Sistemas Operacionais:
Fundamentos, Histórico e Estruturas de Núcleo

Prof. Dr. Márcio Castro
marcio.castro@ufsc.br

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

1 Introdução

Introdução

- **O que se espera de um sistema de computação?**
  - Execução de **programas de usuário**
  - Programas servem para auxiliar os usuários a realizar determinadas tarefas
- **Programas**
  - Possuem **muito em comum**: acesso ao disco, memória, E/S, ...
  - **Problema:** programas podem apresentar **necessidades conflitantes** se utilizados simultaneamente por um ou mais usuários (**disputa de recursos**)

Introdução

- **Sistema Operacional (SO)**
  - Camada de ***software*** **colocada entre o** ***hardware*** **e os programas** que executam tarefas para os usuários
  - **Controla** e **coordena** o uso do *hardware* entre vários programas aplicativos e usuários

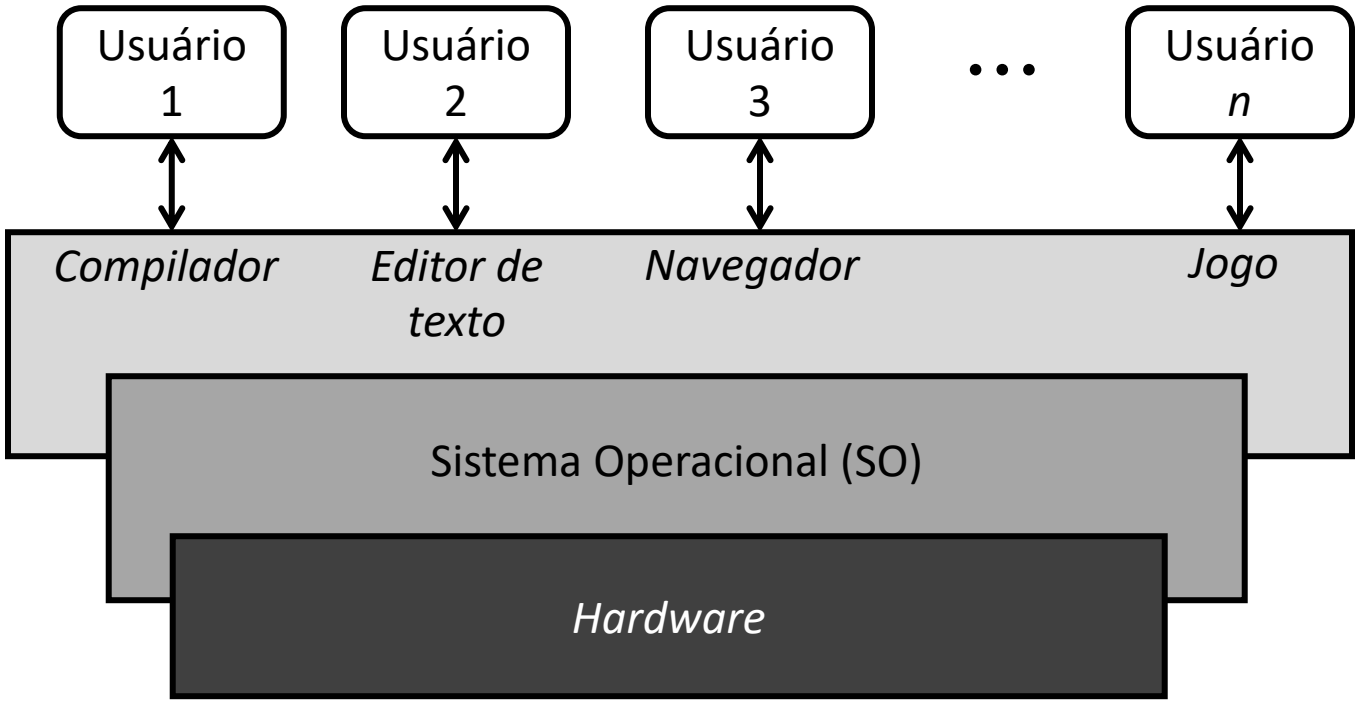
Introdução

- **Programas aplicativos**
  - Programas utilizados pelos usuários “mais comuns” de um sistema operacional
  - **Exemplos**: firefox, word, adobe photoshop, ...

- **Programas de sistema**
  - Programas de base do SO que fornecem serviços ou abstrações para os programas aplicativos
  - **Exemplos**: gerenciador de tarefas (Windows), ps (Linux), ...

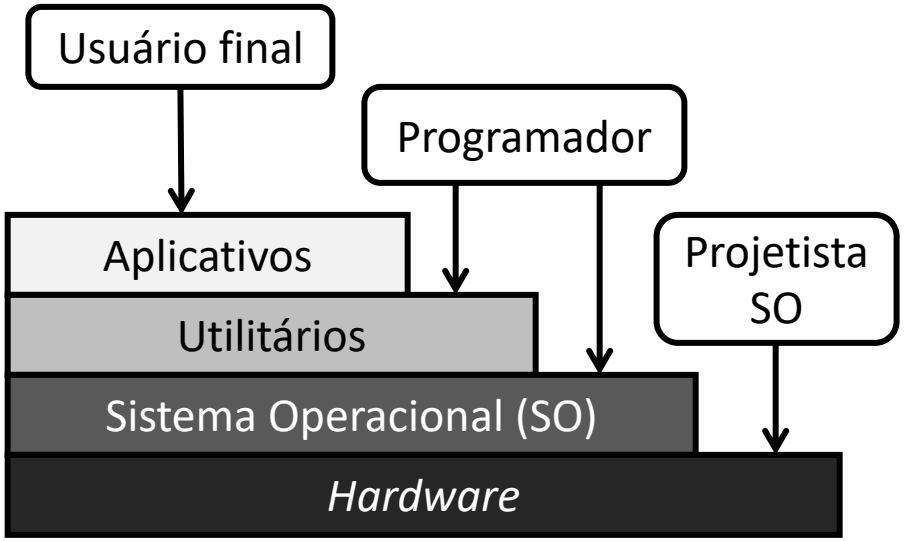
Introdução

- Interação entre usuários, programas aplicativos e SO



Objetivos do SO

- **Eficiência**
  - Maximizar o uso do *hardware* (**distribuição dos recursos**)
- **Conveniência**
  - Esconder os detalhes de baixo nível (**abstração**)
- **Interface entre o usuário e o *hardware***



Serviços oferecidos pelo SO

- Carregamento/descarregamento de programas na memória
- Gerência e sistema de arquivos
  - Criar, ler, escrever, ...
- Utilização e gerência de periféricos
  - Alocação, leitura, escrita, ...
- Gestão de usuários
- Proteção entre usuários
- Interface com o usuário
- Detecção de erros
  - *Hardware* e programas

2

Histórico

Um pouco sobre a evolução dos SOs

Histórico dos SOs

- **Inicialmente**
  - SO **inexistente**
  - Usuário é o **programador** e **operador** da máquina
  - Alocação do computador feito por planilha

- **Problemas**
  - Péssima utilização dos recursos
  - Usuário era obrigado a ter um conhecimento aprofundado da máquina

Sistemas em lote (*batch*)

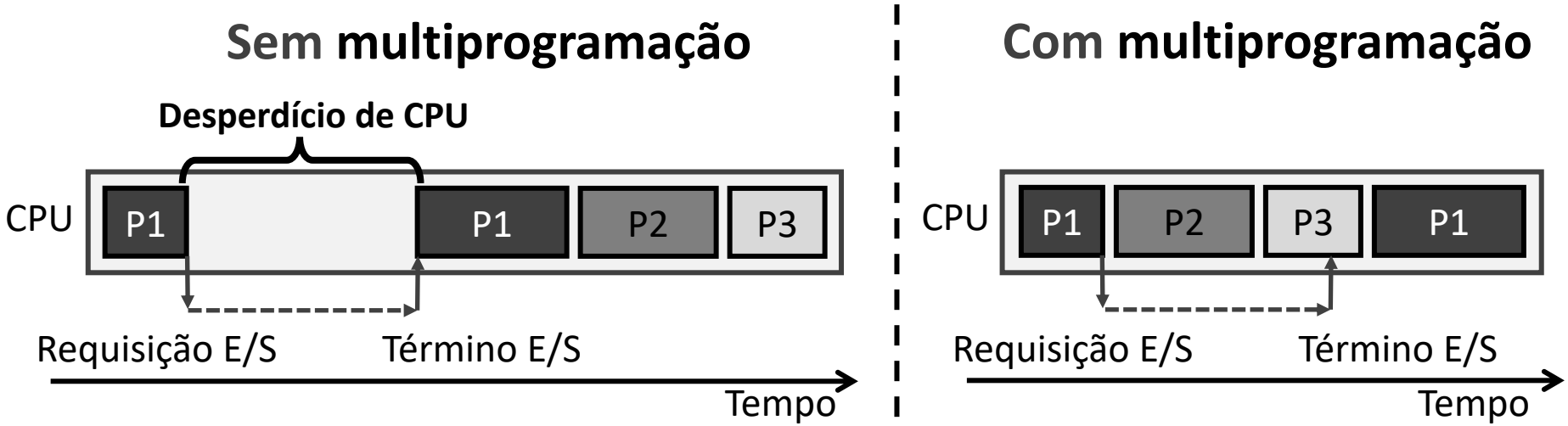
- Introdução de **operadores de máquina profissionais** (usuário não era mais operador da máquina)
- **Conceito de *job***
  - **Programa a ser compilado e executado**, juntamente com seus dados de entrada (na época eram cartões perfurados)
  - *Jobs* eram organizados **em lote (*batch*)** de acordo com as suas necessidades (ex. *jobs* que utilizavam o mesmo compilador, mesmos dados de entrada, ...)
- **Alternância entre *jobs***
  - Ainda era manual

Monitor residente

- Evolução do sistema em lote → **sequenciamento automático de *jobs***
  - Programa que reside permanentemente na memória
  - Controle da máquina é transferido de um *job* a outro
  - Considerado o **primeiro SO (rudimentar)**
- **Funcionamento:**
  - O monitor residente inicia sua execução na máquina
  - Quando um *job* é **submetido**, este toma o controle da máquina
  - O controle **retorna ao monitor residente**
- **Problema:** execução de apenas **um *job* por vez**

Sistemas multiprogramados

- **Multiprogramação**
  - **Vários programas na memória** ao mesmo tempo
  - Enquanto um programa aguarda por um recurso outro programa é executado
  - **Melhor utilização dos recursos**



Sistemas multiprogramados

- Duas inovações possibilitaram o surgimento da multiprogramação em SOs
- **Interrupções**
  - Permitem a **sinalização de eventos** no SO
- **Discos magnéticos**
  - Acesso randômico a diferentes programas no disco
  - **Melhor desempenho** em acessos de leitura e escrita

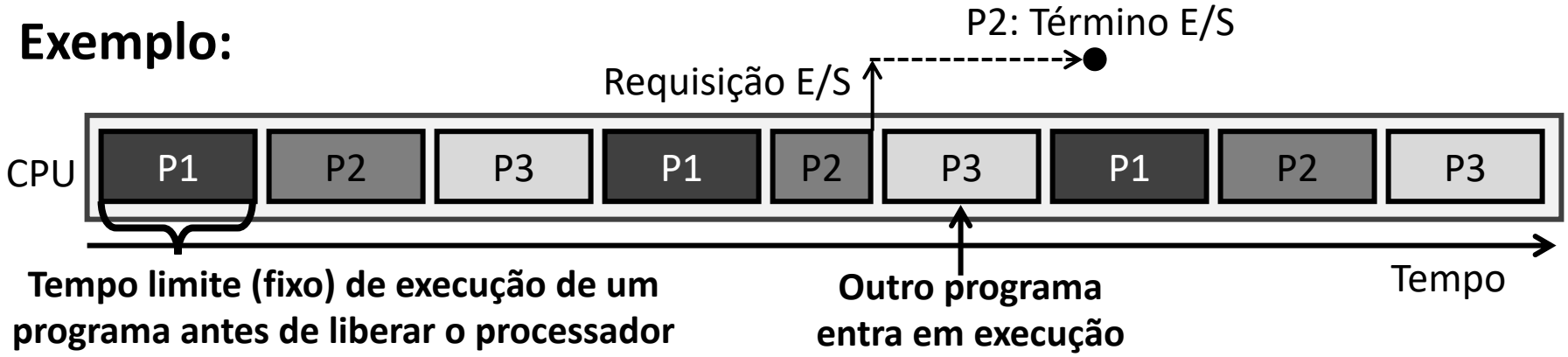
Sistemas de tempo compartilhado (timesharing)

- É um tipo de sistema multiprogramado
  - O SO alterna a execução dos programas com uma **frequência tão alta** que permite a **interação do usuário com diversos programas ao mesmo tempo**
  - Ilusão de que os programas estão sendo executados **simultaneamente**
  - **São os SOs que usamos hoje em dia!**

Sistemas de tempo compartilhado (timesharing)

- **Divisão do tempo** de uso do processador entre os usuários e programas
  - Tempo de resposta é **importante**

Exemplo:



3 Principais componentes de um SO

Componentes de um SO

- Um SO não é um bloco único e fechado de *software* executando sobre o *hardware*
  - É composto por diversos componentes com objetivos e funcionalidades específicas
- **Componentes básicos**
  1. Código de inicialização
  2. Núcleo
  3. *Drivers*
  4. Programas utilitários

1. Código de inicialização

- É o código executado na inicialização do sistema
- São realizadas **tarefas complexas**, como reconhecer os dispositivos instalados, testá-los e configurá-los adequadamente para seu uso posterior
- Outra tarefa importante é **carregar o núcleo do sistema operacional** em memória e iniciar sua execução

2. Núcleo

- É o **coração** do sistema operacional, responsável **pela gerência dos recursos do *hardware*** usados pelas aplicações
- Ele também implementa as **principais abstrações** utilizadas pelos programas aplicativos
- **Chamadas de sistema:** maneira pela qual aplicações podem requisitar serviços do SO
  - **Padrão:** POSIX (IEEE)

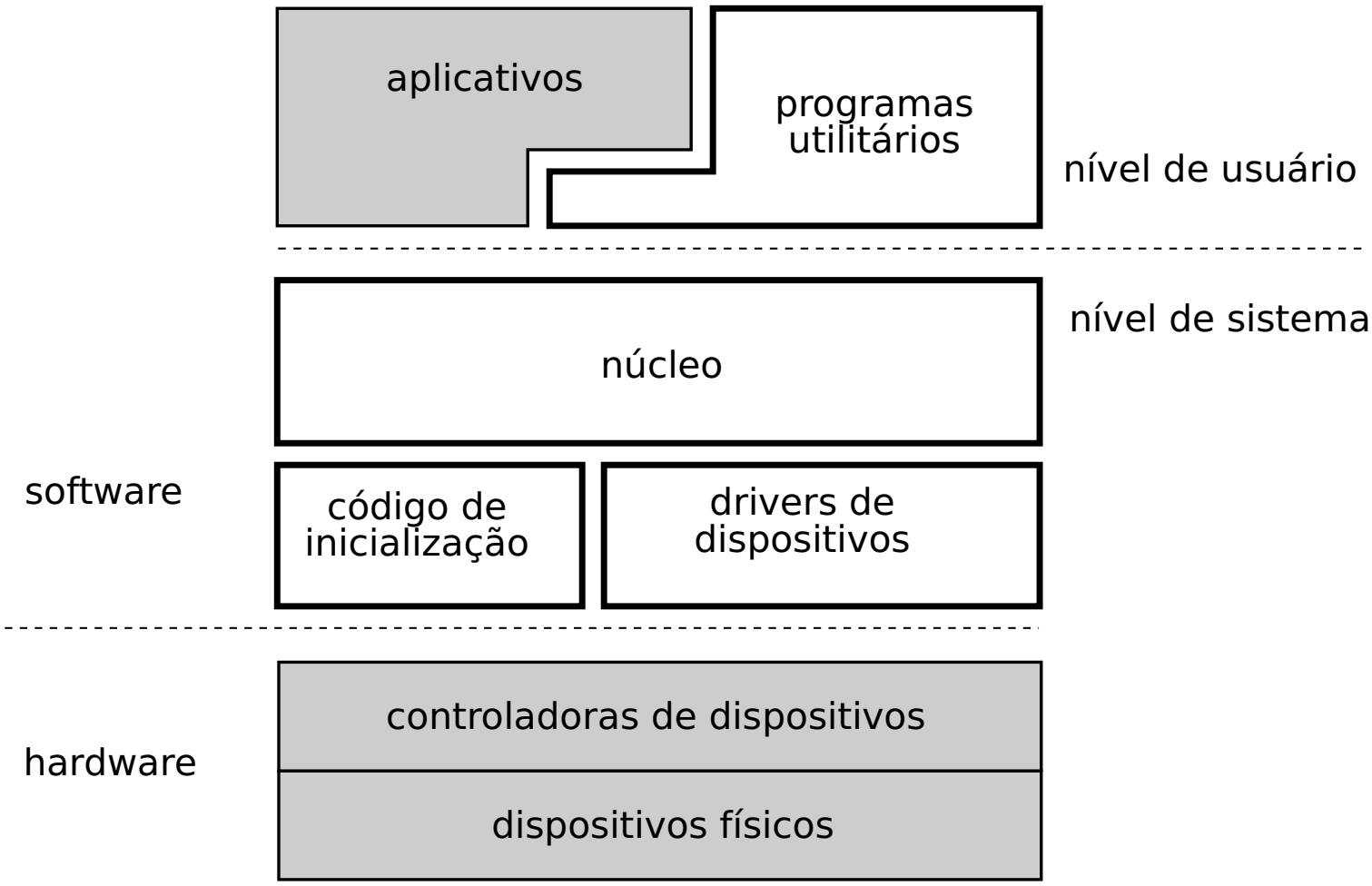
3. Drivers

- **Módulos específicos** para acessar os dispositivos físicos
- Existe um driver **para cada tipo de dispositivo**, como discos rígidos IDE, SCSI, portas USB, placas de vídeo, etc.
- Muitas vezes o driver é **construído pelo próprio fabricante** do *hardware* e fornecido em forma compilada para ser acoplado ao restante do SO

4. Programas utilitários

- São programas que facilitam o uso do sistema, fornecendo **funcionalidades complementares ao núcleo**
- **Exemplos de utilitários**
  - Formatação de discos e mídias
  - Configuração de dispositivos
  - Manipulação de arquivos
  - Interpretador de comandos (terminal)
  - Interface gráfica e gerência de janelas

Componentes de um SO



Execução de tarefas e preemptividade

- **Sistemas monotarefa vs. multitarefa**
  - **Sistema monotarefa:** somente um programa logicamente ativo no sistema
  - **Sistema multitarefa:** vários programas logicamente ativos no sistema
- **Dois tipos de sistemas multitarefa**
  - **Não preemptivos:** programas se executam do início ao fim sem serem interrompidos
  - **Preemptivos:** permite que programas em execução sejam interrompidos para que outros programas possam ser executados

Execução de tarefas e preemptividade

- **Quais sistemas são preemptivos?**
  - Sistemas em lote
  - Monitor residente
  - Sistemas multiprogramados
  - Sistemas *timesharing*

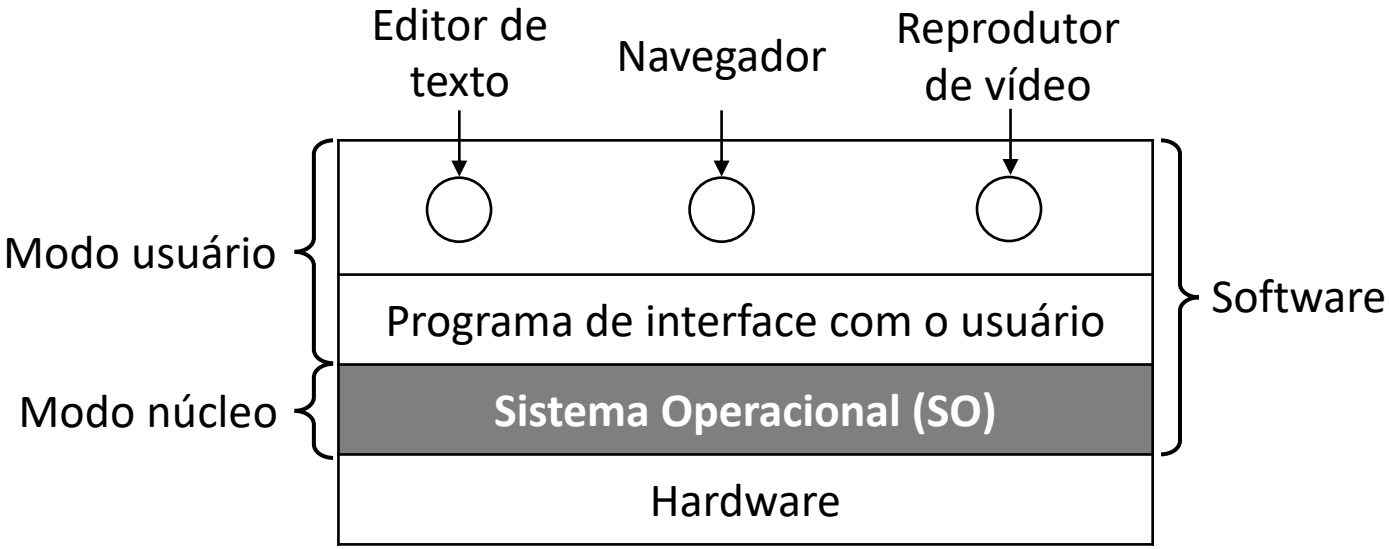
Preemptivos

4 Estruturas de núcleo

Quais são as abordagens para construção de núcleos de SOs?

Modos de operação de um processador

- **Modo núcleo (*kernel mode*)**: acesso completo ao *hardware* e qualquer instrução pode ser executada
  - SOs executam nesse modo
- **Modo usuário (*user mode*)**: apenas um subconjunto de instruções podem ser executadas
  - Programas de usuário executam nesse modo



Estruturas de núcleo

- **Principais estruturas**
  - Núcleo monolítico
  - Micronúcleo
  - Cliente-servidor
  - Máquinas virtuais

Núcleo monolítico

- É a abordagem mais comum de todas
- O SO é executado como um único programa no modo núcleo
- **Chamadas de sistema**
  - Requisição de serviços ao SO
  - Rotinas do SO com interface padronizada
  - Modo usuário → modo núcleo
- **Exemplos:** Windows, Linux e MacOS

Micronúcleo

- **SO é dividido em módulos pequenos**
  - Um módulo é executado no modo núcleo (**micronúcleo**)
  - O restante é executado como programas de usuário
- *Bugs* em módulos do SO executados em modo usuário não “quebram” o SO inteiro
- **Exemplos**: Symbian e MINIX 3

Cliente-servidor

- Pequena variação do micronúcleo
- Distinção entre:
  - **Servidores**: programas que prestam algum serviço do SO
  - **Clientes**: programas usam os serviços do SO
- Comunicação entre servidores e clientes através de trocas de mensagens

Máquinas virtuais

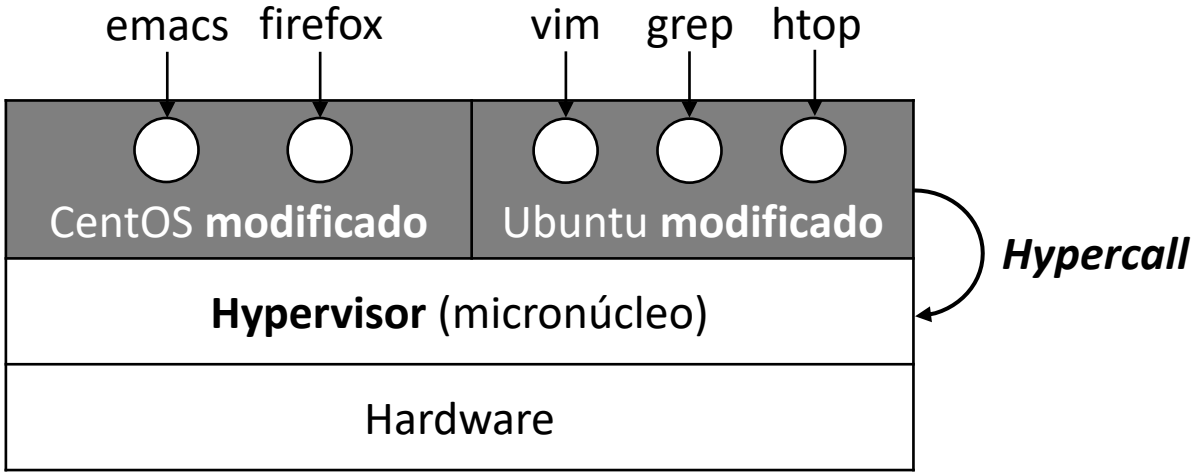
- Um ***hypervisor*** cria a ilusão de múltiplas máquinas (virtuais) no mesmo *hardware* físico
- Um computador pode ser hospedeiro de múltiplas máquinas virtuais (VMs)
- **Vantagens**
  - Uma falha em uma VM não afeta nenhuma outra
  - Baixo custo: menos máquinas físicas
  - Testes e prototipação facilitada
- **Exemplos**: VirtualBox, VMWare, Parallels e KVM

Tipos de virtualização

- **Paravirtualização**
- **Virtualização completa**
  - Hypervisor Tipo 1
  - Hypervisor Tipo 2

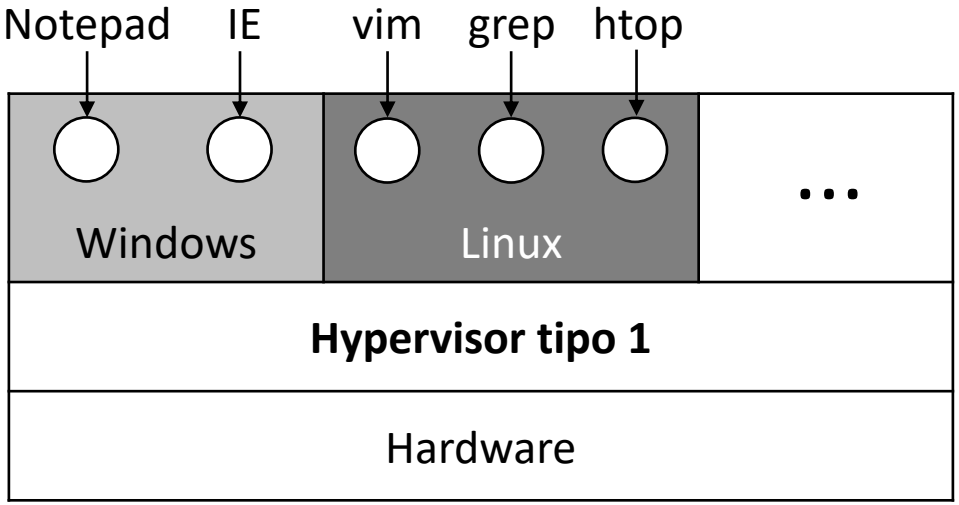
Paravirtualização

- Não cria um *hardware* virtual
  - ***Hypercalls*** permitem o SO hóspede enviar solicitações explícitas a um *hypervisor*
- O SO precisa ser modificado (mais comum) ou usar drivers de paravirtualização (menos comum)



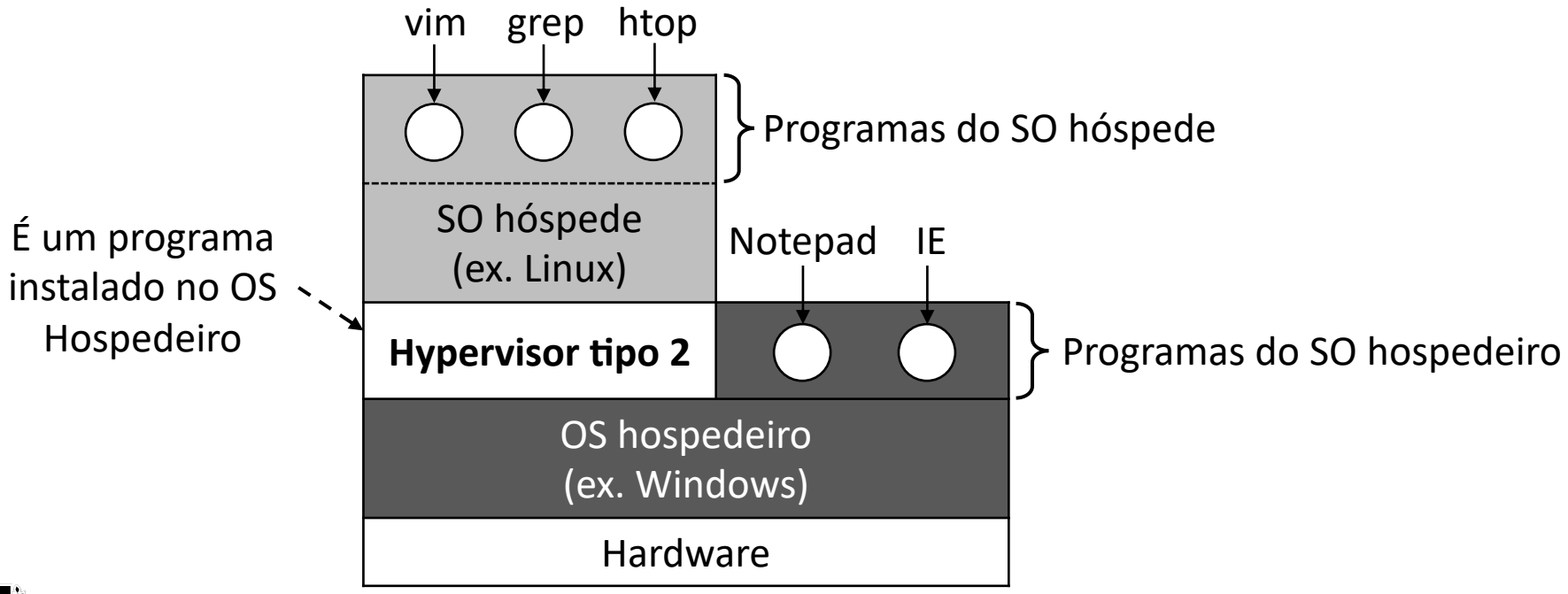
Virtualização completa: hypervisor tipo 1

- Camada de *software* de baixo nível que executa em modo privilegiado sobre o *hardware*
- Cria múltiplos *hardwares* virtuais idênticos ao *hardware* real
- SOs executam sobre o *hardware* virtual
- **Exemplos:** Xen e vSphere



Virtualização completa: hypervisor tipo 2

- O *hypervisor* tipo 2 é um programa que executa sobre um **SO hospedeiro**
  - **SO hóspede** executa sobre o *hypervisor* tipo 2
- **Exemplos:** VMWare Fusion e VirtualBox



Obrigado pela atenção!

Dúvidas? Entre em contato:

- marcio.castro@ufsc.br
- www.marciocastro.com

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

LaPeSD

Distributed Systems Research Lab

www.lapesd.inf.ufsc.br