



Infraestrutura de Software



UNIVERSIDADE
FEDERAL
DE PERNAMBUCO



Ciência da Computação :: IF677

- Professor: Carlos Ferraz <cagf>
- Monitoria: em formação
- Horários:
 - Quartas (08-10h) e Sextas, 08-10h (D-005)
- Laboratório: G1, normalmente
- URL: <http://www.cin.ufpe.br/~cagf/if677/2015-1>
- **Haverá aulas em horários extraclasse, com os monitores**



Informações relacionadas

Pré-requisitos

- Bom conhecimento em programação em alguma linguagem de alto nível (Java, C++, C etc.)
- Conhecimento desejável em linguagem de montagem (Assembly)

Beneficiários dos conhecimentos adquiridos nesta disciplina

- Programação de sistemas
- Sistemas distribuídos
- Redes
- entre outros



Dimensões do conhecimento em Infra-estruturas de Software

- Conceitual
 - Ex: concorrência
- Tecnológica:
 - aspectos técnicos de implementação e funcionalidades
- Prática:
 - laboratório, exercícios e projeto

Para qualquer profissional em computação é importante saber como uma infra-estrutura de software funciona, e não apenas como pode ser utilizada

250,06 GB FUJITSU MJA2250BH FFS G1 Media

- Macintosh HD
- SuperDrive

First Aid Erase RAID Restore


If Repair Disk is unavailable, click Verify Disk. If the disk needs repairs, you'll be given instructions for repairing the disk from the Recovery HD.

If you have a permissions problem with a file installed by the OS X installer, click Repair Disk Permissions.

Show details Clear History

Verify Disk Permissions Verify Disk

Repair Disk Permissions Repair Disk

 **Mount Point :** /
Format : Mac OS Extended (Journaled)
Owners Enabled : Yes
Number of Folders : 218.329

Capacity : 249,2 GB (249.199.591.424 Bytes)
Available : 73,91 GB (73.905.532.928 Bytes)
Used : 175,29 GB (175.294.058.496 Bytes)
Number of Files : 885.614



O Hardware

- Computador (físico)
- Programável
 - Para cada máquina um conjunto diferente de instruções

Heterogeneidade



Complexidade

The screenshot shows the Windows Task Manager Performance tab. The 'Processes' tab is selected and circled in red. The 'CPU Usage' section shows 1% usage. The 'Memory' section shows 380 MB usage. The 'Physical Memory (MB)' table shows 2046 Total, 168 Cached, 1665 Available, and 1504 Free. The 'System' table shows 11029 Handles, 569 Threads, 37 Processes, 0:00:02:26 Up Time, and 389 / 4394 Commit (MB). The 'Processes: 37' status bar is circled in red, and the 'CPU Usage: 1%' and 'Physical Memory: 18%' status bars are circled in green.

Physical Memory (MB)	
Total	2046
Cached	168
Available	1665
Free	1504

System	
Handles	11029
Threads	569
Processes	37
Up Time	0:00:02:26
Commit (MB)	389 / 4394

hardware

redes

Infra-estruturas de Suporte

a

Usuários / Programas de Usuários



Infra-estrutura de Software

sistemas operacionais, middleware

Infra-estrutura de Hardware



Infra-estrutura de Comunicação





Infraestruturas de Software

- Um **sistema operacional** é a “**ferramenta**” mais básica (*middleware* nem tanto) de qualquer sistema de computação, e
- é importante para qualquer profissional em computação entender como ele(s) funciona(m)
- O entendimento dos conceitos envolvidos na construção de um sistema operacional (e um pouco de *middleware*) permite
 - o melhor entendimento dos mecanismos e
 - ferramentas disponíveis para o usuário,para que este possa fazer o **uso mais adequado dos recursos do sistema**

Sistemas operacionais / *middleware* visam gerenciar a operação de computadores de modo a oferecer a seus usuários flexibilidade, eficiência, segurança, transparência e compartilhamento de recursos

Sistemas operacionais / *middleware* visam gerenciar a operação de computadores de modo a oferecer a seus usuários **flexibilidade**, eficiência, segurança, transparência e compartilhamento de recursos

Sistemas operacionais / *middleware* visam gerenciar a operação de computadores de modo a oferecer a seus usuários flexibilidade, **eficiência**, segurança, transparência e compartilhamento de recursos

Sistemas operacionais / *middleware* visam gerenciar a operação de computadores de modo a oferecer a seus usuários flexibilidade, eficiência, segurança, transparência e compartilhamento de recursos

Sistemas operacionais / *middleware* visam gerenciar a operação de computadores de modo a oferecer a seus usuários flexibilidade, eficiência, segurança, transparência e compartilhamento de recursos

Sistemas operacionais / *middleware* visam gerenciar a operação de computadores de modo a oferecer a seus usuários flexibilidade, eficiência, segurança, transparência e **compartilhamento de recursos**

Visão ampla: grande porte, desktops, tablets, celulares, TV etc.

Sistemas operacionais / *middleware* visam gerenciar a operação de computadores de modo a oferecer a seus usuários flexibilidade, eficiência, segurança, transparência e compartilhamento de recursos

4 grupos básicos: processamento, memória, armazenamento (arquivos), entrada e saída



- Existe aqui um programa (PowerPoint) rodando,
 - usando o **processador** da máquina,
 - ...a **memória**,
 - ...manipulando um **arquivo** armazenado no **disco**,
 - ...aparecendo na **tela**,
 - ...recebendo comandos, via **teclado**

Como se faz?



Um Sistema Operacional...

- [é um conjunto de programas que] **visa esconder as peculiaridades do hardware**



Máquina mais fácil de ser utilizada, mais amigável e mais segura



- [é um conjunto de programas que] **gerencia os recursos disponíveis**
 - processo/processador
 - memória
 - disco/arquivos
 - outros dispositivos de entrada/saída – teclado, tela, mouse etc.



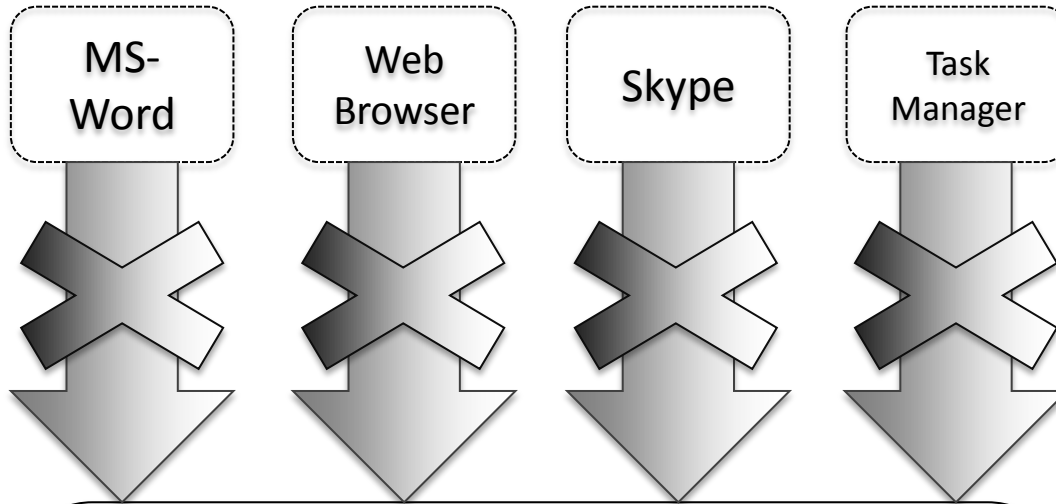
Eficiência,
compartilhamento e
resolução de
possíveis conflitos

- Gerência de processo
- Gerência de memória
- Gerência de disco/
armazenamento – Sistema de
Arquivos
- Gerência de entrada/saída

Usuários

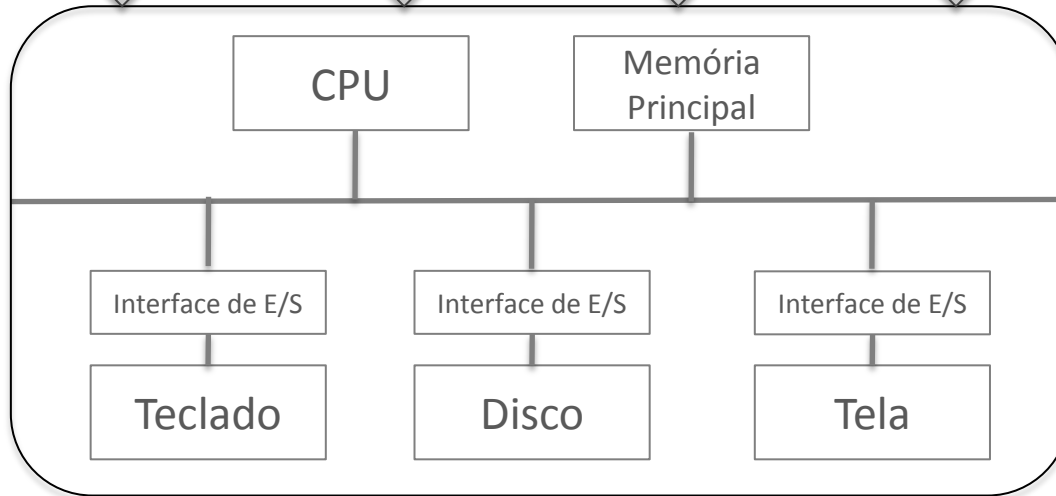


Aplicações de software



Aplicações de software **NÃO** usam o hardware diretamente

Arquitetura de hardware de um computador típico



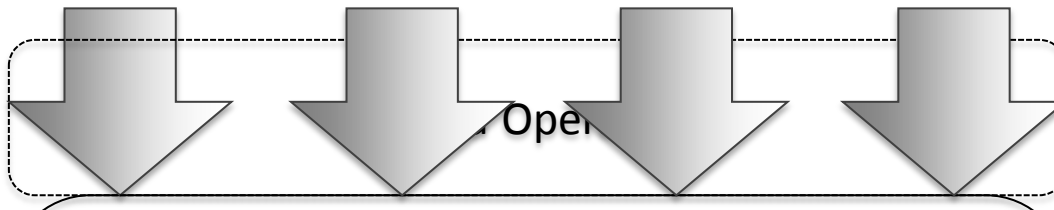
Usuários



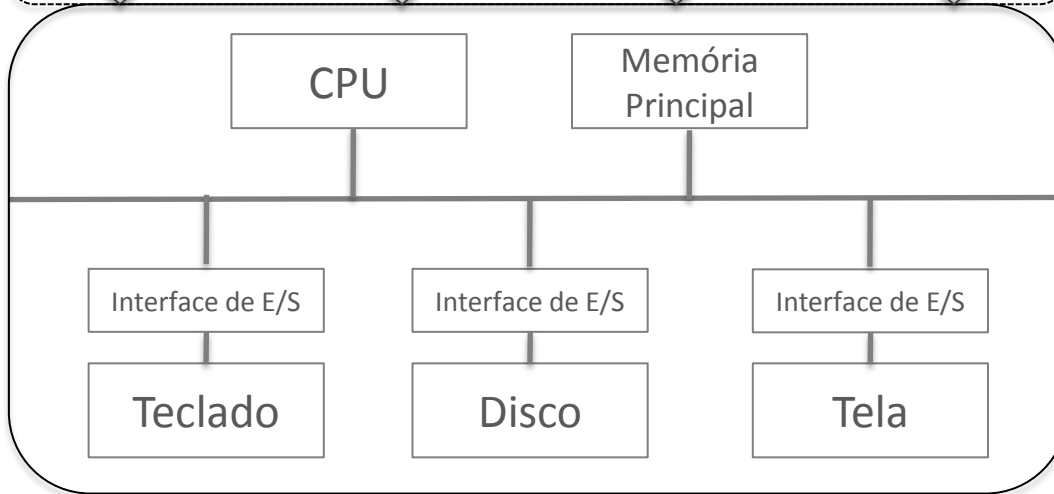
Aplicações de software



Infra-estrutura de software



Arquitetura de hardware de um computador típico



Software é abstrato

Abstração de hardware e **compartilhamento** de recursos

proteção, eficiência etc.

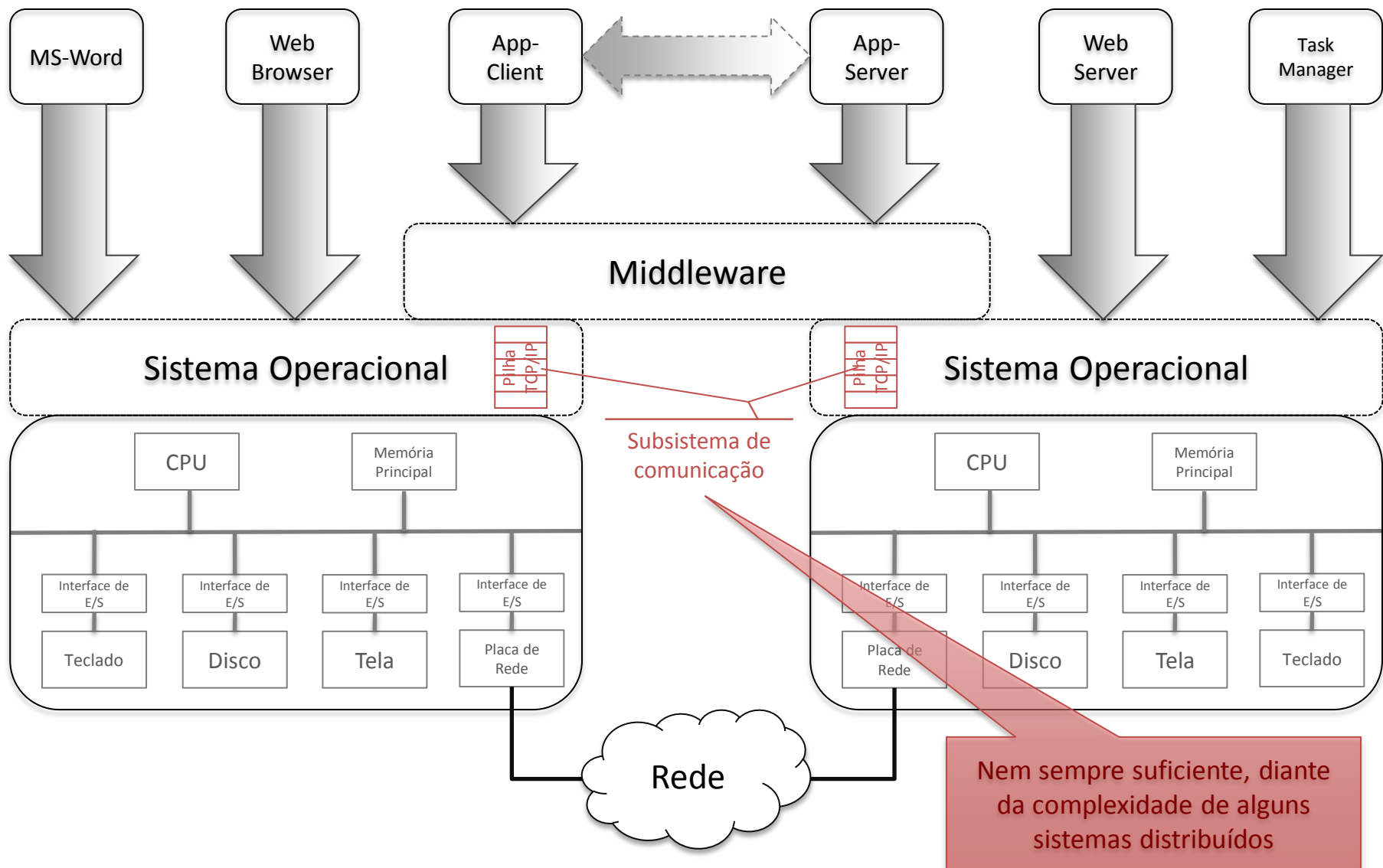
Hardware é concreto (físico)



- E se o sistema for distribuído em rede?
 - Ex.: Web browser e servidor
- ... É preciso gerenciar também recursos de rede/**comunicação**, entre outras coisas

Um middleware...

- [é um conjunto de serviços que] **dá suporte a sistemas de software distribuídos**



Nem sempre suficiente, diante da complexidade de alguns sistemas distribuídos



Ao final do curso você **deverá** ser capaz de...

- **Explicar** o funcionamento de um SO
 - Dos pontos de vista de: mecanismo de abstração e gerenciamento de recursos
- **Aplicar** vários dos conceitos discutidos, como processos, *threads*, concorrência, interrupções e escalonamento, no **desenvolvimento** de aplicações do mundo real
- **Usar** infra-estruturas existentes para computação [incl. distribuída]



...E não deverá ser capaz de

- **Projetar** um novo sistema operacional
- **Implementar** um novo sistema operacional
- **Estender** um sistema operacional existente
- **<os mesmos verbos>** uma plataforma de *middleware*
- Existem disciplinas mais apropriadas para isso:
 - IF709 -IMPLEMENTAÇÃO SIST. OPERACIONAIS
 - IF749 -TÓPICOS AVANÇ. SIST. DISTRIBUÍDOS



Avaliação

- Provas (EE1 e EE2)
- Projeto (EE3) – **em duplas!**
 - Lista de exercícios sobre programação concorrente
- Nota Final = $(EE1 + EE2 + EE3) / 3$



Material de Estudo

- Transparências das aulas
 - <http://www.cin.ufpe.br/~cagf/if677/2015-1/>
- Livros
 - Parte I: Sistemas Operacionais Modernos – 2ª Edição. A. Tanenbaum, 2003
 - Opção: Modern Operating Systems 3e. Prentice-Hall, 2008 (Já em Português, edição 2010)
 - Parte II:
 - Distributed Systems: Principles and Paradigms. Andrew Tanenbaum, Maarten van Steen. Prentice-Hall, 2002
 - Distributed Systems: Concepts and Design (4th / 5th Edition). George Coulouris, Jean Dollimore, Tim Kindberg. Addison-Wesley, 2005 / 2011

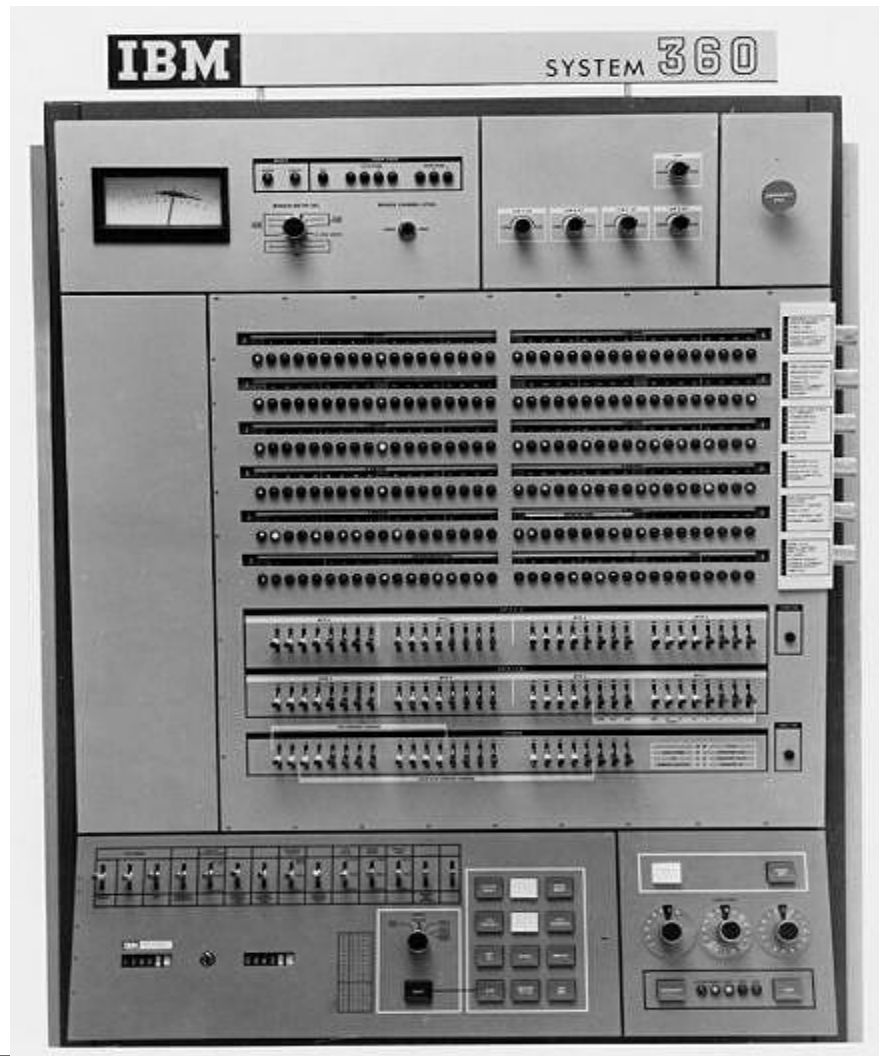


Conclusão

- Sistema Operacional
 - Mecanismo de abstração dos dispositivos subjacentes
 - Gerenciador de recursos
- Middleware
 - Plataforma de suporte de valor agregado a sistemas distribuídos



IBM System 360 Console

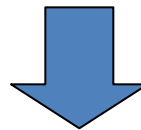




Computador Moderno

- Componentes físicos (**hardware**)
 - Um ou mais processadores
 - Memória
 - Discos
 - Impressoras
 - Vários outros dispositivos de E/S (tela, mouse...)

**Um Sistema
Complexo!!!**



- Gerenciar todos **estes** componentes requer **abstração** – *um modelo mais simples do computador*
- É isso que é o **sistema operacional**

Concreto

Tangível

Hardware

hardware

Abstrato

Intangível

Software

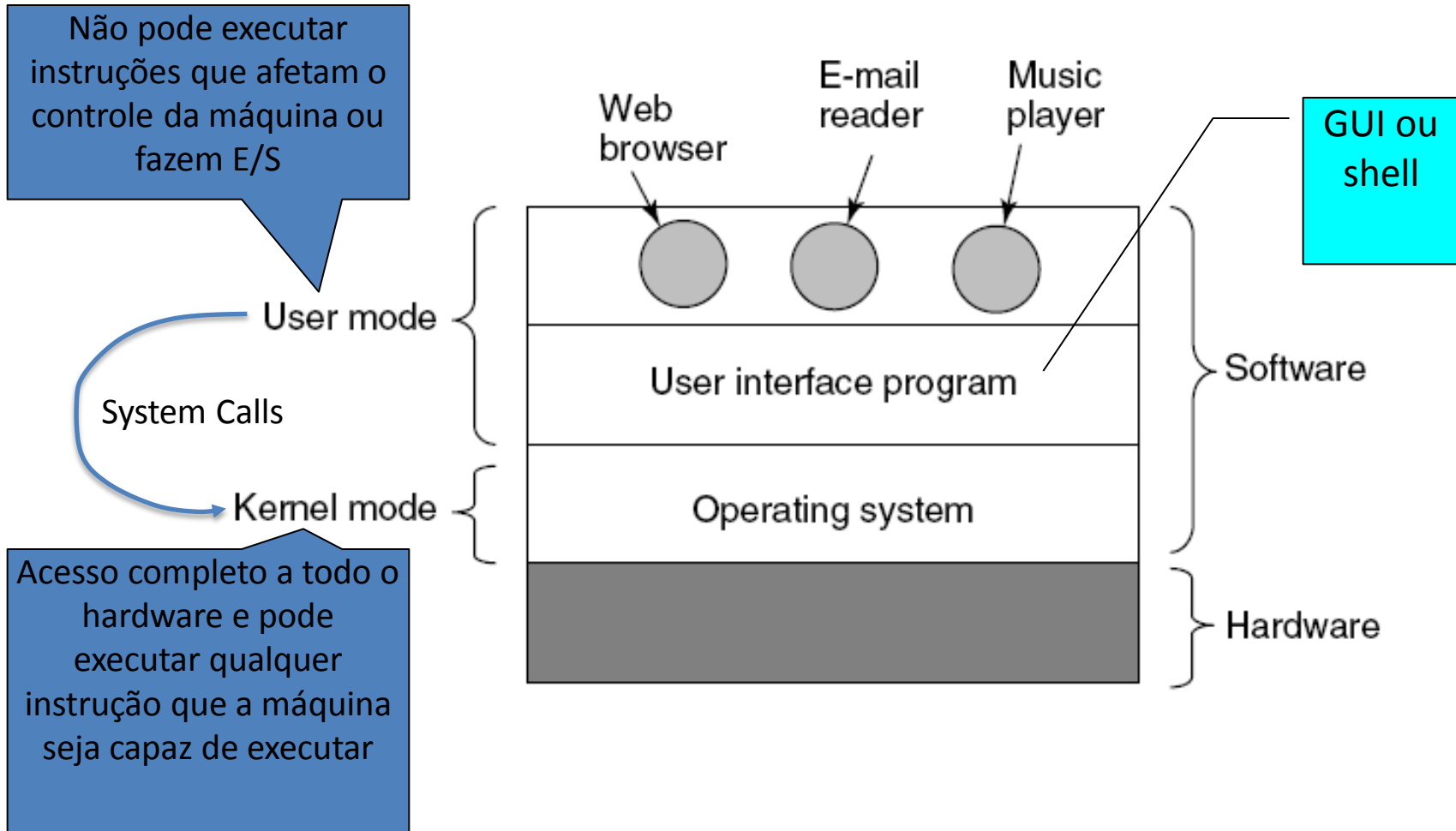
software

COMPLEXIDADE





Sistema Computacional em Camadas



Sistema Operacional

Sistema Operacional

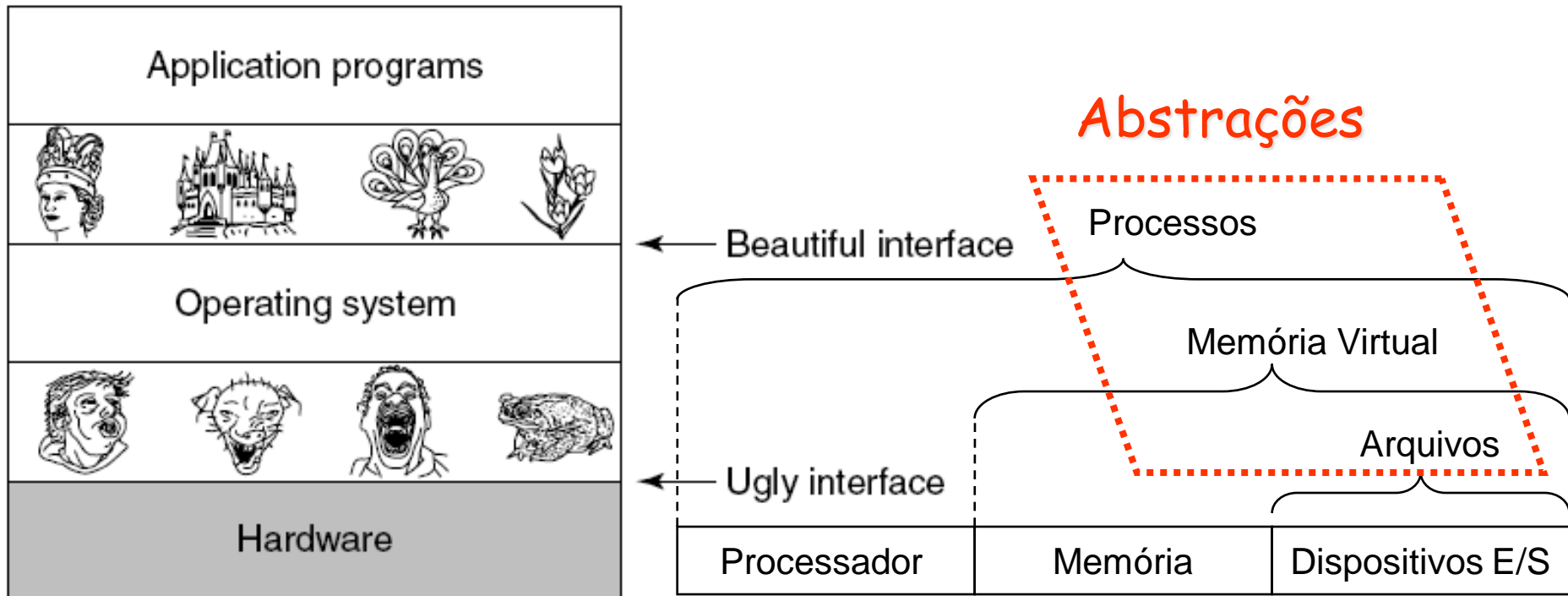
Máquina Abstrata

Gerenciador de Recursos



Máquina Estendida

- Sistemas operacionais tornam o hardware pouco atraente em abstrações mais interessantes





Gerenciador de Recursos

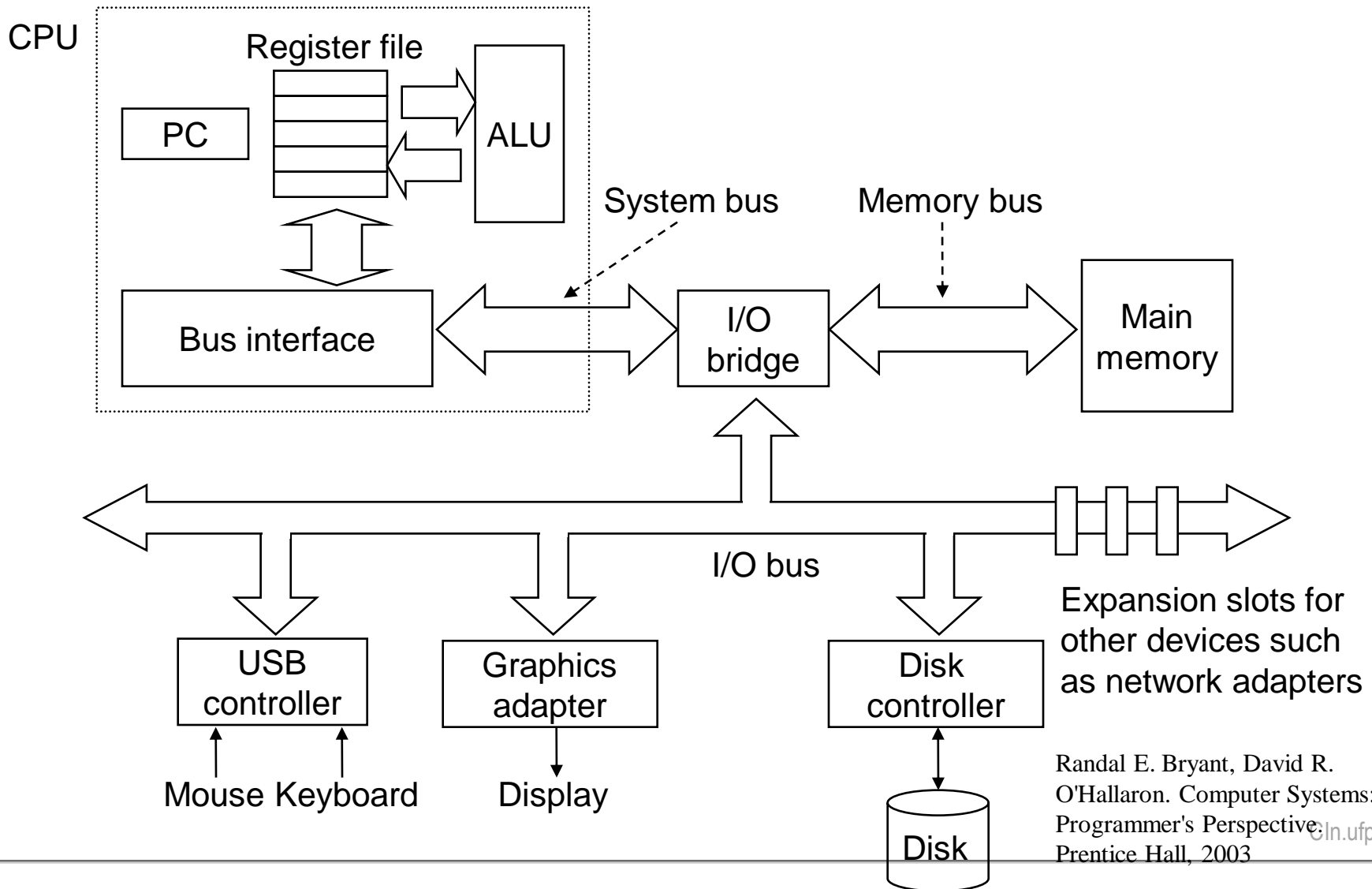
- **Gerencia** e **protege** memória, dispositivos de E/S e outros recursos (hardware)
- Permite o compartilhamento (multiplexação) de recursos
 - no tempo (time-sharing)
 - Ex.: múltiplos programas compartilham o processador (executam) ao mesmo tempo
 - no espaço
 - Ex.: dados de diferentes usuários/arquivos compartilham o espaço em disco

Hardware

Hardware



Um pouco de um computador típico

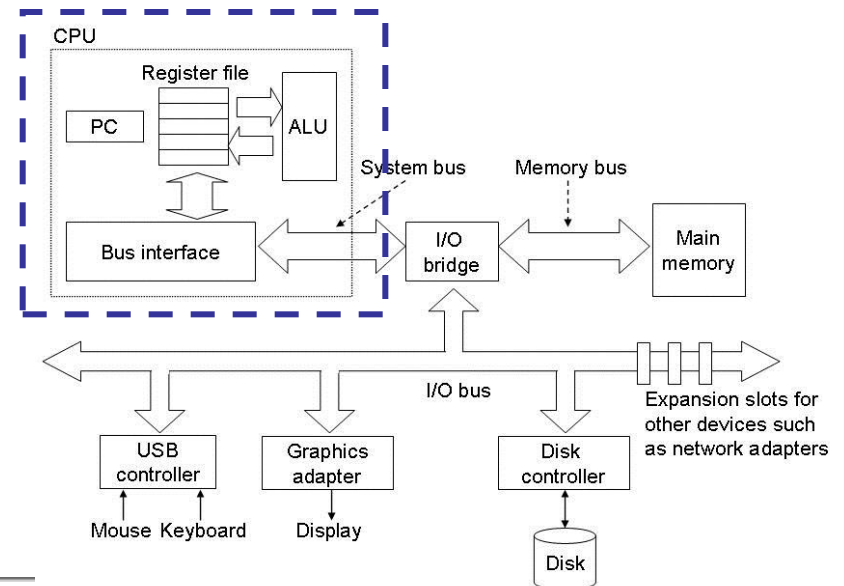




CPU: Central Processing Unit

- Unidade de Controle
- ALU: Unidade Aritmética e Lógica
- Registradores
 - Funcionam como **memória** de acesso **extremamente rápida**
 - **Baixa capacidade** de armazenamento
 - Funções específicas
 - Exemplos de registradores
 - PC (program counter): contém o endereço da próxima instrução a ser executada
 - Instruction register: onde é copiada cada instrução a ser executada

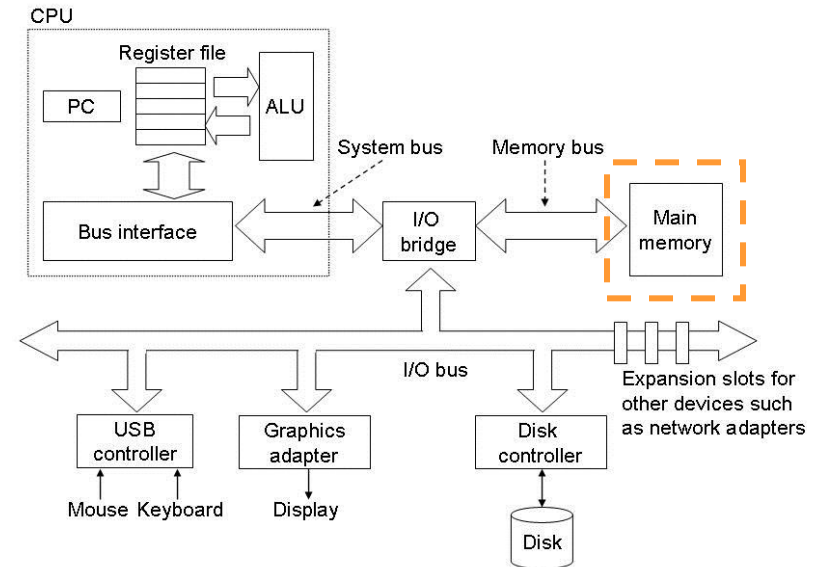
- A CPU, seguidamente, executa instruções requisitadas à memória
 - Ciclo *fetch-decode-execute*:
 1. busca instrução na memória
 2. atualiza PC
 3. decodifica instrução
 4. executa instrução





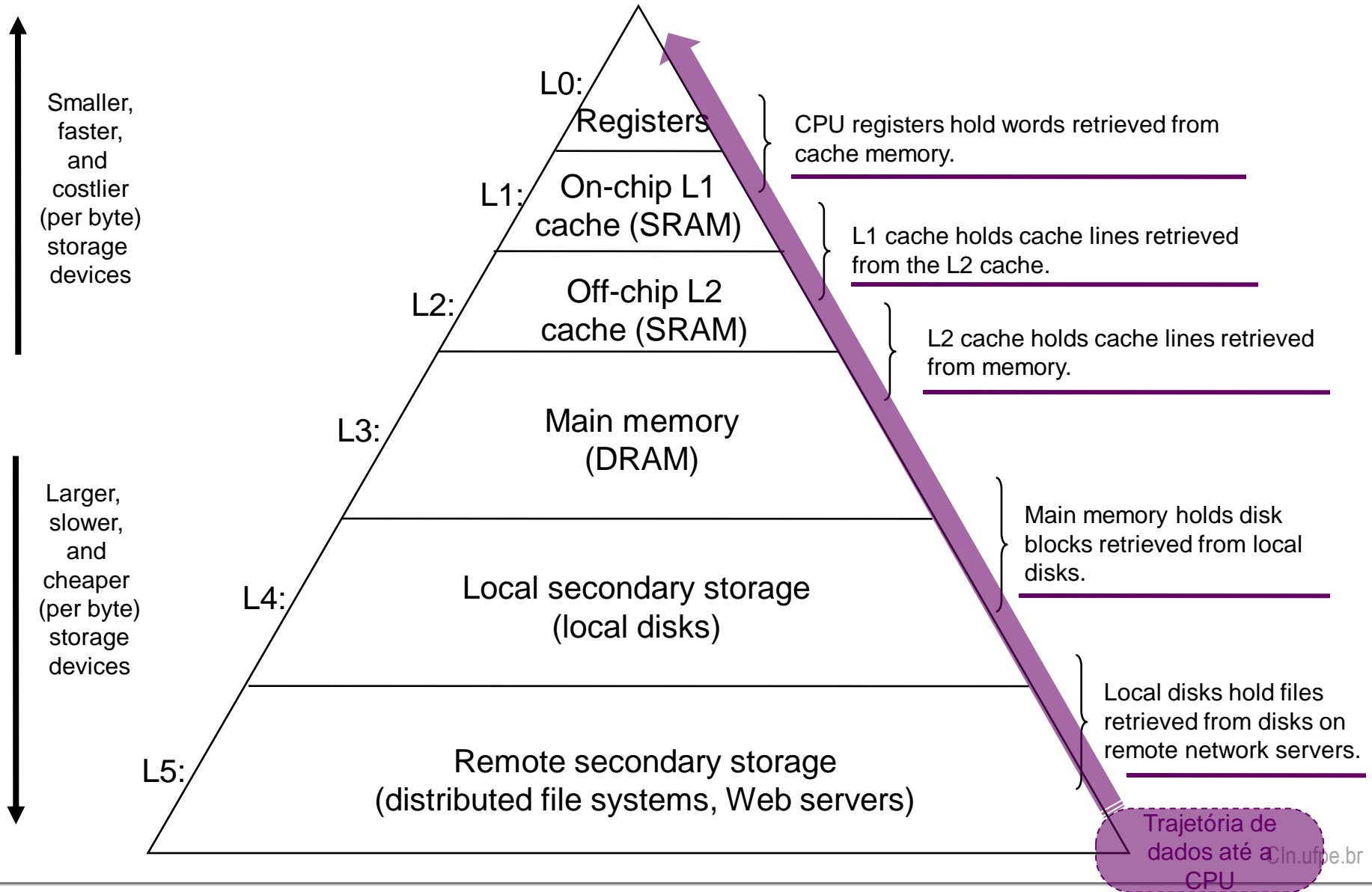
Memória

- Logicamente, a memória principal corresponde a um enorme vetor (array) de bytes
 - cada posição tem um endereço único (índice do vetor)
- Os registradores da CPU muitas **vezes** são usados para armazenar endereços de memória
 - Assim, o número de bits em cada registrador **limita** o número de **posições de memória endereçáveis**
 - Ex.: 8 bits → 256 posições...





Hierarquia de Memória



Software Básico

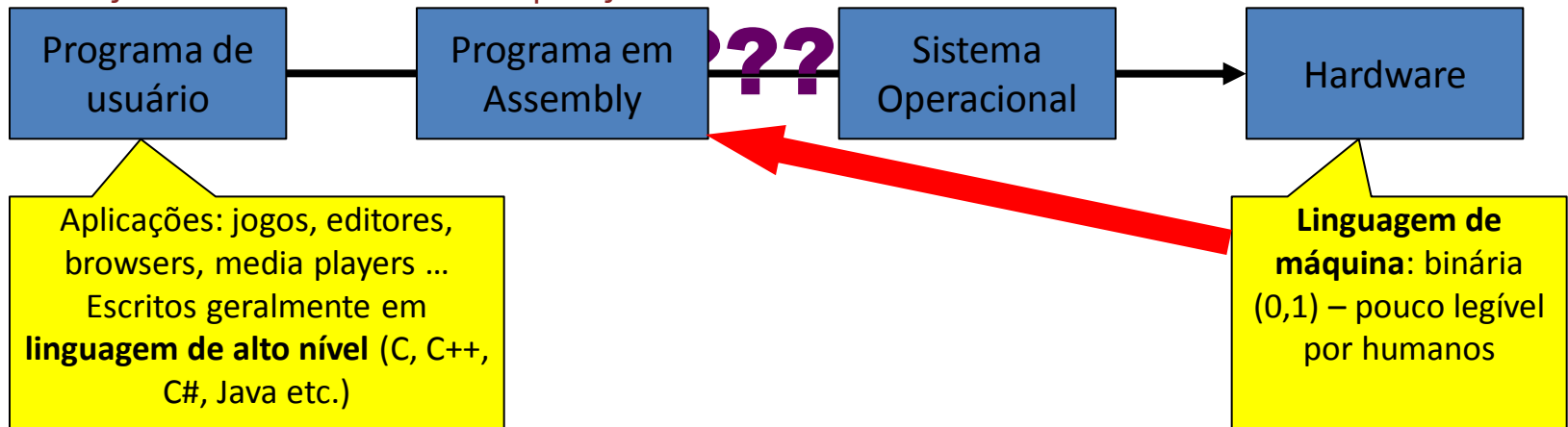
SOFTWARE BÁSICO



Software Básico

[A. Raposo e M. Endler, PUC-Rio, 2008]

- “Conhecendo mais sobre o que está ‘por baixo’ do programa, você pode escrever programas mais eficientes e confiáveis”
- Abstrações em um sistema de computação:



- A linguagem de montagem (Assembly) é um mapeamento direto da linguagem de máquina, mas que introduz várias “facilidades” (ou “menos dificuldades”) para o programador
 - usa “apelidos” das instruções de máquina, mais fáceis de lembrar do que seu valor hexadecimal exigido pelo processador
 - Ex.: `mov eax, edx`

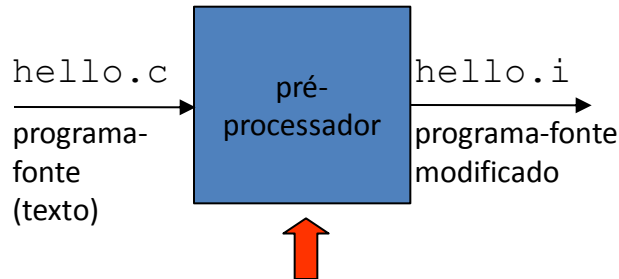
└─ move o que está no registrador de dados para o acumulador



Gerando um executável

- `unix> gcc -o hello hello.c`

```
1. #include <stdio.h>
2. int main()
3. {
4.     printf("hello, world\n");
5. }
```



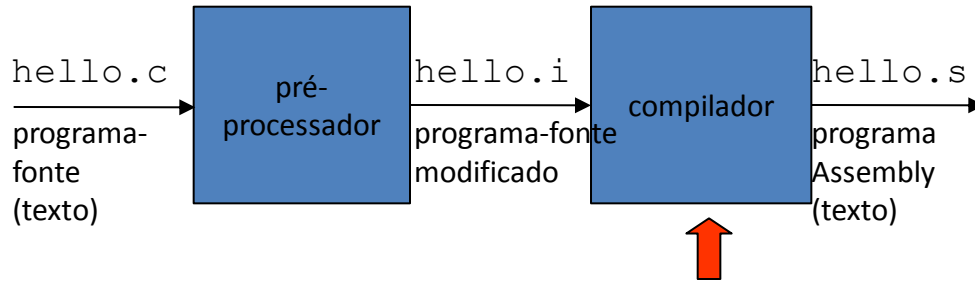
- Modifica o programa em C de acordo com diretivas começadas com #
 - Ex.: `#include <stdio.h>` diz ao pré-processador para ler o arquivo `stdio.h` e inseri-lo no programa fonte
- O resultado é um programa expandido em C, normalmente com extensão `.i`, em Unix



Gerando um executável

- `unix> gcc -o hello hello.c`

```
1. #include <stdio.h>
2. int main()
3. {
4.     printf("hello, world\n");
5. }
```



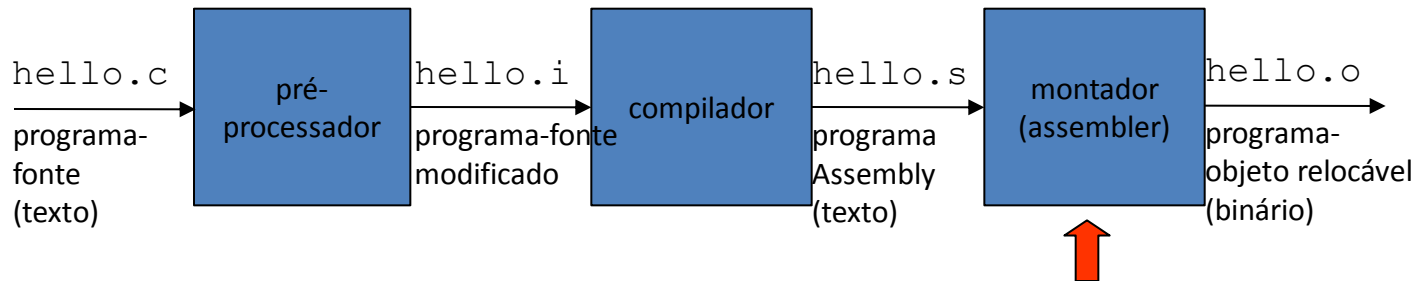
- Compilador traduz o programa .i em um programa em Assembly
 - É o formato de saída comum para os compiladores nas várias linguagens de programação de alto nível
 - i.e., programas em C, Java, Fortran, etc vão ser traduzidos para a mesma linguagem Assembly



Gerando um executável

- `unix> gcc -o hello hello.c`

```
1. #include <stdio.h>
2. int main()
3. {
4.     printf("hello, world\n");
5. }
```



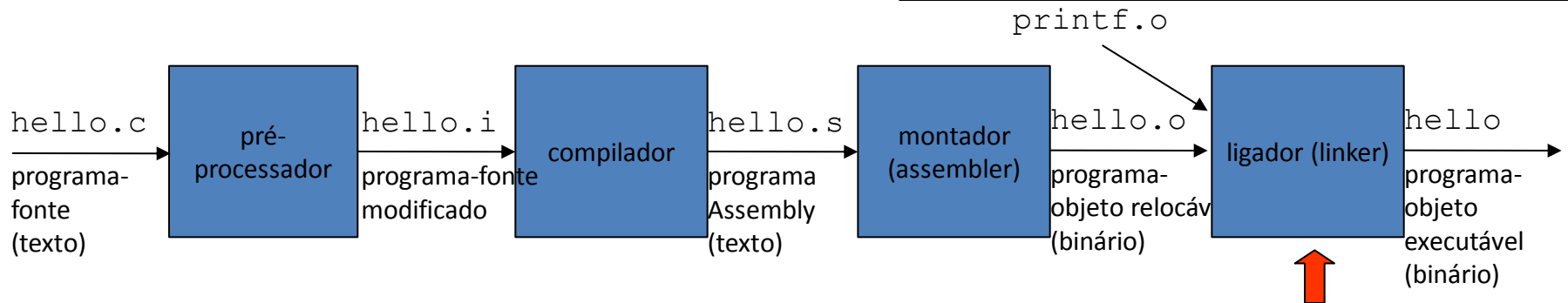
- Montador (Assembler) transforma o programa em Assembly em um programa binário em linguagem de máquina (chamado programa-objeto)
 - Os módulos de programas, compilados ou montados, são armazenados em um formato intermediário (“*Programa-Objeto Relocável*” – extensão `.o`)
- Endereços de acesso e a posição do programa na memória ficam **indefinidos**



Gerando um executável

- `unix> gcc -o hello hello.c`

```
1. #include <stdio.h>
2. int main()
3. {
4.     printf("hello, world\n");
5. }
```



- O ligador (linker) gera o programa executável a partir do .o gerado pelo assembler
 - No entanto, pode haver funções-padrão da linguagem (ex., `printf`) que não estão definidas no programa, mas em outro arquivo .o pré-compilado (`printf.o`)
 - O ligador faz a junção dos programas-objeto necessários para gerar o executável

Execução

EXECUÇÃO

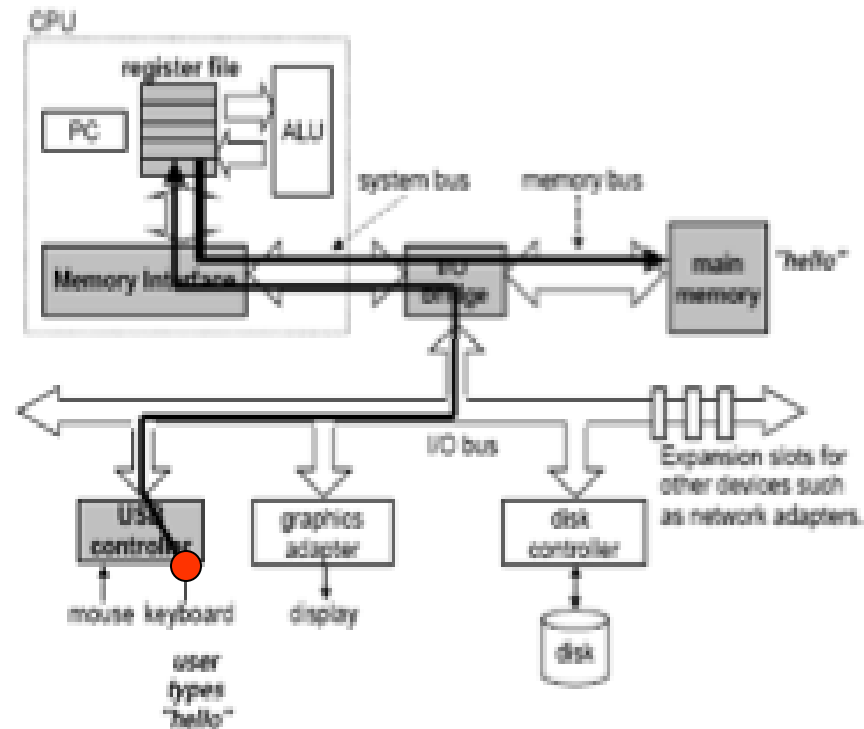
Como acontece...



Processo

Conceito: Um programa em execução

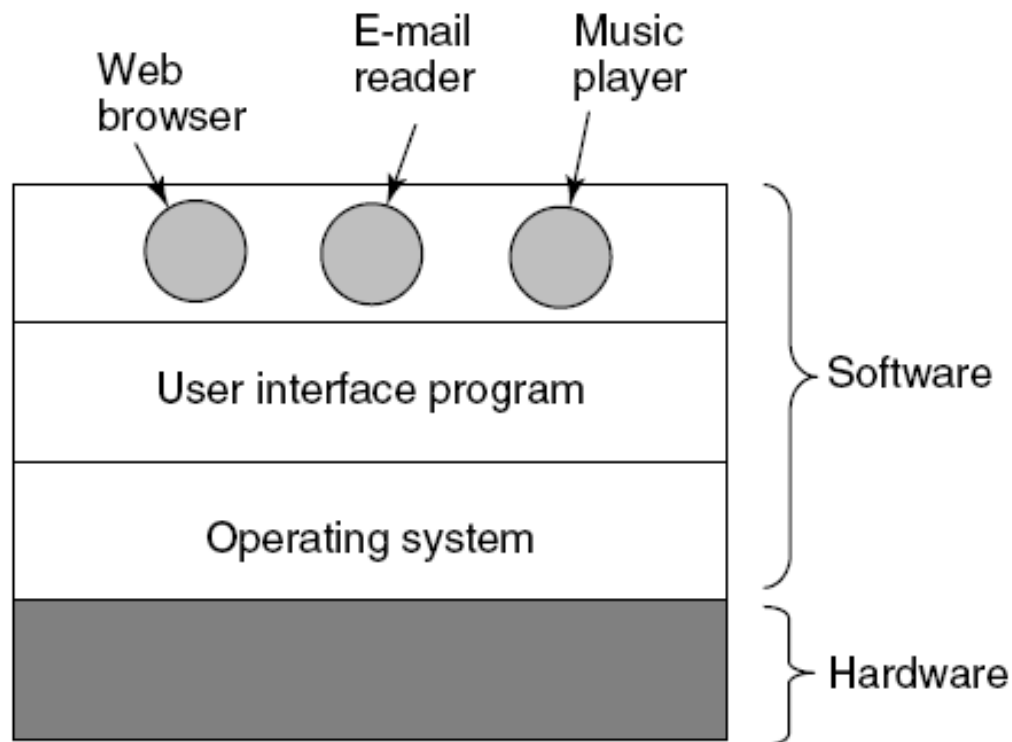
Ao digitar “hello”, os caracteres são passados para um registrador e depois para memória principal





Mais de um programa em execução

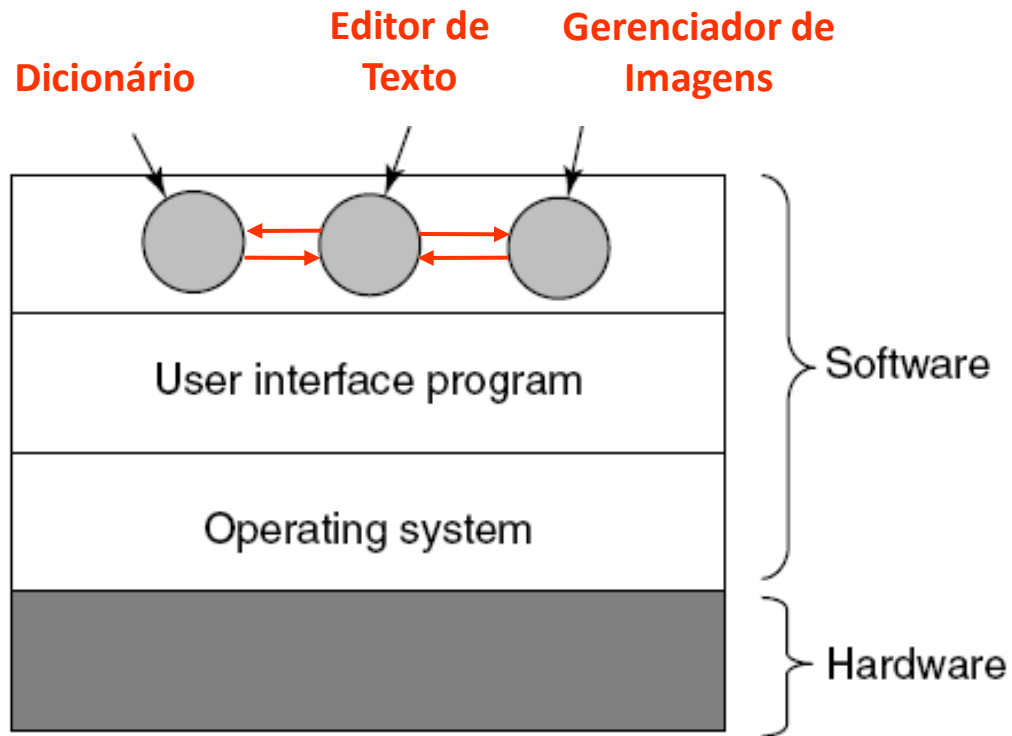
- Múltiplos processos vs. um (ou [poucos] mais) processador(es) ⇒ **como pode???**





Processos Comunicantes

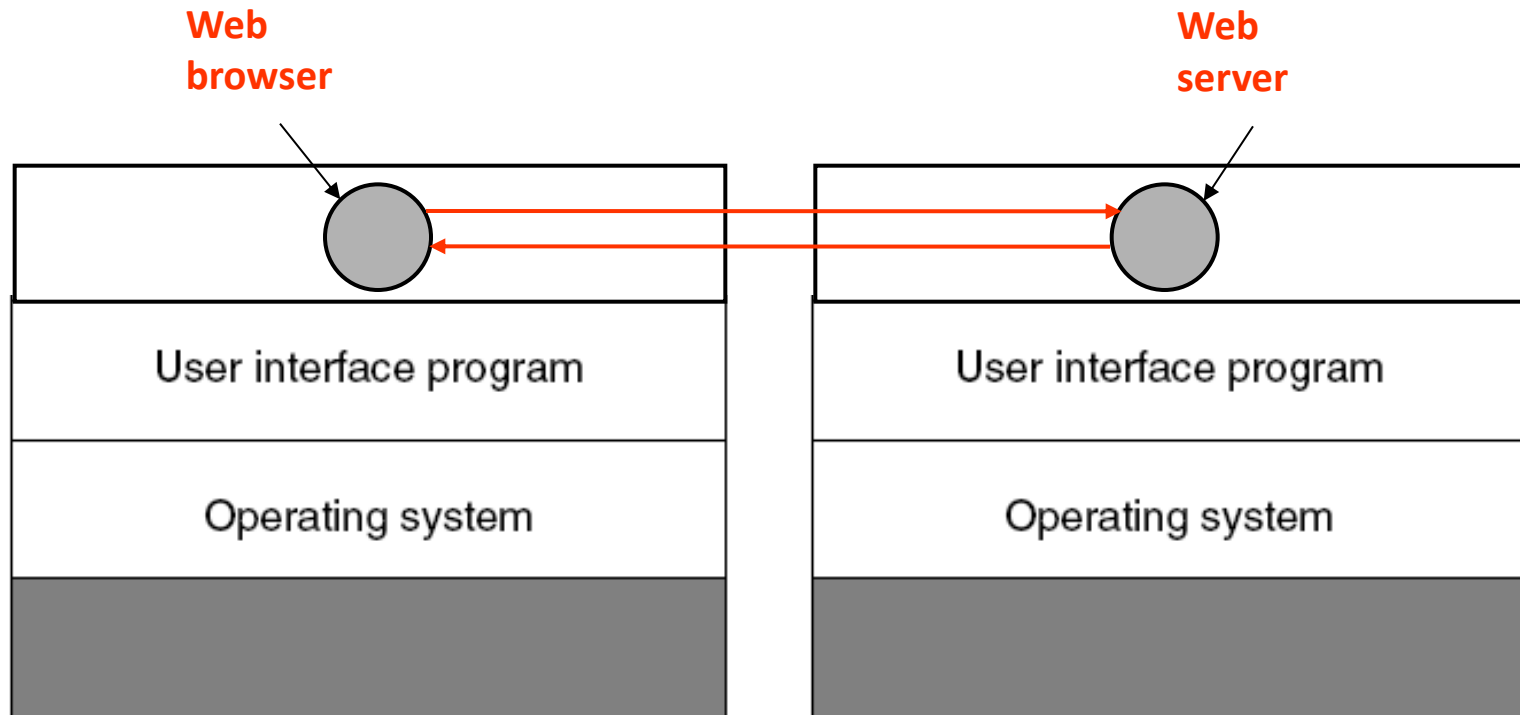
- Como pode???





Sistemas Distribuídos

- Processos em máquinas distintas e que se comunicam





Sistemas Distribuídos

Como fazer funcionar aplicações distribuídas que usam diferentes sistemas de computador (hardware), sistemas operacionais e software de aplicação (ex. linguagens de programação), interconectadas por diferentes redes?

heterogeneidade

O problema da **interoperabilidade**

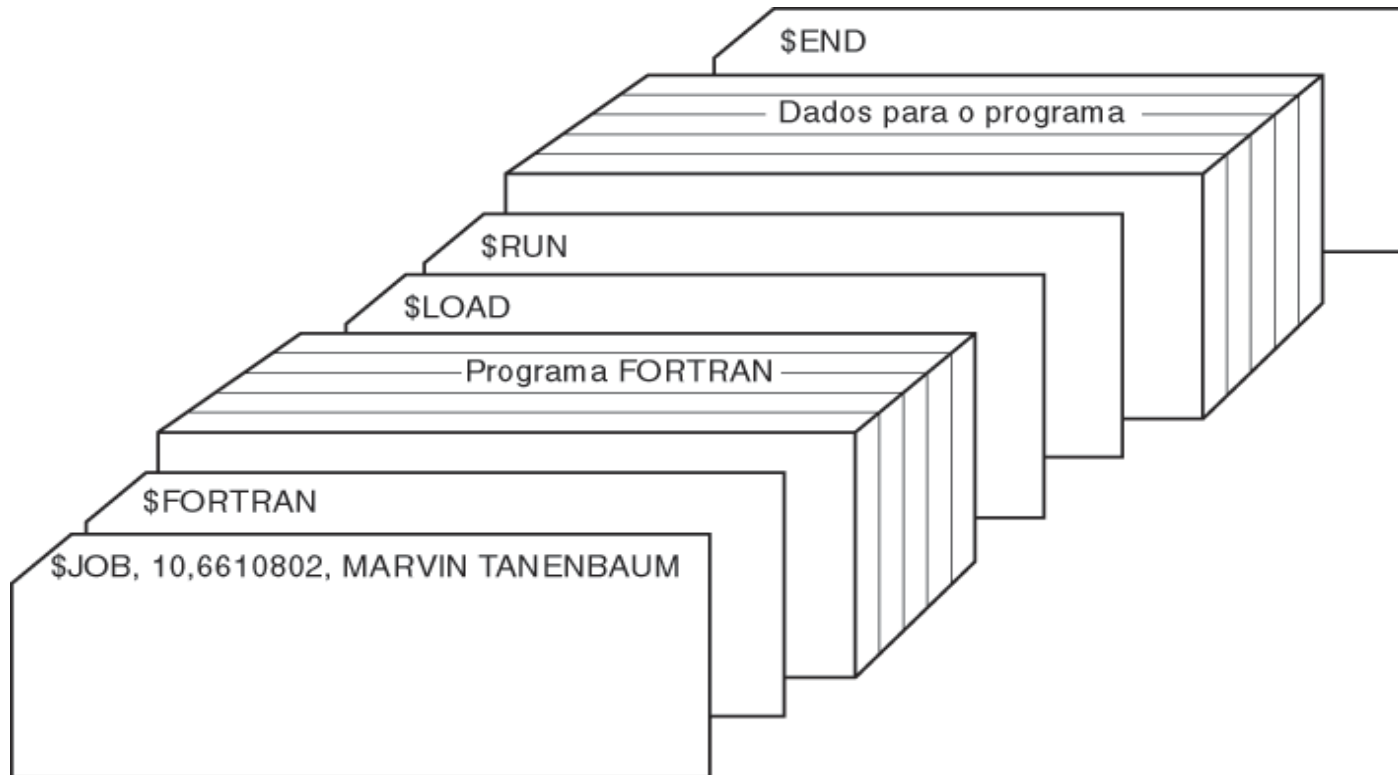


História dos Sistemas Operacionais

- Primeira geração: 1945 - 1955
 - Válvulas, painéis de programação
- Segunda geração: 1955 - 1965
 - transistores, **sistemas em lote**
- Terceira geração: 1965 – 1980
 - CIs (circuitos integrados) e **multiprogramação**
- Quarta geração: 1980 – presente
 - Computadores pessoais
- Hoje: onipresença – computação ubíqua



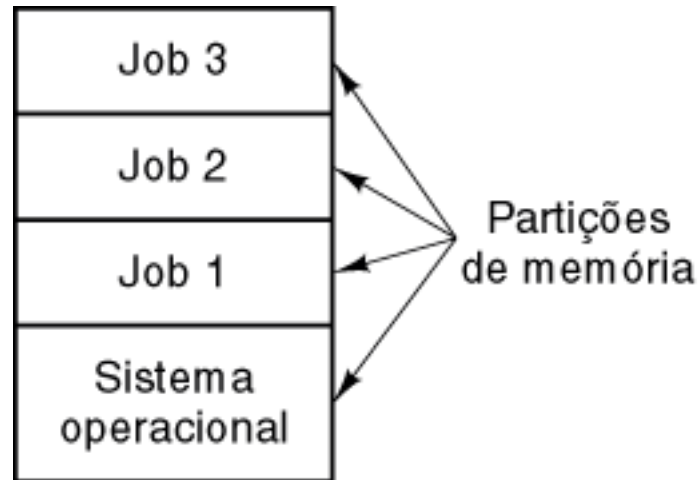
História dos Sistemas Operacionais



- Estrutura de um job típico (lote de cartões) – 2a. geração



História dos Sistemas Operacionais



- Sistema de **multiprogramação**
 - Três jobs na memória – **3a. geração**



Diversidade de Sistemas Operacionais

- Sistemas operacionais de **computadores de grande porte** (*mainframe*)
- Sistemas operacionais de servidores / **redes**
- Sistemas operacionais de **multiprocessadores** (paralelismo)
- Sistemas operacionais de computadores pessoais
- Sistemas operacionais de dispositivos portáteis/ **móveis** (ex. celulares)
- Sistemas operacionais de **tempo-real**
- Sistemas operacionais **embarcados**
- Sistemas operacionais de cartões inteligentes
- Sistemas operacionais de sensores



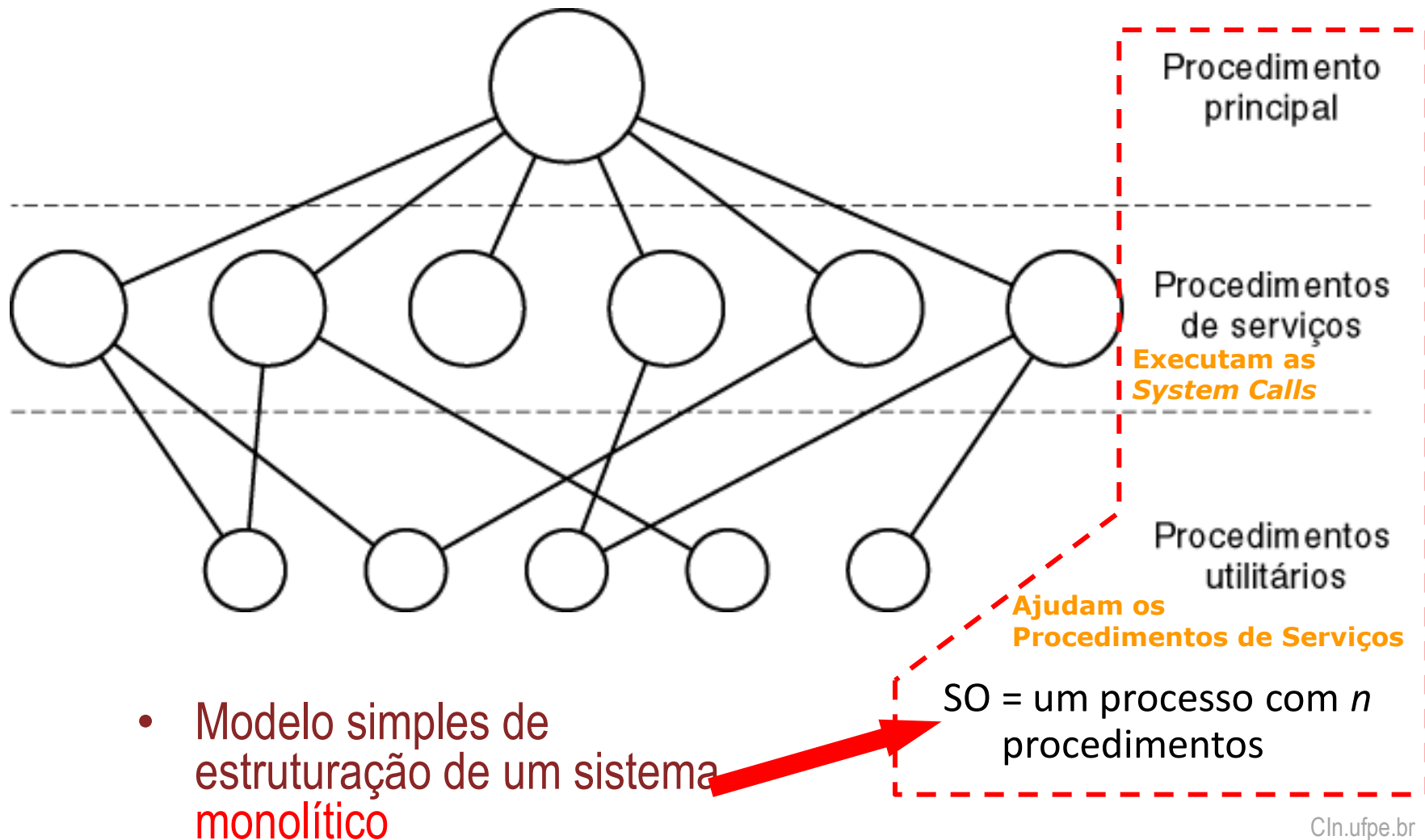
Estruturação de Sistemas Operacionais

- Monolítico
- Camadas
- Cliente-Servidor
- Virtualização



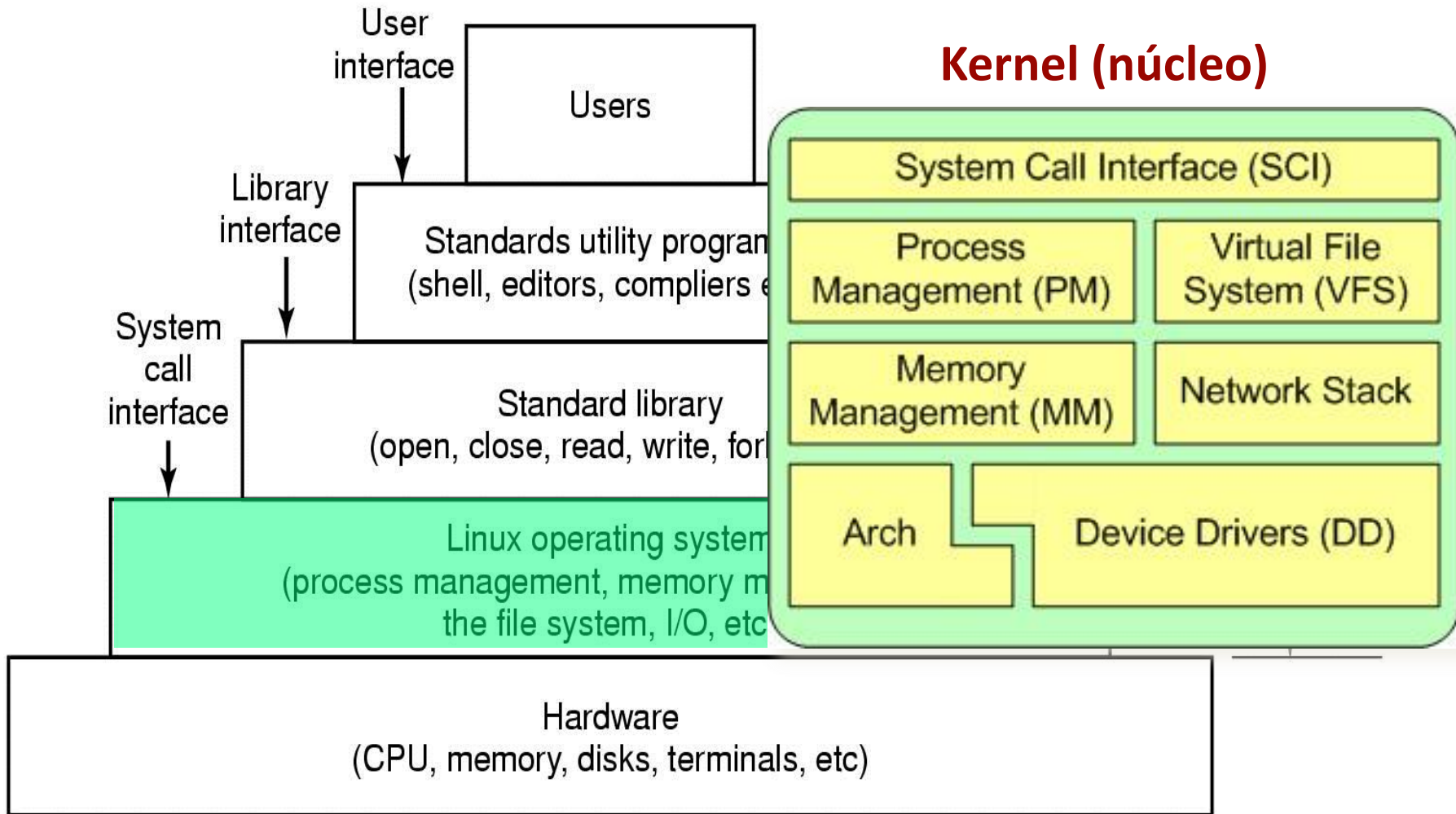
Estrutura de Sistemas

Operacionais: Sistema Monolítico



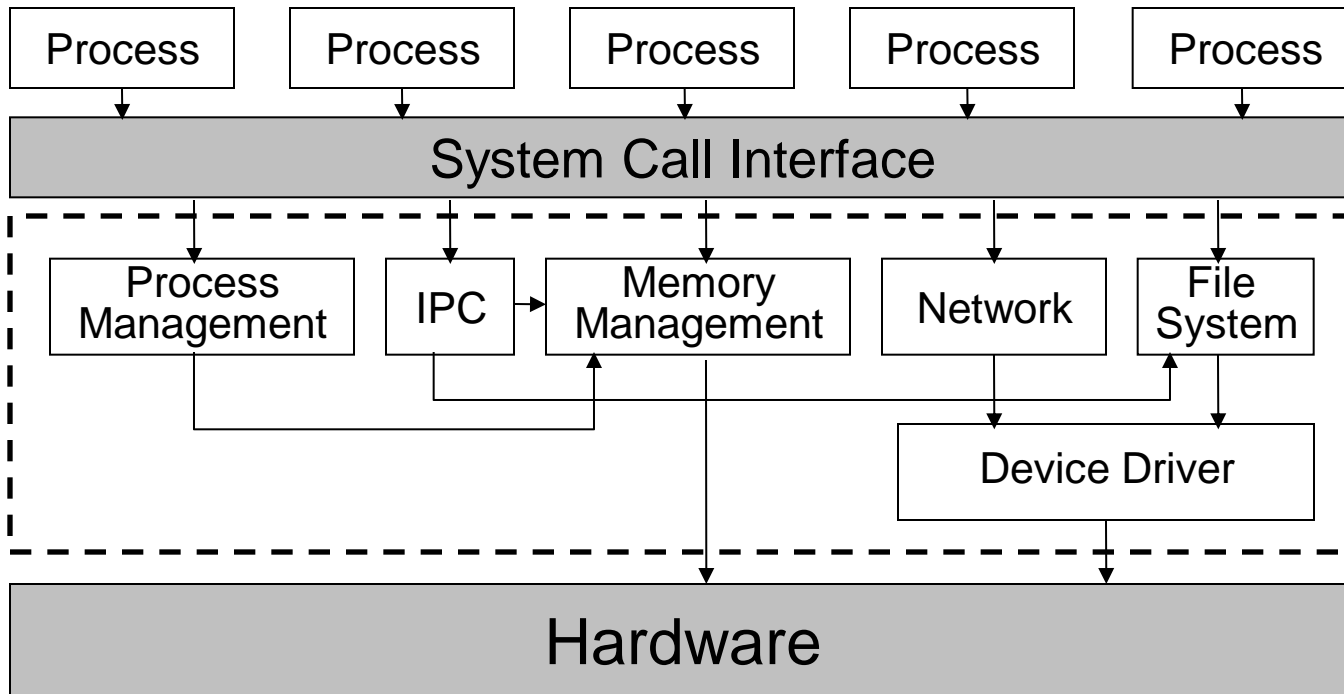


Camadas em Linux





Linux Kernel: Relacionamentos



APPLICATIONS

Home

Contacts

Phone

Browser

...

APPLICATION FRAMEWORK

Activity Manager

Window Manager

Content

View

Notification Manager

Package Manager

Telephony Manager

GTalk Service

LIBRARIES

Surface Manager

Media Framework

OpenGL |

ANDROID

SGL

SSL

ANDROID RUNTIME

Core Libraries

Dalvik Virtual Machine



LINUX KERNEL

Display Driver

Camera Driver

Bluetooth Driver

Flash Memory Driver

Binder (IPC) Driver

USB Driver

Keypad Driver

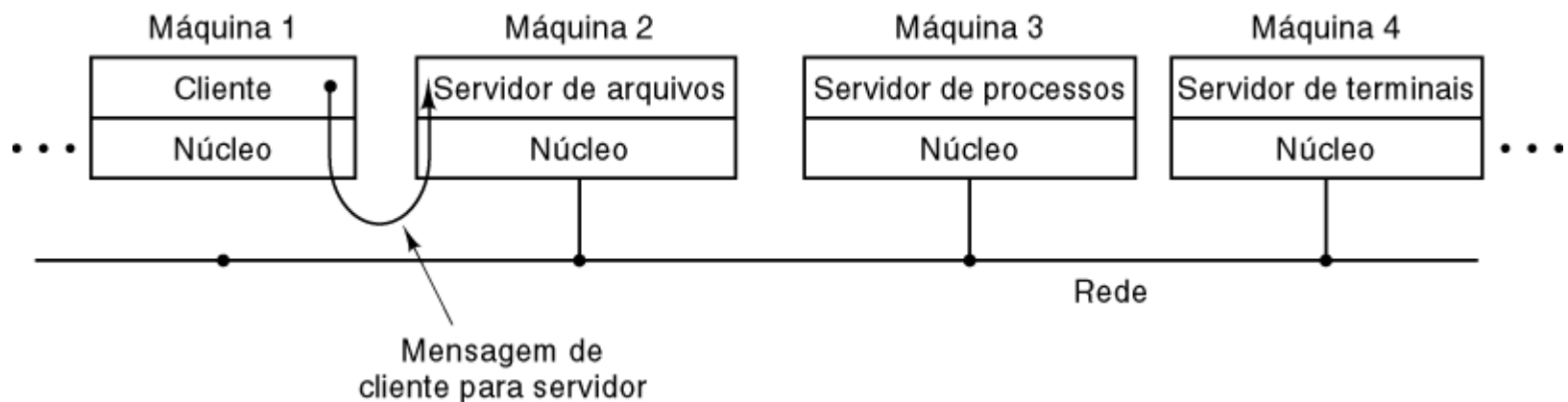
WiFi Driver

Audio Drivers

Power Management

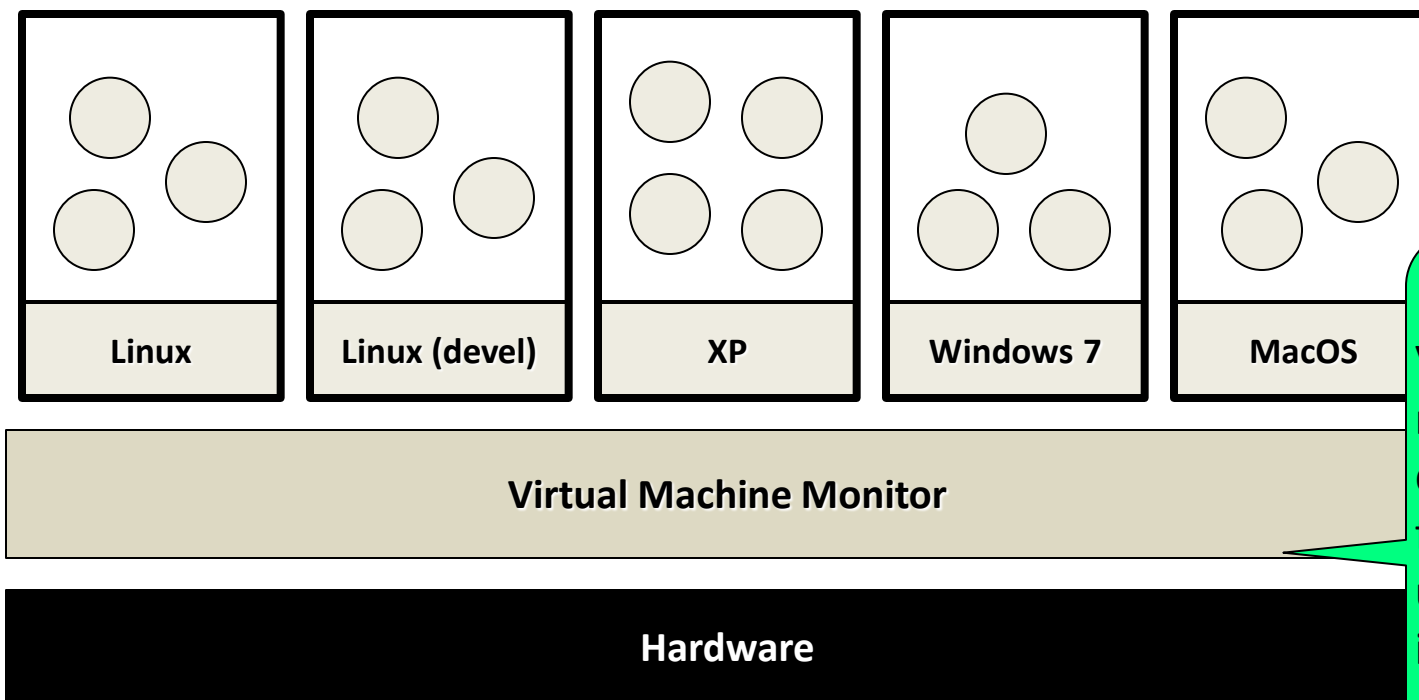
Estrutura de Sistemas Operacionais: Cliente-Servidor

- O modelo **cliente-servidor** em um **sistema (operacional) distribuído**



Estrutura de Sistemas

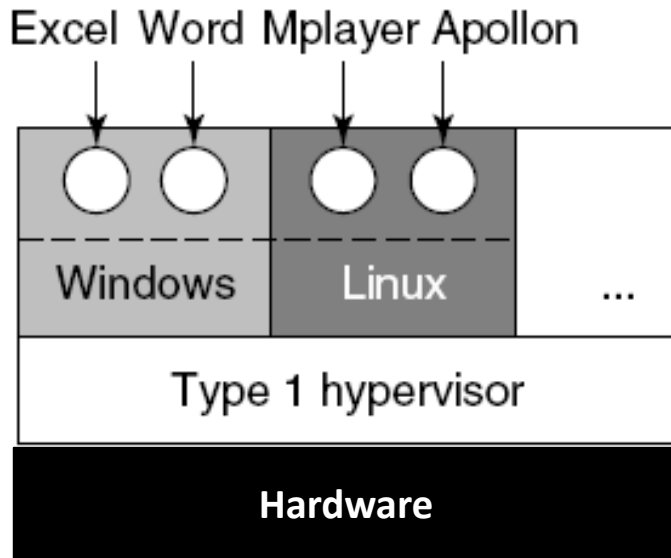
Operacionais: Máquina Virtual (Virtualização)



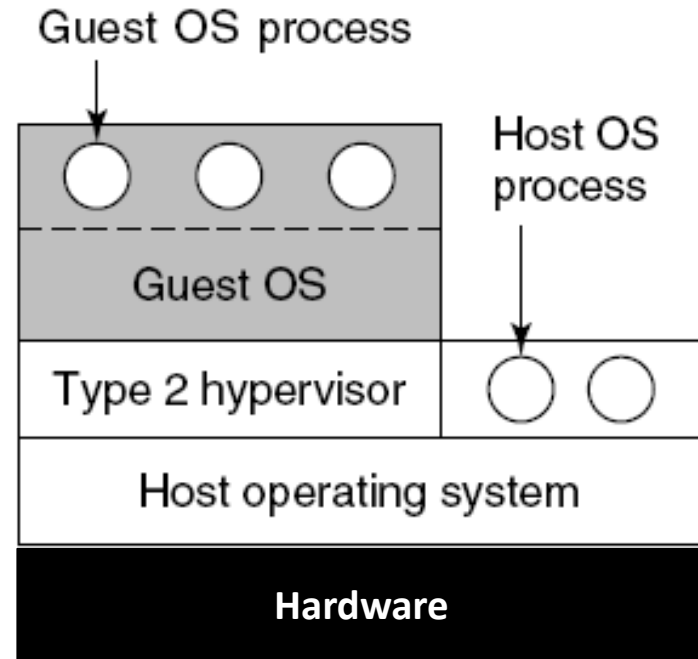
VMM opera na interface de hardware, fornecendo uma interface idêntica para os SOs acima



Virtual Machines: Tipos (Arquiteturas)



Hipervisor Tipo 1



Hipervisor Tipo 2

Ex.: Cloud Computing

