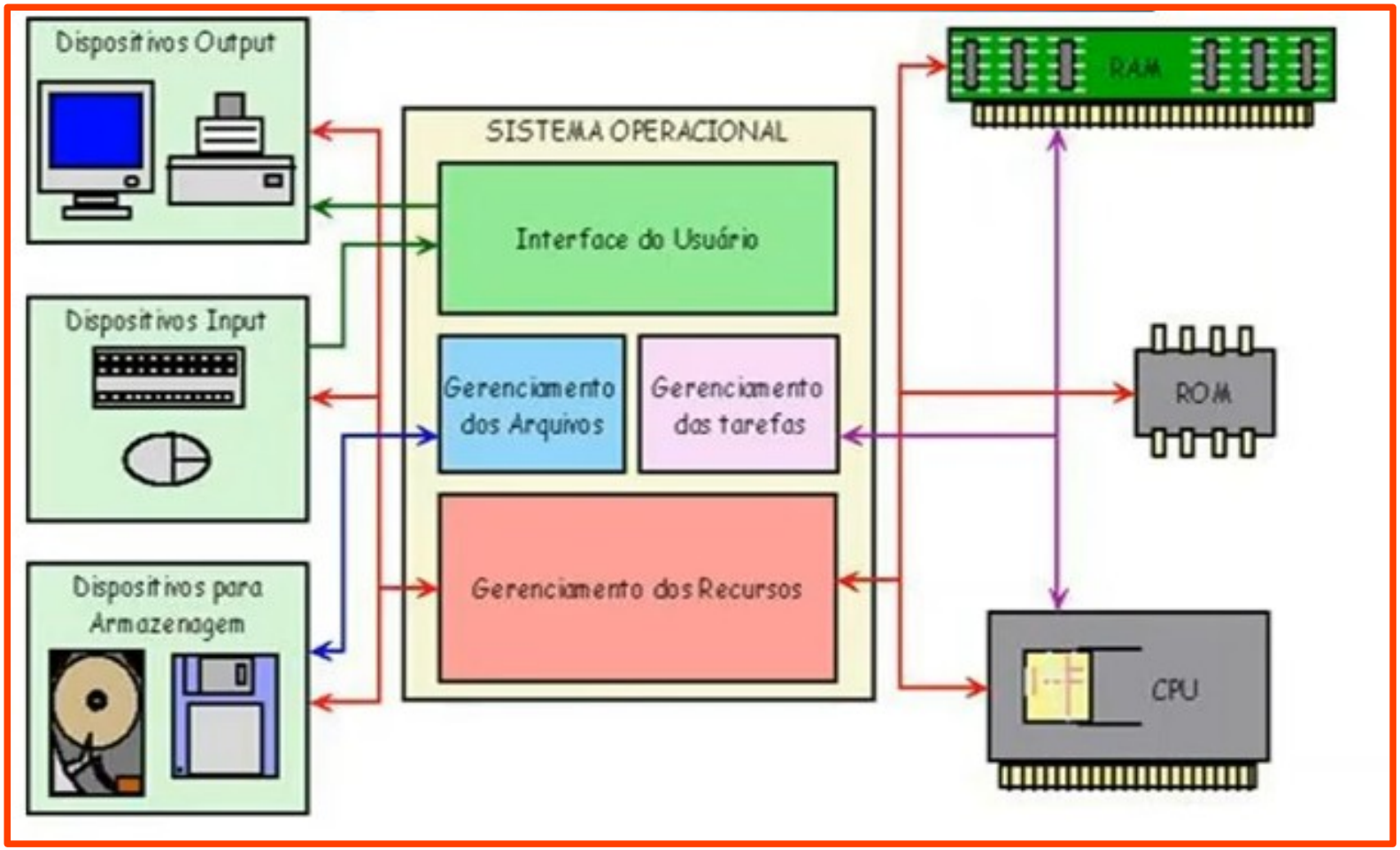
- Descrever os serviços que um sistema operacional fornece aos usuários, processos e outros sistemas
- Discutir as várias formas de estruturar um sistema operacional
- Explicar como sistemas operacionais são instalados e customizados e como é o processo de inicialização (*boot*)

Estrutura


Aplicação
Sistema Operacional
Drivers
Hardware
Hardware Externo



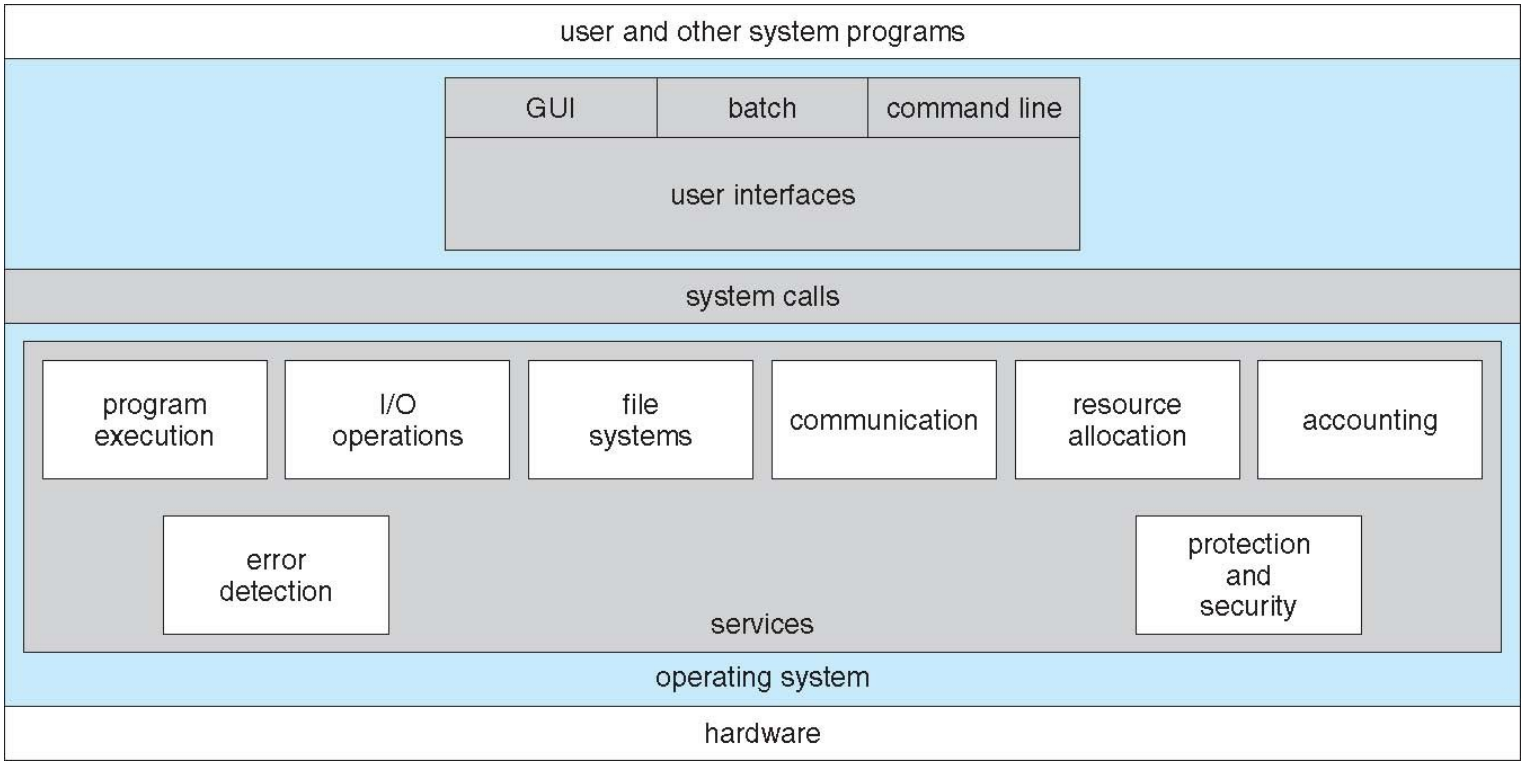
Estrutura


Dispositivos Output
Dispositivos Input
Dispositivos para Armazenagem
SISTEMA OPERACIONAL
Interface do Usuário
Gerenciamento dos Arquivos
Gerenciamento das tarefas
Gerenciamento dos Recursos
RAM
ROM
CPU

Serviços do Sistema Operacional

- Um conjunto de serviços do sistema operacional fornece funções que são úteis ao usuário:
  - Interface com o Usuário – Quase todos os sistemas operacionais possuem uma interface com o usuário (UI)
    - Varia entre **Interface de Linha de Comando** (CLI), **Interface Gráfica** (GUI – *Graphical User Interface*), **Batch** (em lote)
  - Execução de Programas – O sistema deve estar apto a carregar um programa na memória e executá-lo, terminar a execução, seja normalmente ou de forma anormal (indicando o erro)
  - Operações de E/S – Um programa em execução pode requisitar E/S, o que poderá envolver um arquivo ou um dispositivo de E/S.
  - Manipulação de Sistemas de Arquivos – O sistema de arquivo é de especial interesse. Obviamente, programas necessitam ler e escrever arquivos e diretórios, criar e deletar, procurar, listar informações de arquivos e gerenciar permissões.

Uma visão de serviços de um SO



Serviços do Sistema Operacional (Cont.)

- Um conjunto de serviços do sistema operacional fornece funções que são úteis ao usuário (Cont):
  - Comunicações – Processos podem trocar informações, no mesmo computador ou entre computadores conectados em rede
    - Comunicação pode ser via memória compartilhada ou através sistema de troca de mensagens (pacotes movidos pelo SO)
  - Detecção de Erro – SO precisa estar constantemente informado de possíveis erros
    - Pode ocorrer na CPU e no hardware de memória, em dispositivos de E/S, no programa do usuário
    - Para cada tipo de erro, SO deve realizar a ação apropriada para garantir a computação correta e consistente
    - Facilidades de depuração (debugging) podem aumentar a eficiência com que usuários e programadores usam o sistema

Serviços do Sistema Operacional (Cont.)

- Outro conjunto de funções do SO existe para garantir a operação eficiente do próprio sistema através do compartilhamento de recursos
  - **Alocação de Recursos** – Quando múltiplos usuários ou múltiplos jobs executam concorrentemente, recursos devem ser alocados para cada um deles
    - Muitos tipos de recursos – Alguns (como ciclos de CPU, memória principal, e armazenamento de arquivos) deve possuir código especial de alocação, outros (como os dispositivos de E/S) devem possuir requisições gerais e código liberado.
  - **Contabilização (Accounting)** – Manter o registro da quantidade de uso dos recursos pelos usuários e dos tipos de recursos empregados
  - **Proteção e Segurança** – Os donos das informações armazenadas em um sistema computacional multi-usuário ou em rede podem querer controlar o uso da informação, processos concorrentes não devem interferir uns nos outros
    - **Proteção** envolve garantir que todo acesso aos recursos do sistema é controlado
    - **Segurança** no sistema contra estranhos requer autenticação de usuários e até mesmo defesa contra tentativas de acesso inválidas de dispositivos de E/S externos
    - Se um sistema está protegido e seguro, precauções devem ser estabelecidas nele. Uma corrente é tão forte quanto o seu link mais fraco.

Interface com o Usuário de SO - CLI

CLI (*Command Line Interface* – Interface de Linha de Comando) permite a entrada de comandos diretos

- Algumas vezes implementada no kernel, outras por programas de sistemas
- Algumas vezes várias alternativas implementadas – **shells**
- Basicamente obtém um comando do usuário e o executa
  - Algumas vezes comandos internos, Algumas vezes somente nomes de programas (externos)
    - » No último caso, a adição de novas características não requer modificação do *shell*

Interface com o Usuário de SO - GUI

- GUI – *Graphical User Interface* (Interface Gráfica com o Usuário)
- Interface com área de trabalho amigável (*User-friendly* ***desktop*****)**
  - Normalmente mouse, teclado e monitor
  - **Ícones** representando arquivos, programas, ações, etc.
  - Cliques no mouse em objetos da interface causam ações variadas (obter informações, opções, funções de execução, abertura de diretório – conhecido como **pasta**)
  - Inventado no Xerox PARC
- Muitos sistemas hoje incluem tanto interface CLI como GUI
  - Microsoft Windows é GUI com CLI "*command*" *shell*
  - Apple Mac OS X tem a interface "Aqua" GUI com um kernel UNIX abaixo e *shells* disponíveis
  - Solaris é CLI com interfaces GUI opcionais (Java Desktop, KDE)

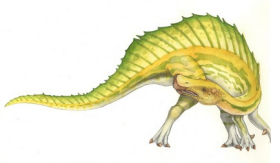
Interpretador de Comandos Bourne Shell

Terminal

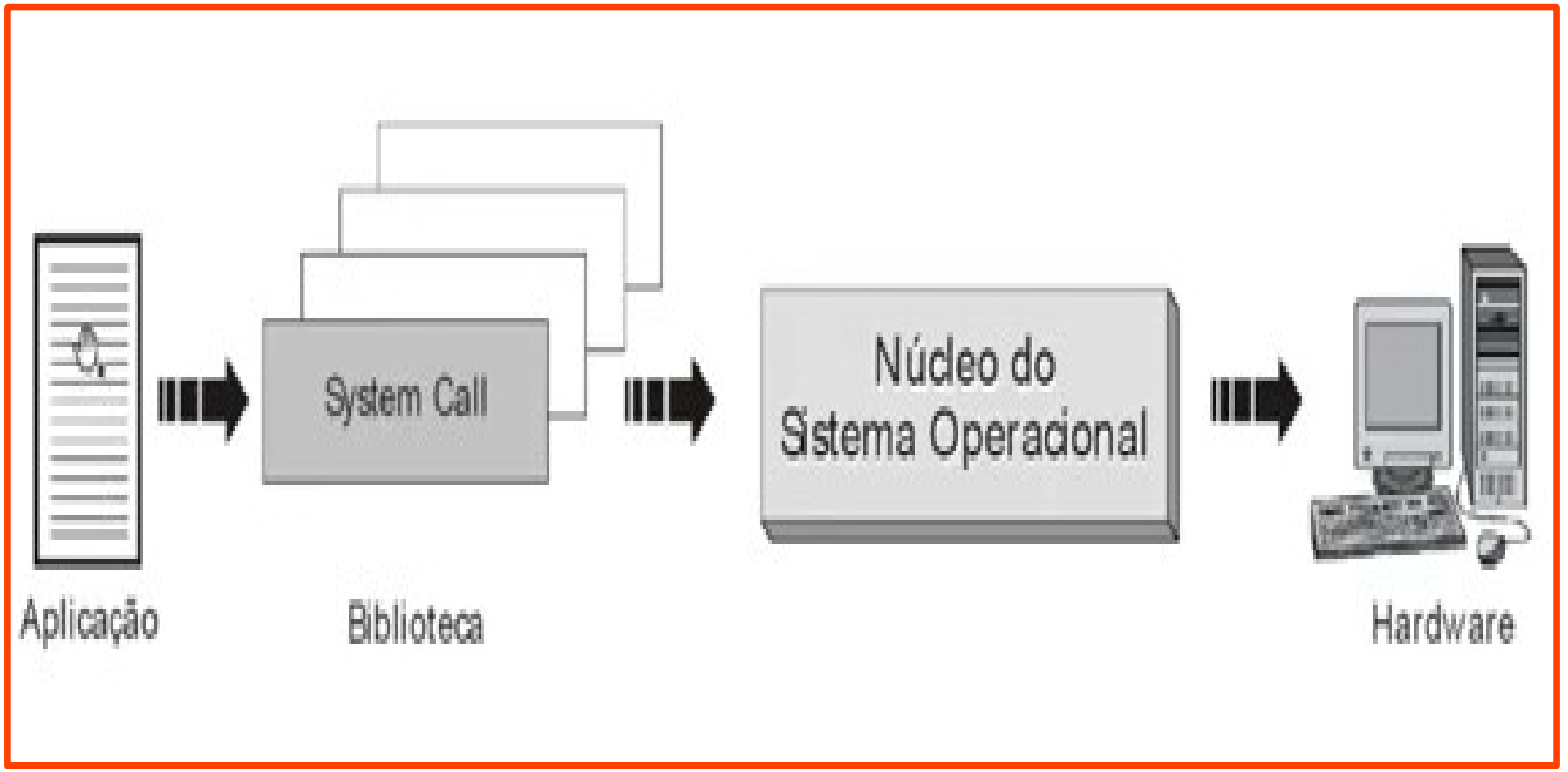
File Edit View Terminal Tabs Help

```
fd0       0.0    0.0    0.0    0.0  0.0  0.0    0.0   0   0
sd0       0.0    0.2    0.0    0.2  0.0  0.0    0.4   0   0
sd1       0.0    0.0    0.0    0.0  0.0  0.0    0.0   0   0
                 extended device statistics
device    r/s    w/s   kr/s   kw/s wait actv  svc_t  %w  %b
fd0       0.0    0.0    0.0    0.0  0.0  0.0    0.0   0   0
sd0       0.6    0.0   38.4    0.0  0.0  0.0    8.2   0   0
sd1       0.0    0.0    0.0    0.0  0.0  0.0    0.0   0   0
(root@pbg-nv64-vm)-(11/pts)-(00:53 15-Jun-2007)-(global)
-(/var/tmp/system-contents/scripts)# swap -sh
total: 1.1G allocated + 190M reserved = 1.3G used, 1.6G available
(root@pbg-nv64-vm)-(12/pts)-(00:53 15-Jun-2007)-(global)
-(/var/tmp/system-contents/scripts)# uptime
 12:53am  up 9 min(s),  3 users,  load average: 33.29, 67.68, 36.81
(root@pbg-nv64-vm)-(13/pts)-(00:53 15-Jun-2007)-(global)
-(/var/tmp/system-contents/scripts)# w
  4:07pm  up 17 day(s), 15:24,  3 users,  load average: 0.09, 0.11, 8.66
User     tty           login@  idle   JCPU   PCPU  what
root     console      15Jun0718days      1         /usr/bin/ssh-agent -- /usr/bi
n/d
root     pts/3        15Jun07           18      4  w
root     pts/4        15Jun0718days                w
(root@pbg-nv64-vm)-(14/pts)-(16:07 02-Jul-2007)-(global)
-(/var/tmp/system-contents/scripts)#
```

A GUI do Mac OS X



Chamadas de Sistema



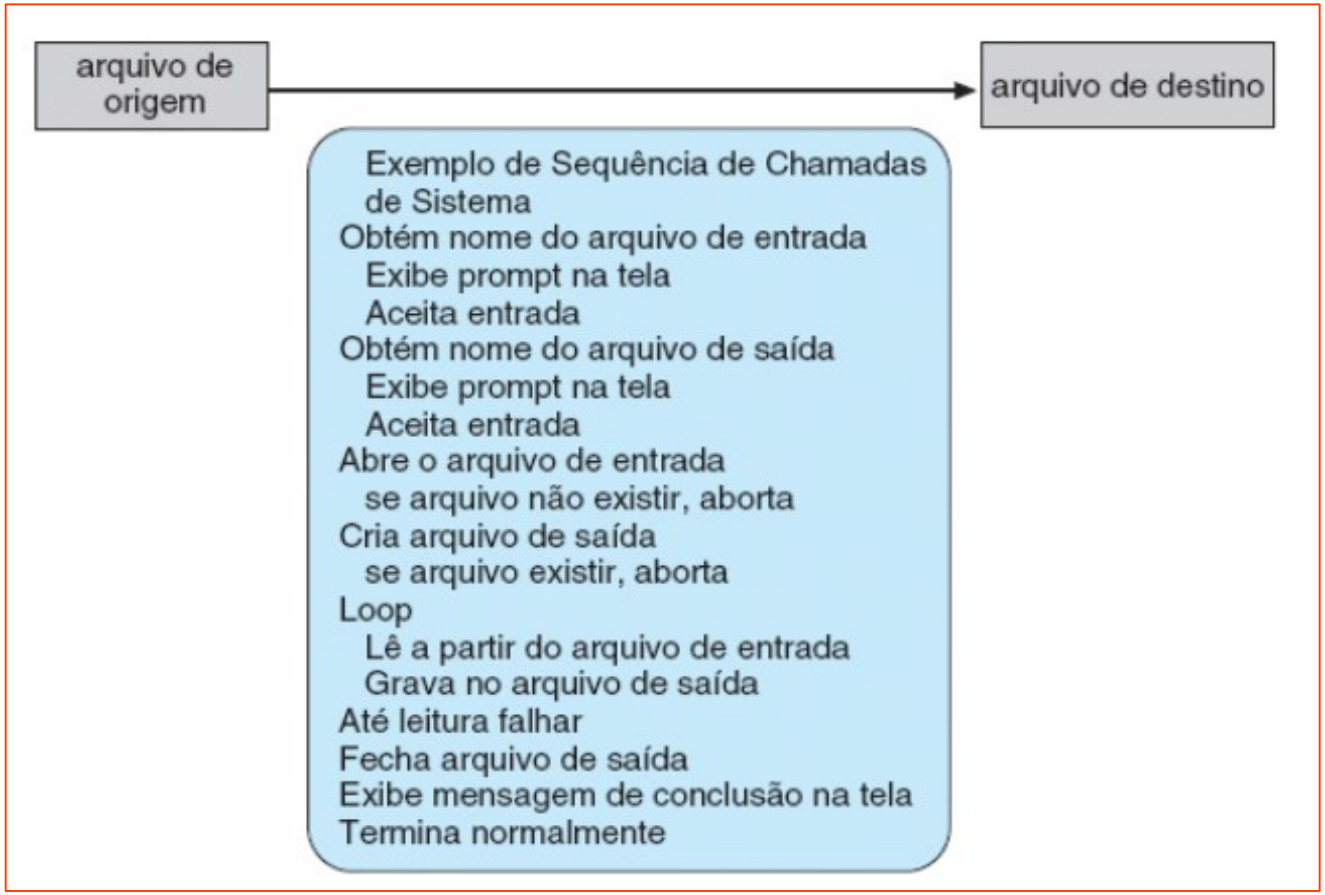
Chamadas de Sistema

- Interface de programação aos serviços fornecidos pelo SO
- Tipicamente escritos em uma linguagem de alto nível (C or C++)
- Geralmente acessada por programas via uma **API** (*Application Program Interface*) do que diretamente pelo uso de chamadas de sistema
- Três APIs mais comuns são Win32 API para Windows, POSIX API para sistemas baseados em POSIX (incluindo virtualmente todas as versões de UNIX, Linux, e Mac OS X), e Java API para a máquina virtual Java (JVM)
- Por que utilizar APIs ao invés das chamadas de sistemas?

(Observe que nomes de chamadas de sistemas utilizadas neste texto são genéricas)

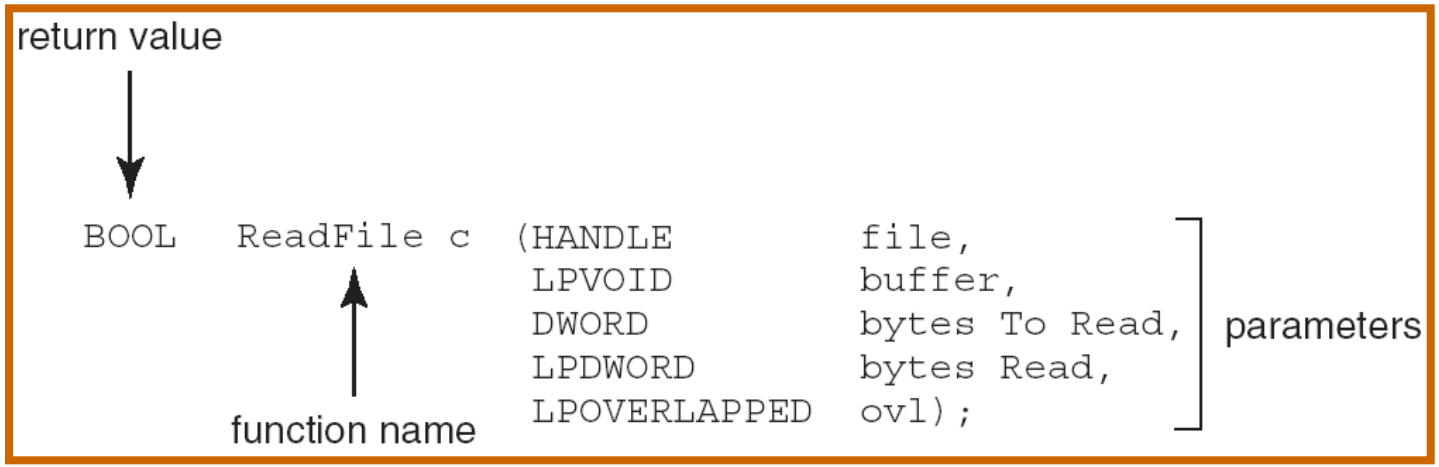
Exemplos de Chamadas de Sistema

- Sequência de chamadas de sistema para copiar o conteúdo de um arquivo em outro

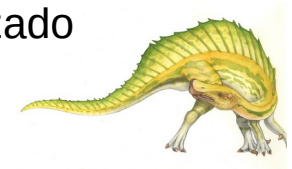


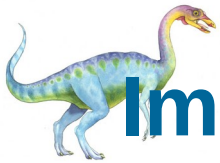
Exemplo de API Padrão

- Considere a função ReadFile() na
- Win32 API— una função para ler de um arquivo



- Uma descrição dos parâmetros passados para ReadFile()
  - HANDLE file—o arquivo a ser lido
  - LPVOID buffer—um buffer no qual os dados serão lidos
  - DWORD bytesToRead—o número de bytes a ser lido para o buffer
  - LPDWORD bytesRead—o número de bytes lidos durante a última leitura
  - LPOVERLAPPED ovl—indidica se E/S sobreposto está sendo utilizado

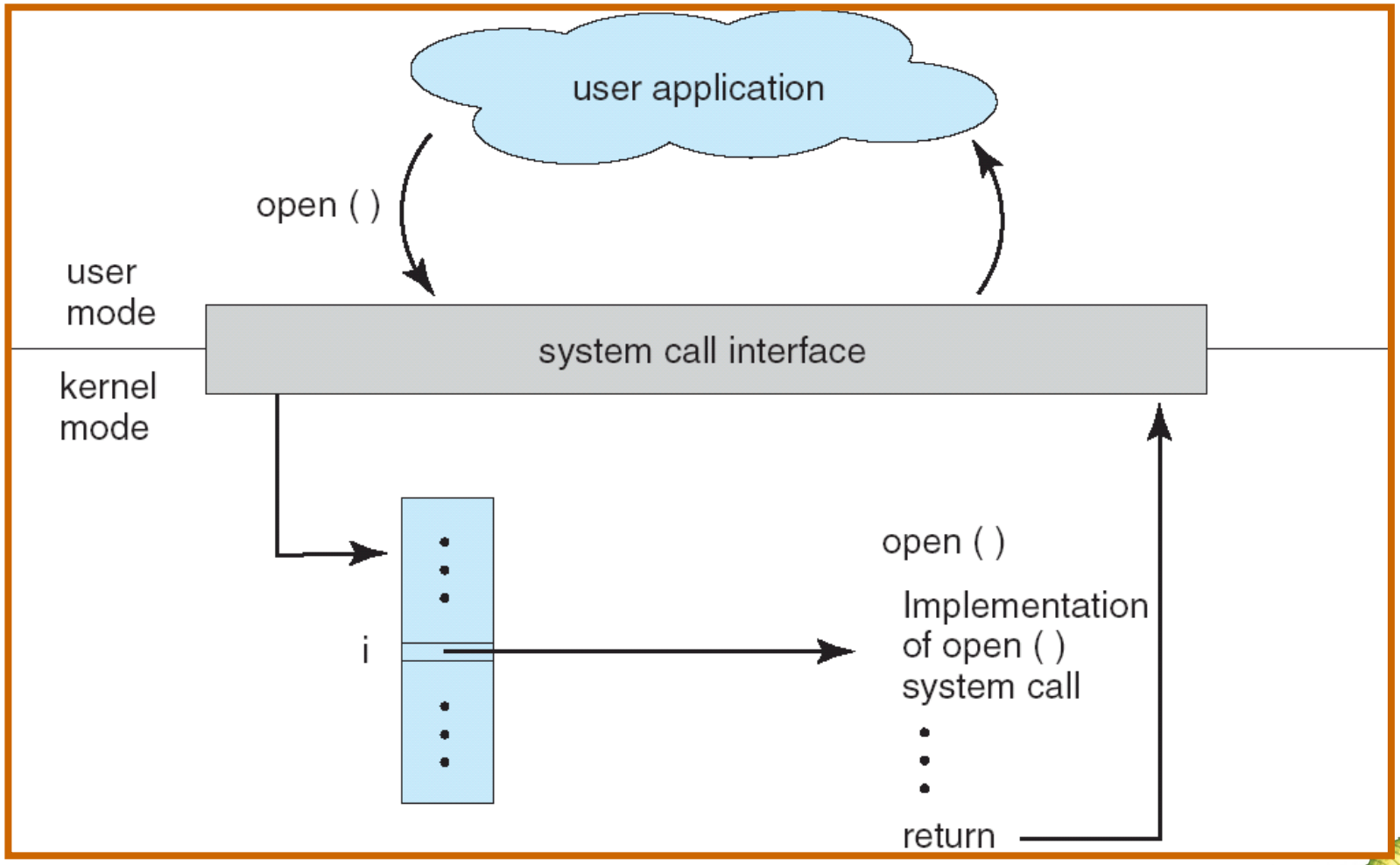




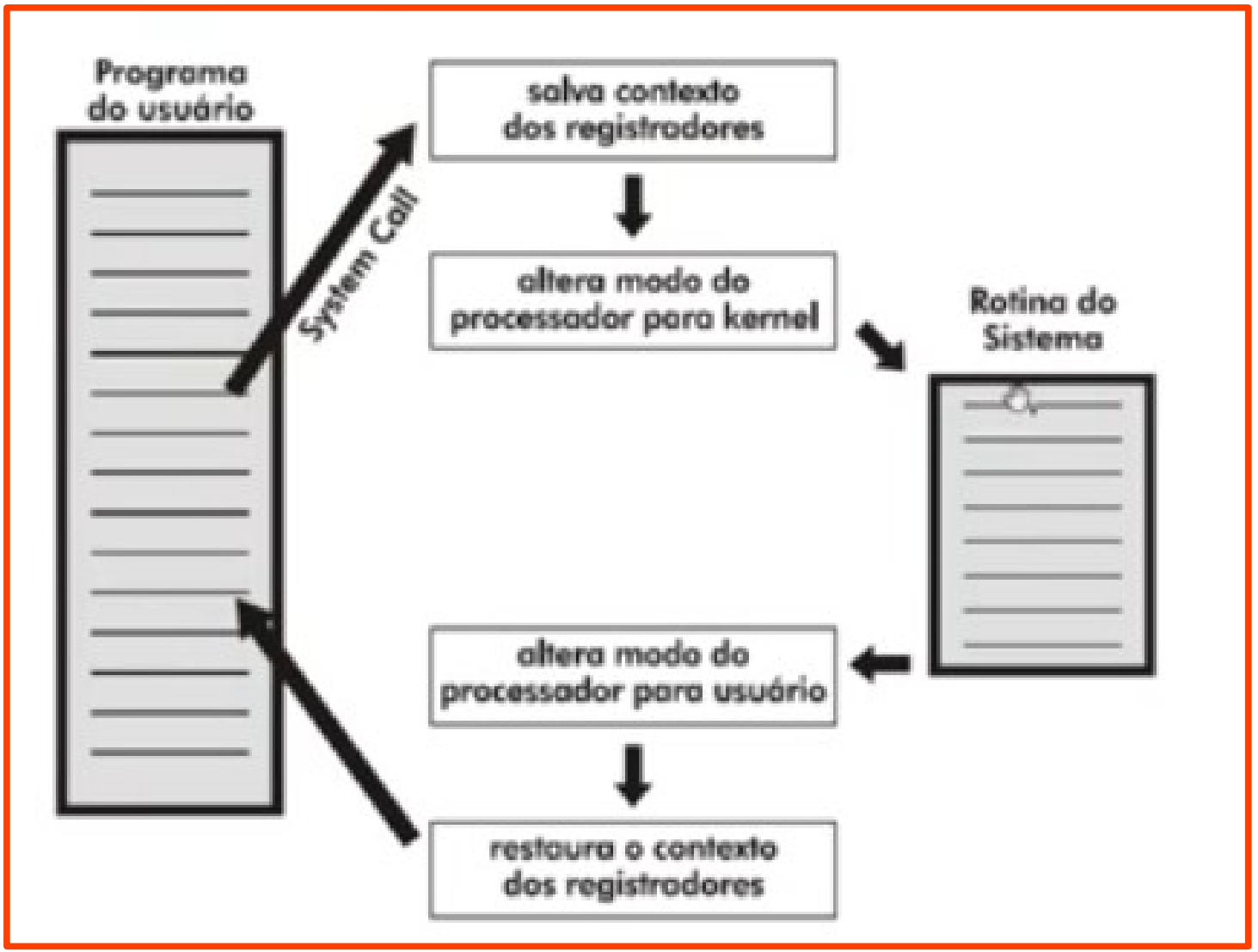
Implementação de Chamadas de Sistema

- Tipicamente, um número é associado com cada chamada de sistemas
  - A interface das chamadas de sistemas mantém uma tabela indexada de acordo com esses números
- A interface das chamadas de sistemas evoca a chamada de sistemas pretendida no kernel do SO e retorna o status e quaisquer valores de retorno
- O chamador não precisa saber nada sobre a implementação da chamada de sistemas
  - Só precisa obedecer a API e entender o que o SO irá realizar em resposta a chamada
  - Grande parte dos detalhes da interface do SO são escondidas dos programadores pela API
    - Gerenciado pela biblioteca de suporte a execução (conjunto de funções construídas em bibliotecas incluídas com o compilador)

API – Chamadas de Sistema – Relação com SO

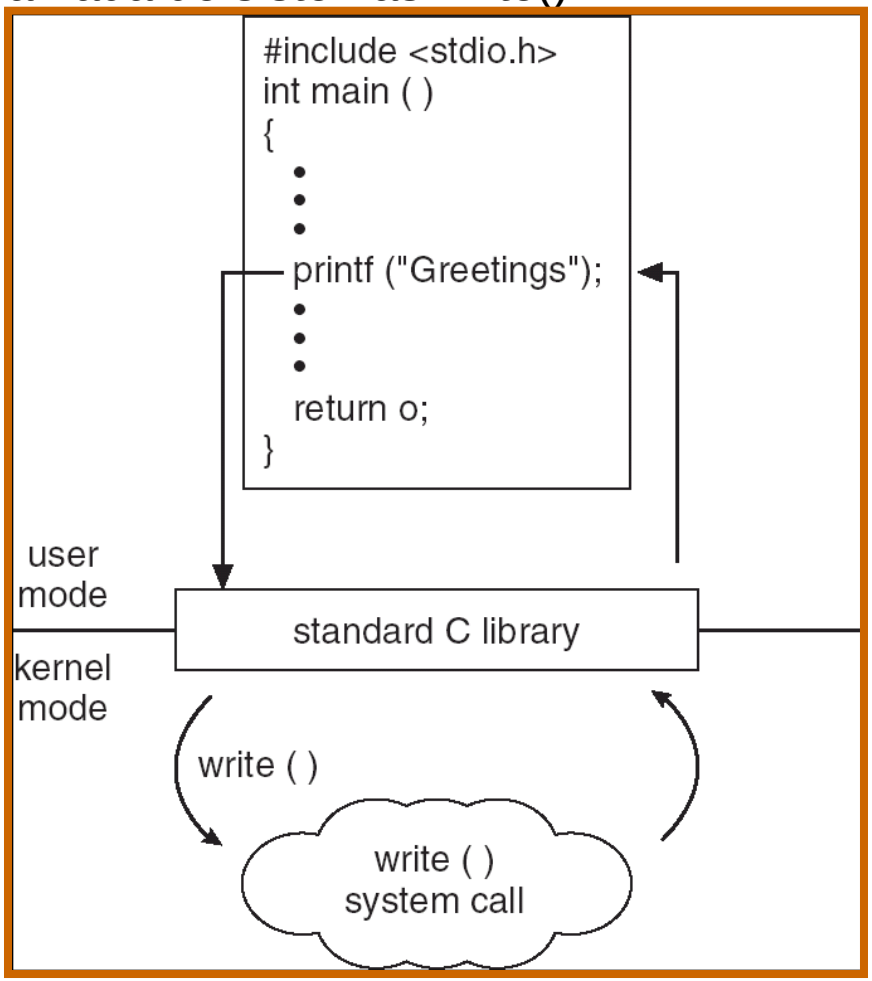


API – Chamadas de Sistema – Relação com SO



Exemplo de Biblioteca C Padrão

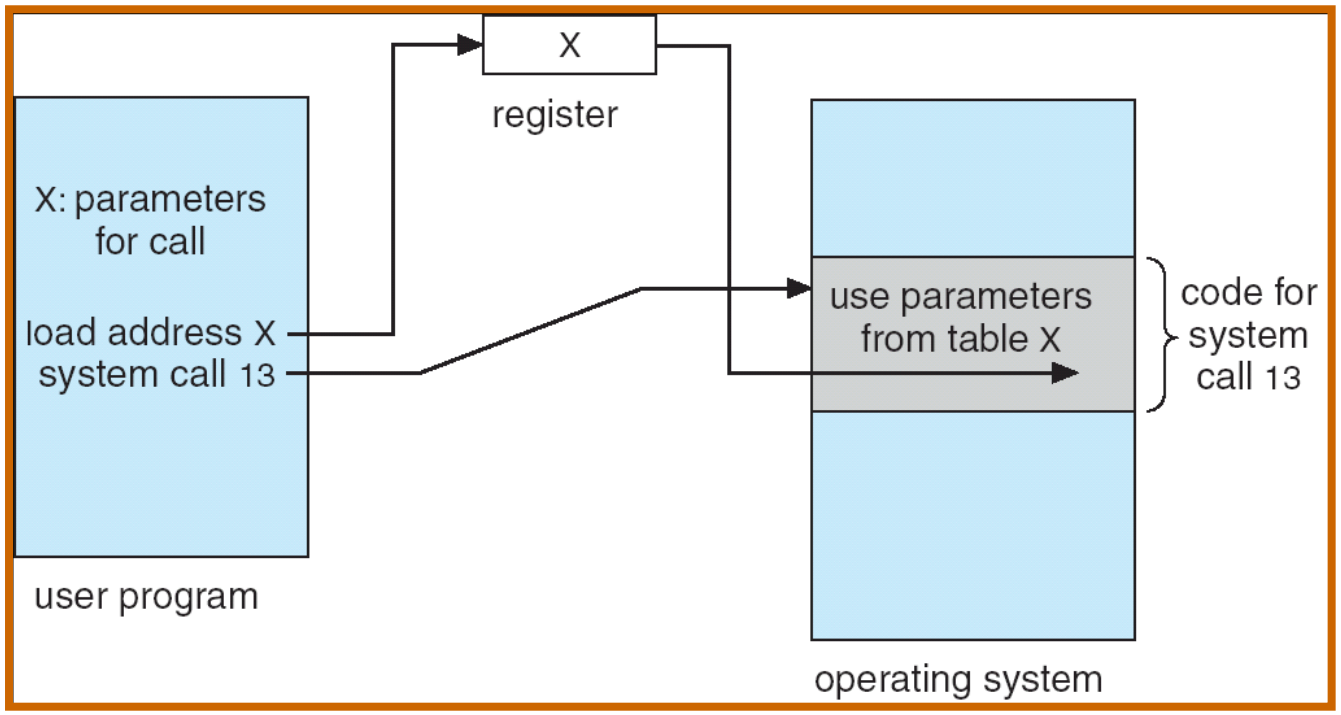
- Programa em C evocando a chamada de biblioteca printf(), que executa a chamada de sistemas write()



Passagem de Parametros nas Chamadas de Sistema

- Seguidamente, mais informações são necessárias do que a simples identificação da chamada de sistemas desejada
  - O tipo exato e conjunto de informações varia de acordo com o SO e com a chamada
- Três métodos gerais sáo usados para passar parâmetros ao SO
  - Mais simples: passar parâmetros em registradores
    - em alguns casos, pode existir mais parâmetros que registradores
  - Parâmetros armazenados em um bloco, ou tabela, na memória, e o endereço do bloco é passado como parâmetro em um registrador
    - Essa aboragem é utilizada pelo Linux e Solaris
  - Parâmetros colocados na *pilha (*empilhados / *push*) pelo programa e removidos (desempilhados / *pop*) desta pelo sistema operacional
  - Métodos de bloco e pilha não limitam o número ou tamanho dos parâmetros que estão sendo passados

Passagem de Parametros via Tabela



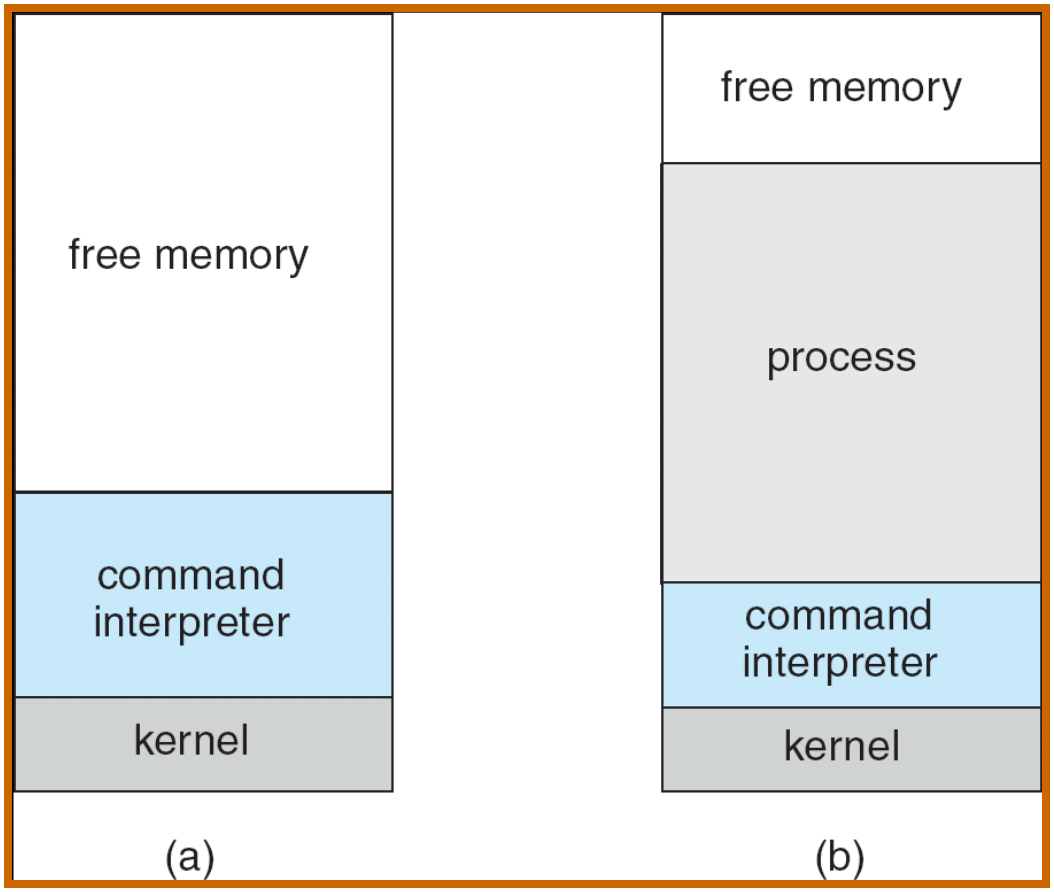
Tipos de Chamadas de Sistema

- Controle de processos
- Gerenciamento de Arquivos
- Gerenciamento de Dispositivos
- Manutenção de Informações
- Comunicações
- Proteção

Exemplos de chamadas de sistemas no Windows e Unix

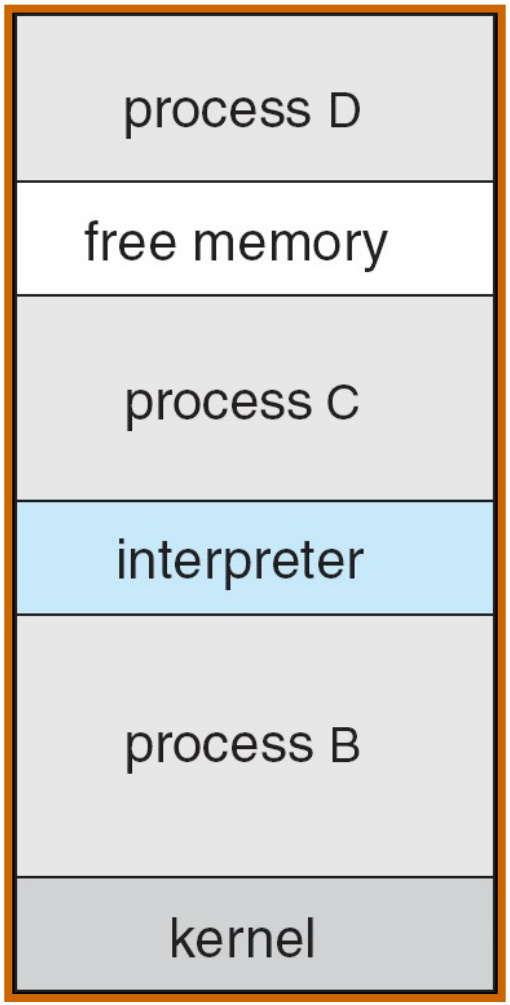
	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

execução do MS-DOS



(a) Na carga do sistema (b) executando um programa

FreeBSD Executando Vários Programas



Programas de Sistemas

- Programas de sistemas fornecem um ambiente conveniente para o desenvolvimento e execução de programas. Eles podem ser divididos em:
  - Manipulação de Arquivos
  - Informações (st*atus*)
  - Modificação de Arquivos
  - Suporte a Linguagens de Programação
  - Execução e carga de programas
  - Comunicações
  - Programas Aplicativos
- Grande parte da visão do usuário de um sistema operacional é definida pelos programas de sistemas, e não pelas chamadas de sistemas

Programas de Sistemas

- Fornecem um ambiente conveniente para desenvolvimento e execução de programas
  - Alguns deles são simples interfaces com o usuário para chamadas de sistemas; outros são consideravelmente mais complexos
- Gerenciamento de Arquivos - Criar, deletar, copiar, renomear, imprimir, dump, listar, e manipular genericamente arquivos e diretórios
- Status de informação
  - Alguns pedem ao sistema informações - data, hora, quantidade de memória disponível, espaço em disco, número de usuários
  - Outros fornecem informações detalhadas de desempenho, depuração e registros (*logging*)
  - Tipicamente, estes programas formatam e direcionam a saída para um terminal ou outro dispositivo de saída
  - Alguns sistemas implementam um registro- usado para armazenar e obter informações de configuração

Programas de Sistemas (cont.)

- Modificação de Arquivos
  - Editores de texto para criar e modificar arquivos
  - Comandos especiais para procurar conteúdos em arquivos e realizar transformações de texto
- Suporte a Linguagens de Programação- Compiladores, montadores, depuradores e interpretadores algumas vezes fornecidos
- Carga e execução de programas- Carregadores absolutos, relocadores, ligadores, carregadores de *overlay*, depuradores para linguagens de alto nível e de máquina
- Comunicações- Fornecer mecanismos para criar conexões virtuais entre processos, usuários e sistemas computacionais
  - Permitir aos usuários enviar mensagens de uma tela para outra, navegar em páginas web, enviar mensagens de correio eletrônico, efetuar *login* remoto, transferir arquivos de uma máquina para outra

Projeto e Implementação de Sistema Operacional

- Projeto e Implementação de SO não é um problema “solucionável”, entretanto algumas aproximações mostraram sucesso
- Estrutura interna de diferentes Sistemas Operacionais podem variar muito
- O começo é definir objetivos e especificações
- Afetado pela escolha do hardware, tipo de sistema
- Objetivos do *Usuário* e do *Sistema*
  - Objetivos do Usuário – o sistema operacional deve ser conveniente ao uso, fácil de aprender, confiável, seguro e rápido
  - Objetivos do Sistema – o sistema operacional deve ser fácil de projetar, implementar e manter; bem como ser flexível, confiável, livre de erros e eficiente

Projeto e Implementação de Sistema Operacional (Cont.)

- Princípio importante para separar

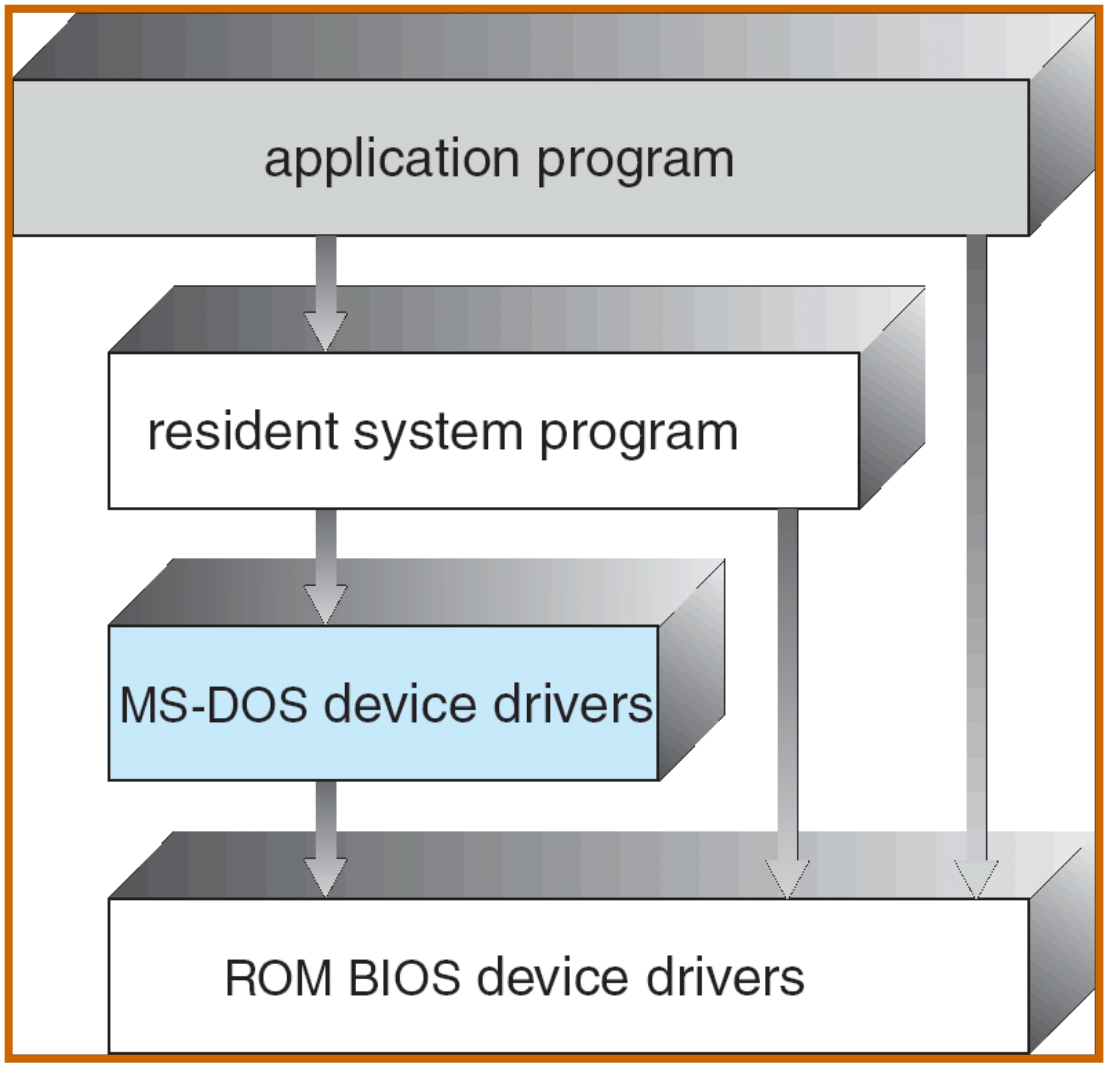
  Política: O que será feito?
  Mecanismo: Como será feito?

- Mecanismos determinam como fazer algo, políticas decidem o que será feito
  - A separação de política e mecanismo é um princípio muito importante. É possível obter máxima flexibilidade se decisões políticas possam ser realizadas posteriormente

Estrutura Simples

- MS-DOS – escrito para fornecer a maior funcionalidade no menor espaço
  - Não é dividido em módulos
  - Apesar do MS-DOS ter alguma estrutura, sua interface e seus níveis de funcionalidade não são bem separados

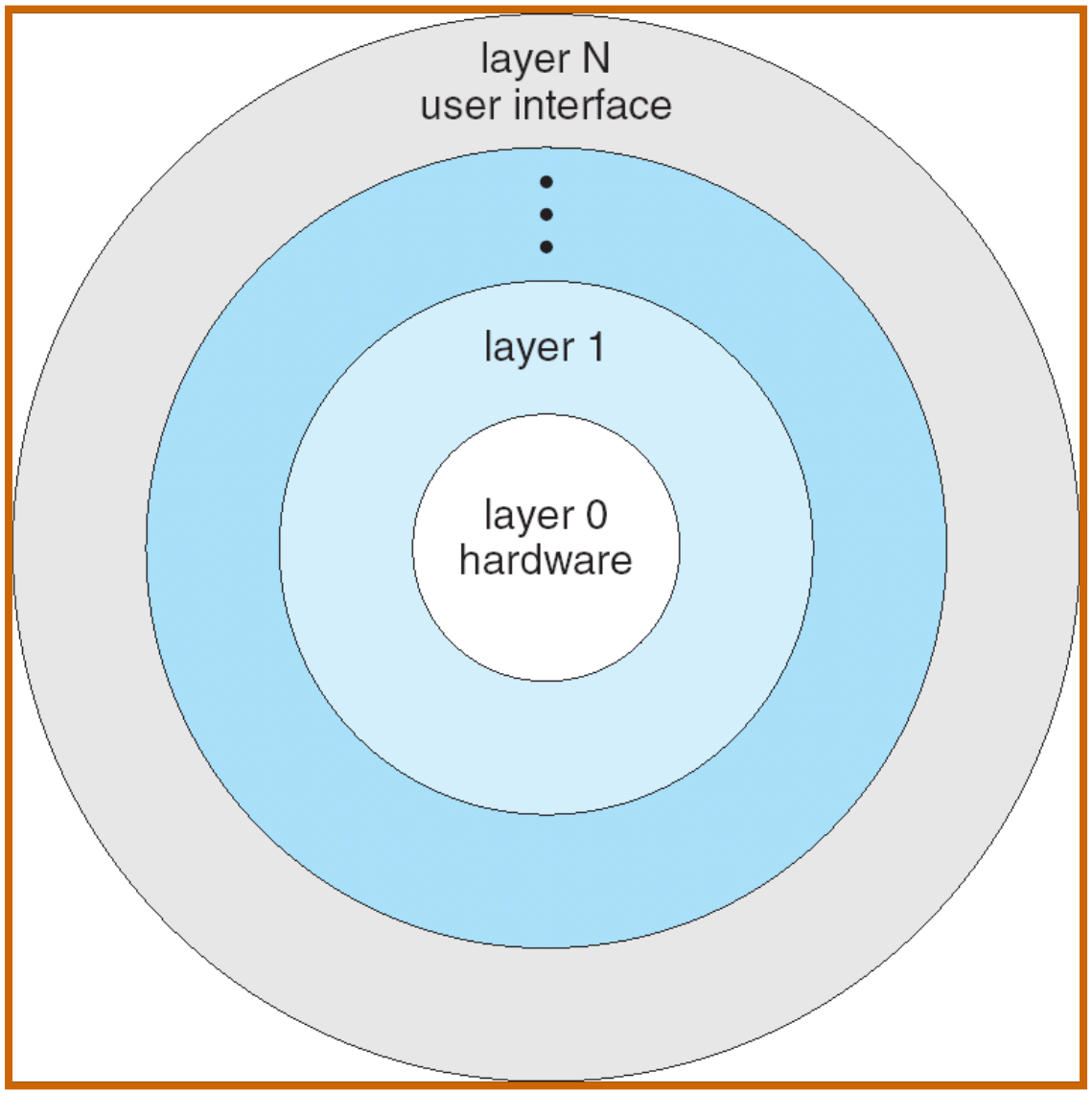
Estrutura em Camadas do MS-DOS



Estrutura em Camadas

- Um sistema operacional é dividido em um número de camadas (ou níveis), casa uma construída no topo das camadas abaixo. A camada mais inferior (camada 0) é o hardware; A camada de mais alto nível(camada N) é a interface com o usuário.
- Com modularidade, camadas são selecionadas de forma que cada uma use as funções (operações) e serviços somente das camadas de mais baixo nível.

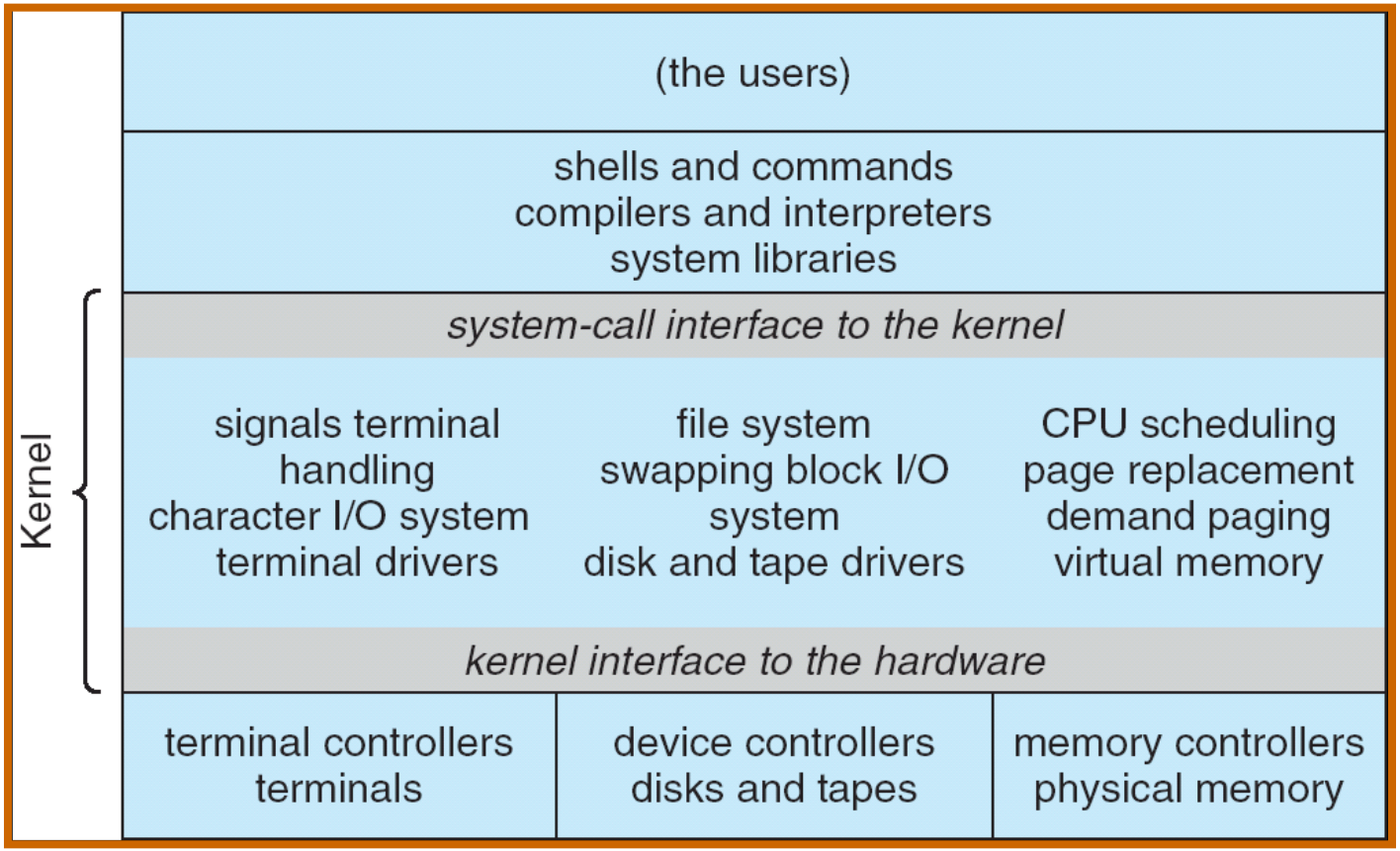
Sistema Operacional em Camadas



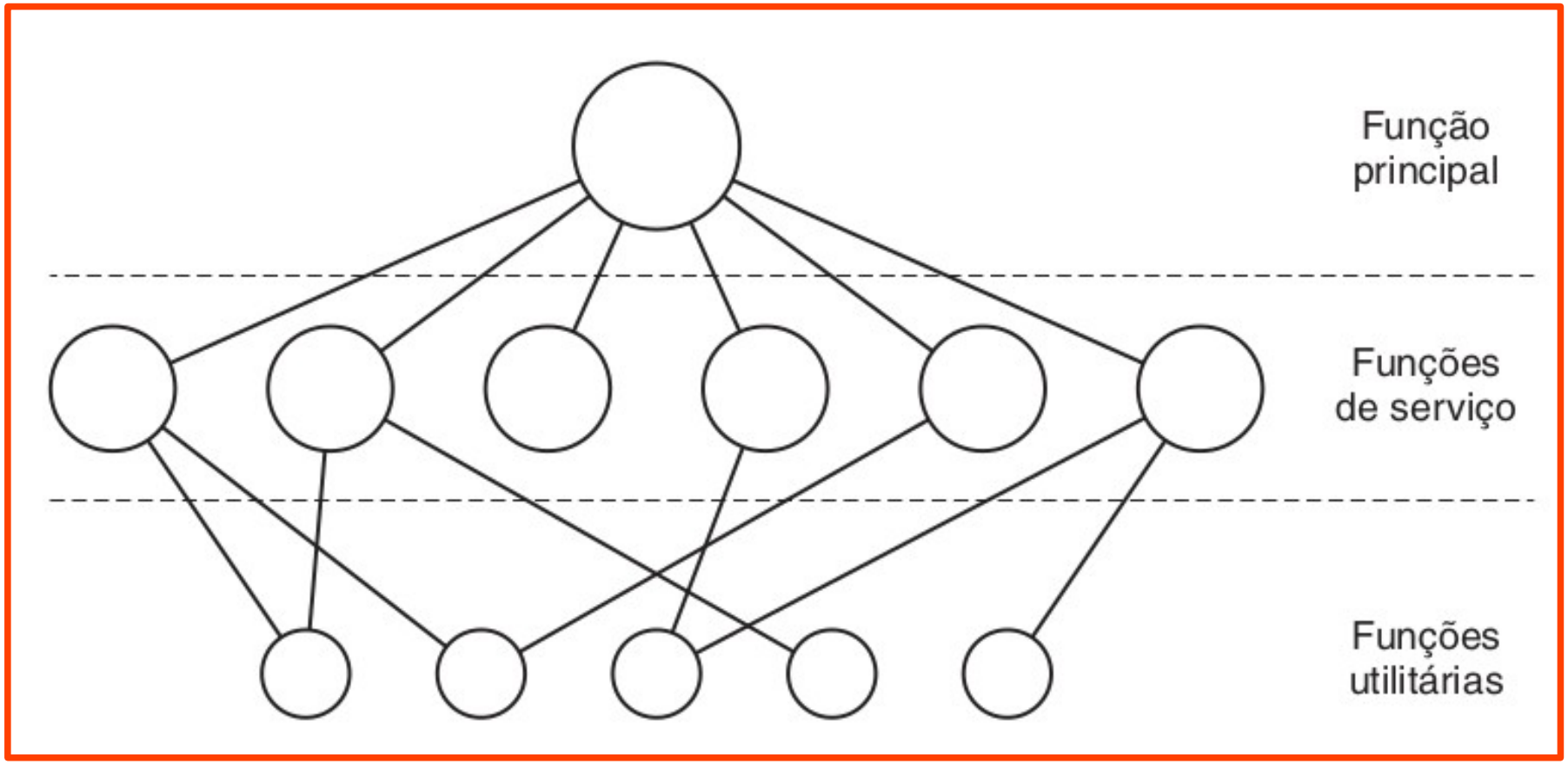
UNIX

- UNIX – limitado pela funcionalidade do hardware, o sistema operacional UNIX original tinha estrutura limitada. O SO UNIX consiste de duas partes separáveis
  - Programas de Sistemas
  - O kernel
    - Consite de tudo abaixo da interface de chamadas de sistemas e acima do hardware físico
    - Fornece o sistema de arquivos, escalonamento da CPU, gerência de memória e outras funções do sistema operacional; um grande número de funções para um nível

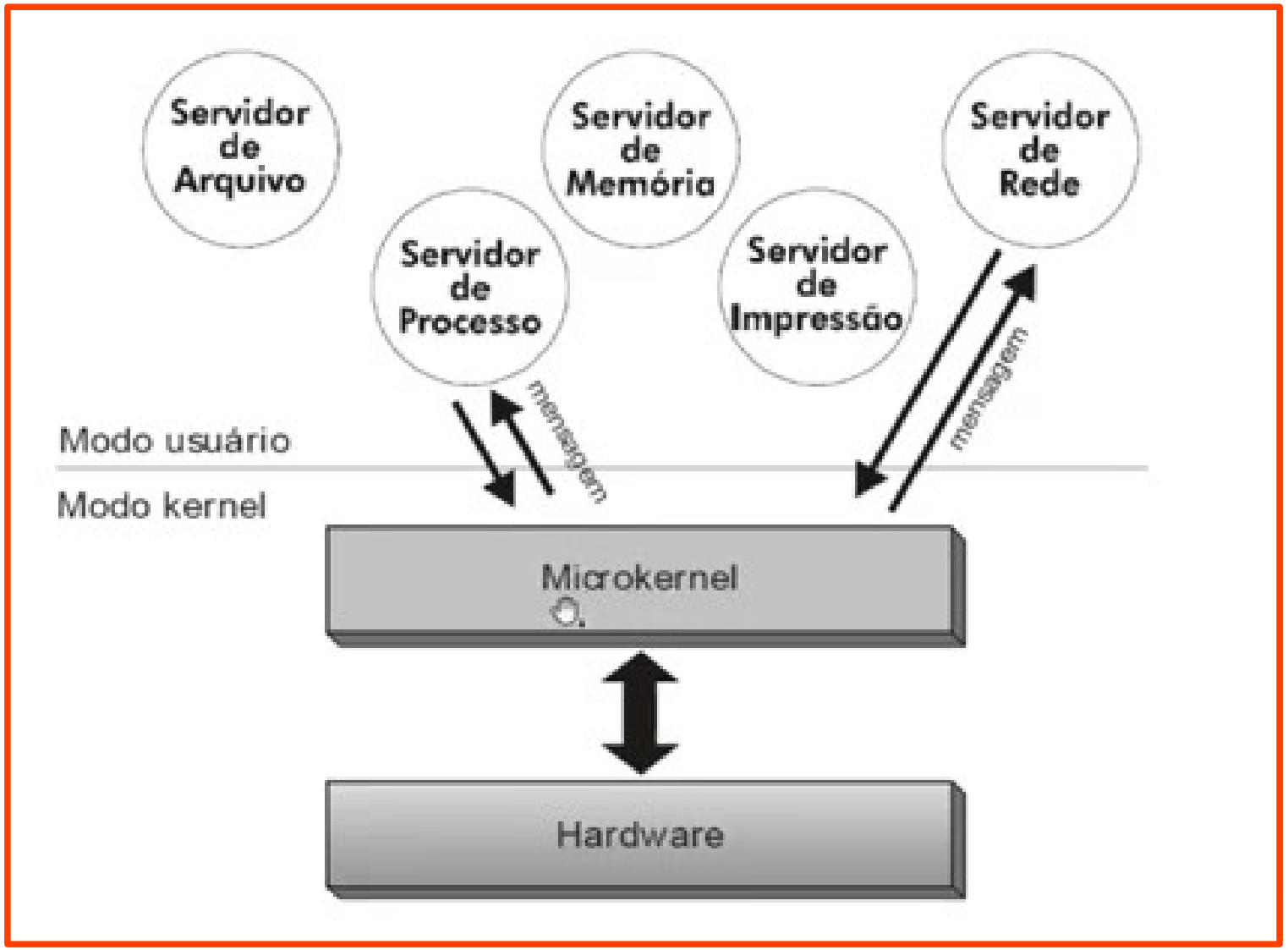
Estrutura do Sistema UNIX



Estrutura de um Sistema Monolítico



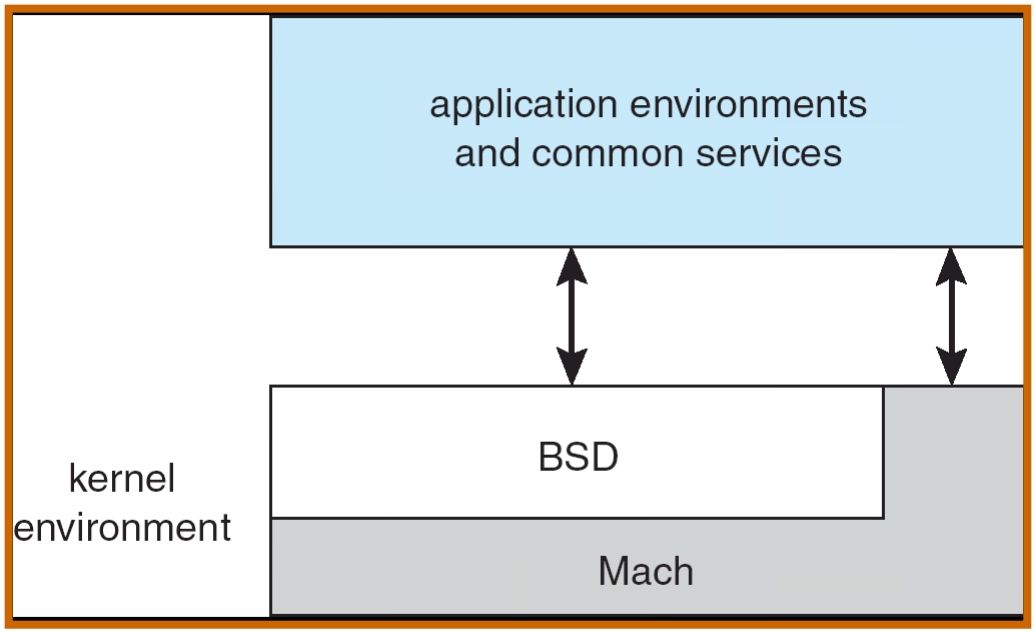
Estrutura Microkernel



Estrutura Microkernel

- Move tanto quanto possível do kernel para o espaço do "*usuário*"
- Comunicação ocorre entre módulos em nível usuário usando troca de mensagens (*message passing*)
- Benefícios:
  - Facilidade de estender um microkernel
  - Facilidade de portar o sistema operacional para novas arquiteturas
  - Mais confiabilidade (menos código está executando em modo kernel)
  - Mais seguro
- Desvantagem:
  - Sobrecarga causada pela comunicação entre o modo usuário e o modo kernel

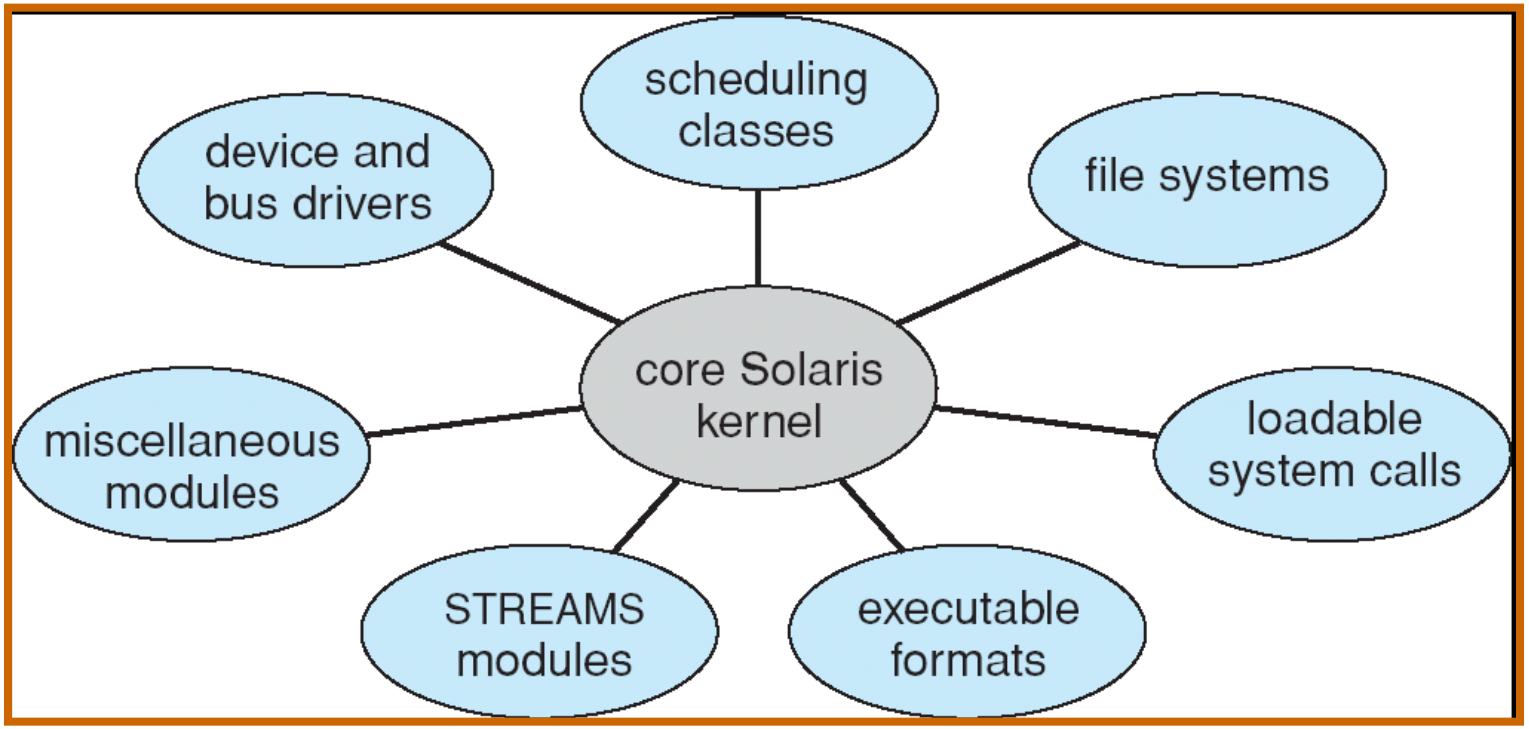
Estrutura do Mac OS X



Módulos

- Grande parte dos sistemas operacionais modernos implementam módulos no kernel
  - Usa a abordagem orientada a objetos
  - Cada componente chave é separado
  - Cada módulo se comunica com outra através de interfaces conhecidas
  - Cada módulo é carregado no kernel quando necessário
- Resumindo, similar à estrutura em camadas porém mais flexível

Estrutura Modular do Solaris

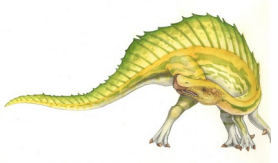


Alguns Sistemas Operacionais

Windows	monolítico	
Linux	monolítico	
UNIX	monolítico	
MacOS	monolítico BSD	micronúcleo Mach
Android	monolítico	
IOS	monolítico BSD	micronúcleo Mach
Minix	micronúcleo	

Máquinas Virtuais

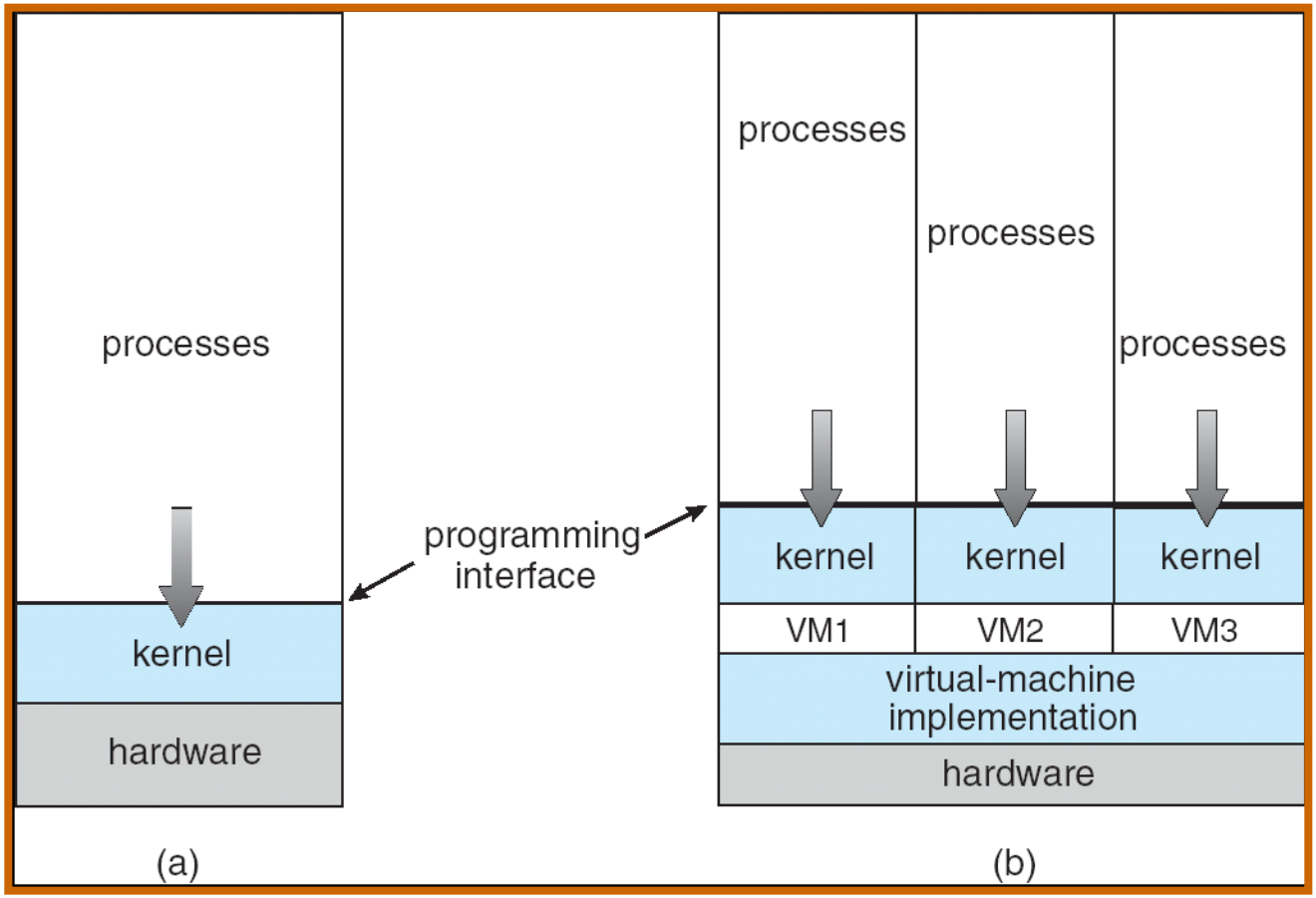
- Uma *máquina virtual* leva a abordagem em camadas ao extremo, para a sua conclusão lógica. Ela trata o hardware e o kernel do sistema operacional como se ambos fossem hardware.
- Uma máquina virtual fornece uma interface *idêntica* a do puro hardware abaixo.
- O sistema operacional cria a ilusão de múltiplos processos, cada um executando em seu próprio processador com sua própria memória (virtual).
- Cada **convidado**(*guest*) acessa uma cópia (virtual) do hardware



Máquinas Virtuais: História e Benefícios

- Primeira aparência comercial em mainframes IBM em 1972
- Fundamentalmente, múltiplos ambientes de execução (diferentes sistemas operacionais) podem compartilhar o mesmo hardware
- Proteção entre cada ambiente
- Algum compartilhamento (controlado) de arquivos pode ser permitido
- Comunicação entre ambientes, outros sistemas via rede
- Útil para desenvolvimento / teste
- Consolidação (*Consolidation*) de muitos recursos de baixo nível em poucos sistemas mais ocupados
- “Open Virtual Machine Format” – formato padrão de máquinas virtuais que permite a uma VM executar com muitas diferentes plataformas hospedeiras de máquinas virtuais

Máquinas Virtuais (Cont.)

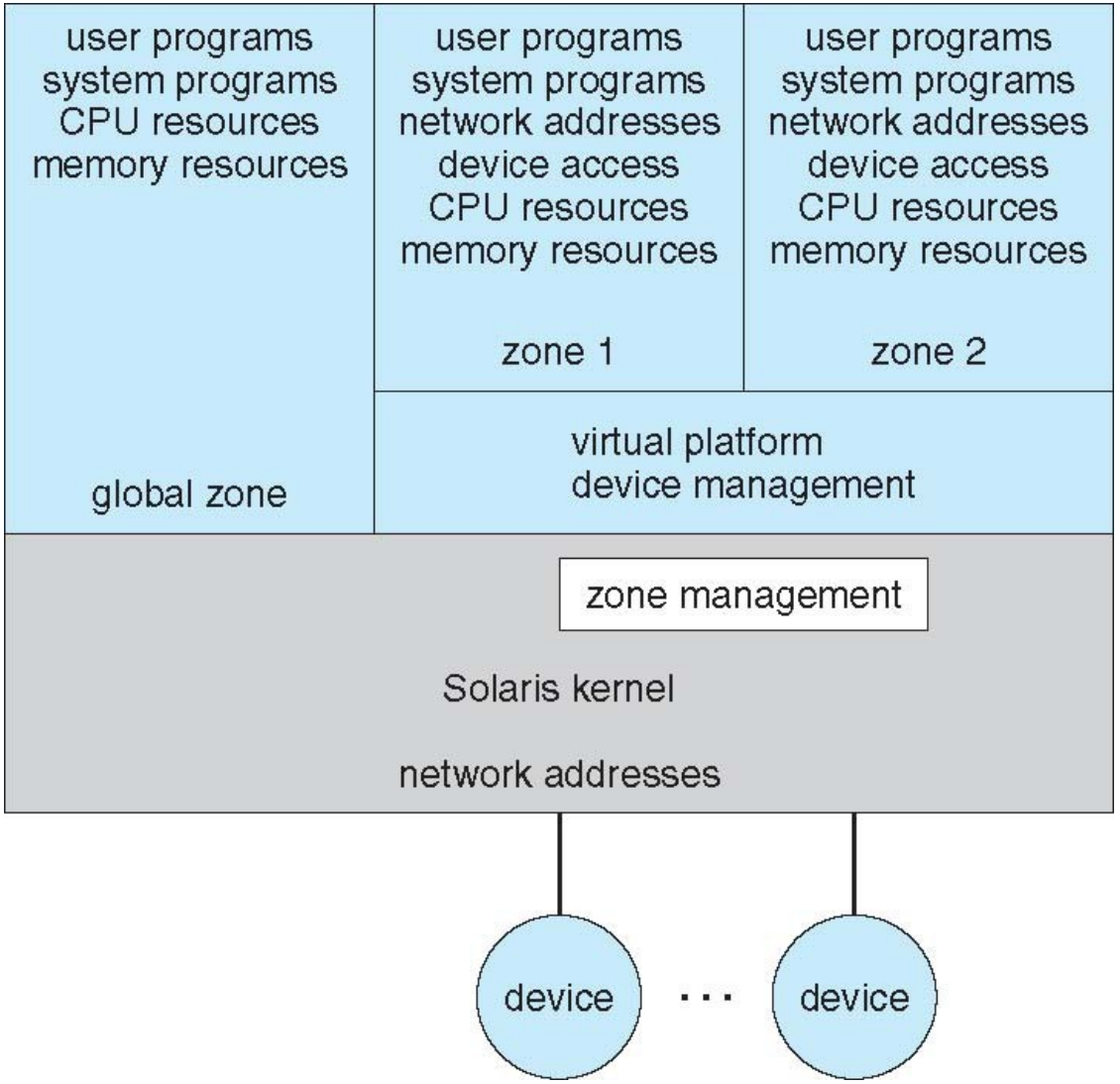


(a) Máquina não virtual (b) Máquina virtual

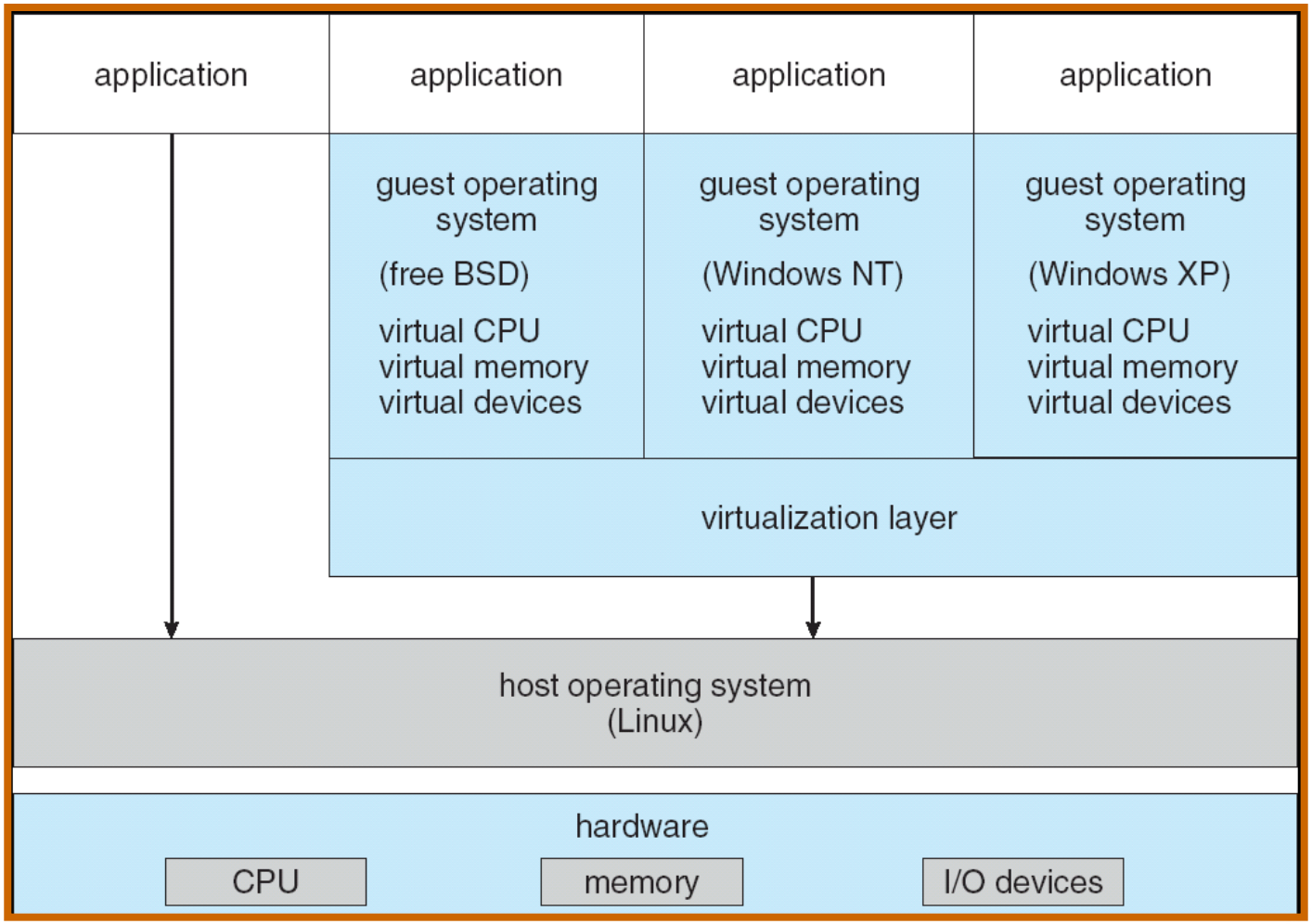
Para-virtualização

- Apresenta a *convidados* um sistema similar mas não igual ao hardware
- *Convidado* deve ser modificado para executar no hardware paravirtualizado
- Convidado pode ser um SO, ou no caso do Solaris 10, aplicativos executando em um *container*

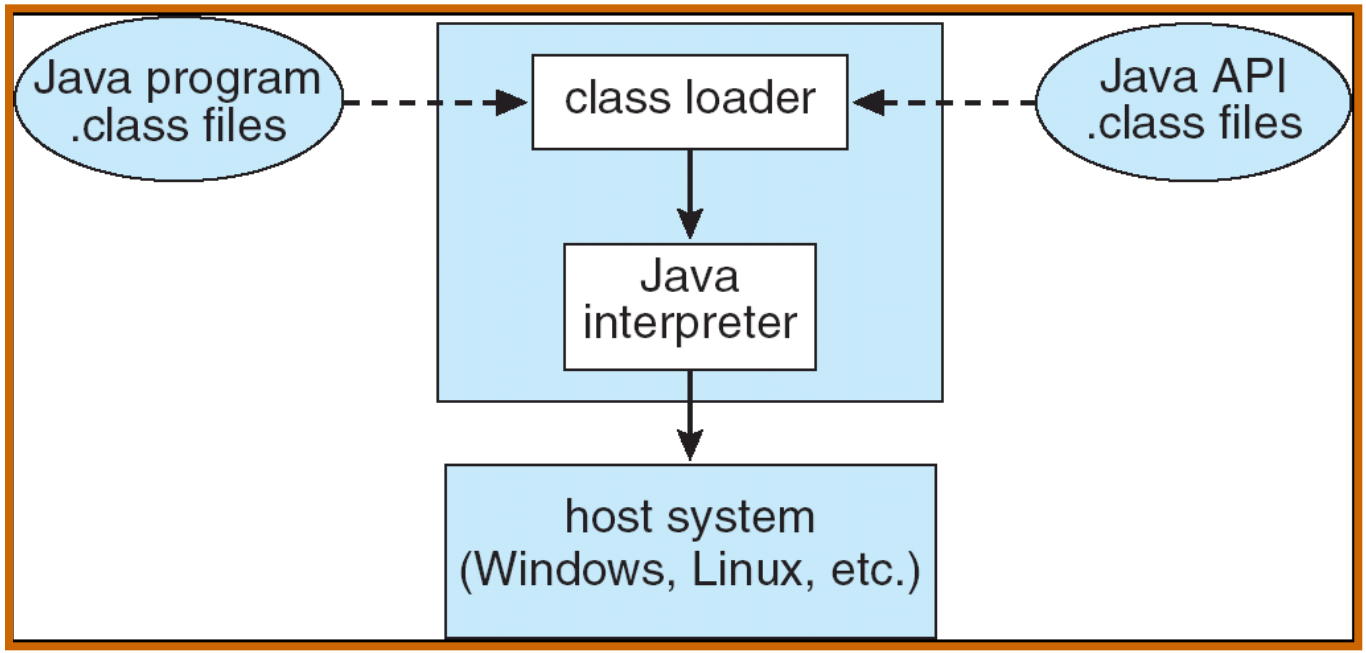
Solaris 10 com dois *Containers*



Arquitetura VMware

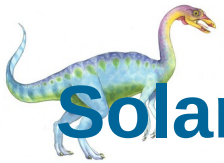


A Máquina Virtual Java – JVM



Depuração de Sistema Operacional

- Depuração é encontrar e corrigir erros ou *bugs*
- Sos geram arquivos de registro (*logs*) contendo informações de erros
- A falha de um aplicativo pode gerar arquivos *core dump* capturando a memória do processo
- Falhas dos sistemas operacionais podem gerar arquivos *crash dump* contendo memória do kernel
- Além de travamentos, *performance tuning* pode otimizar o desmpenho do sistema
- Lei de Kernighan: “Depurar é duas vezes mais difícil que escrever código pela primeira vez. Entretanto, se você escreve o código tão inteligente quanto possível, você não é, por definição, inteligente suficiente para depurá-lo”.
- Ferramenta DTrace no Solaris, FreeBSD, Mac OS X permite intrumentação ao vivo na produção de sistemas
  - Sondagens (*probes*) são disparadas quando o código é executado, capturando o estado dos dados e mandando a um consumidor das sondagens



Solaris 10 dtrace Seguindo Chamada de Sistemas

```
# ./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
```

Geração de Sistema Operacional

- Sistemas Operacionais são projetados para executar em qualquer uma máquina de uma determinada classe; o sistema deve ser configurado para cada computador específico.
- Programas SYSGEN obtém informações a respeito da configuração específica do hardware.
- *Booting* – iniciar um computador carregando o kernel.
- *Bootstrap program* – código armazenado em ROM que é capaz de localizar o kernel, carregá-lo na memória e iniciar sua execução.

Inicialização do Sistema – Boot

- Sistema operacional deve estar disponível ao hardware de maneira que este possa iniciá-lo
  - Pequena porção de código – **bootstrap loader**, localiza o kernel, carrega ele na memória e o inicia
  - Algumas vezes processo em dois passos no qual **bloco de boot** em localização fixa carrega o bootstrap loader
  - Quando o sistema é inicializado, execução começa em uma localização fixa de memória
    - *Firmware* usada para armazenar código de boot inicial

Fim do Capítulo 2