

Sistemas Operacionais

Estrutura de um S.O.

(Parte 1)

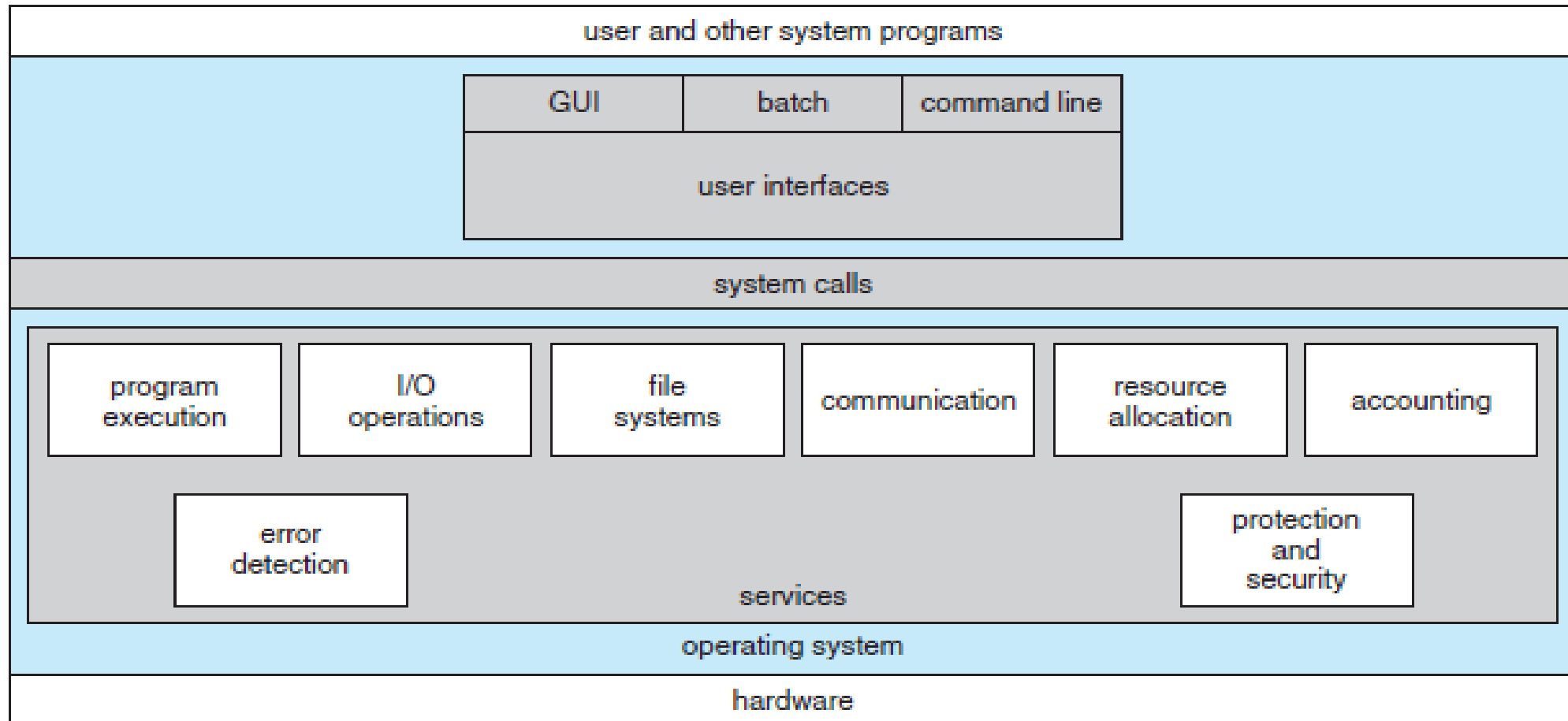
Prof. Otávio Gomes

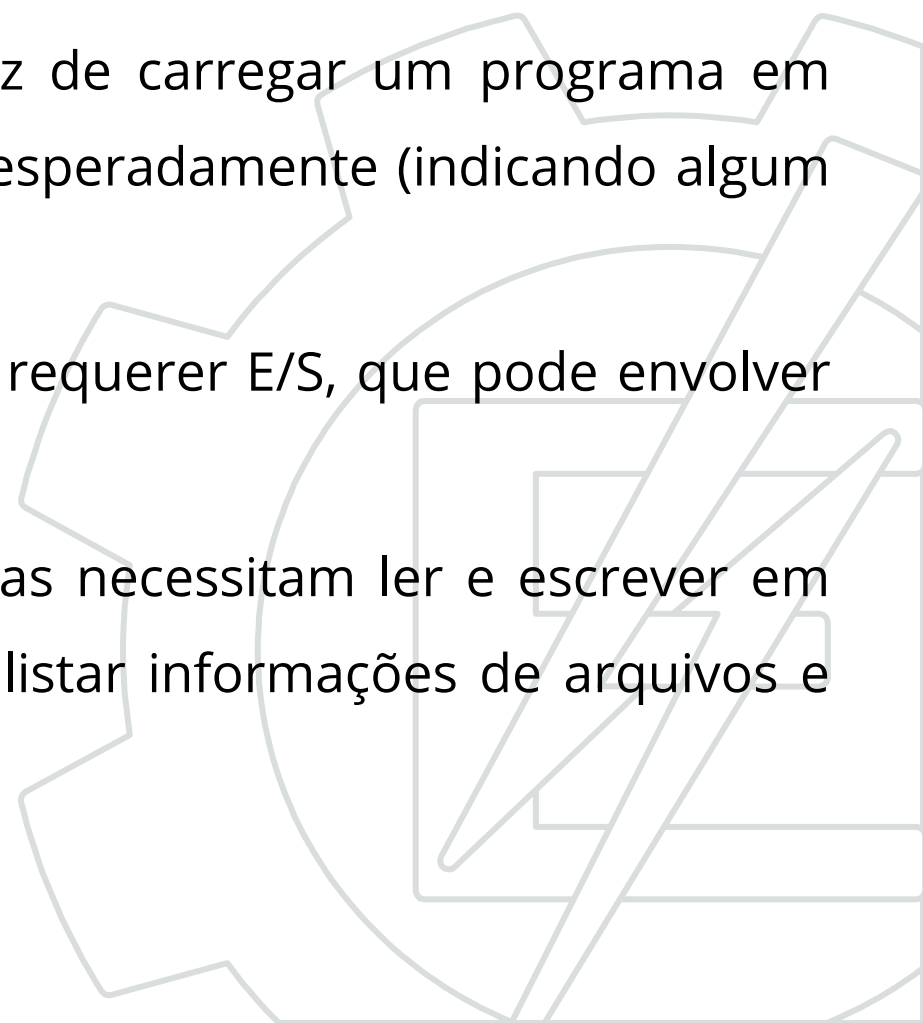
otavio.gomes@unifei.edu.br



Sistema Operacional

Conjunto de serviços



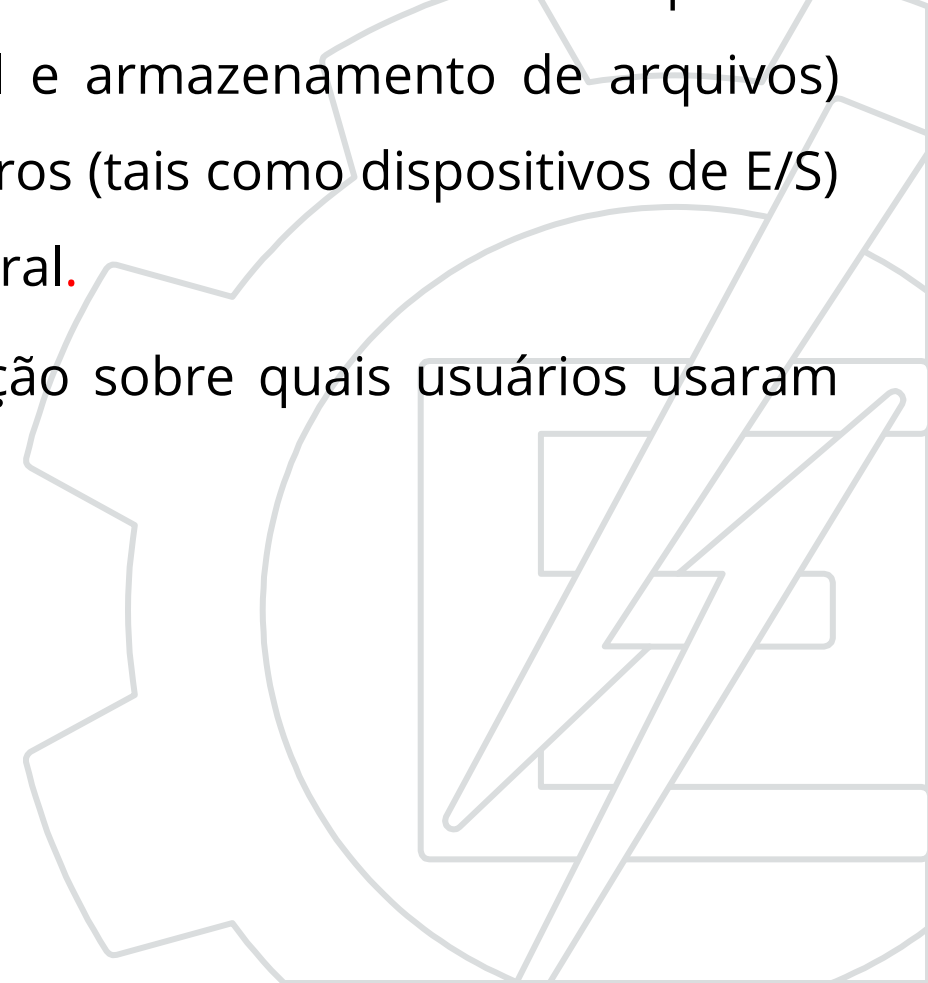
- **Interface com o usuário** – Quase todos SO possuem uma interface com o usuário. Variam entre **linha de Comando**, **Interface Gráfica** e **Lote (*Batch*)**.
 - **Execução de Programas** – O sistema deve ser capaz de carregar um programa em memória e terminar sua execução, normalmente ou inesperadamente (indicando algum erro).
 - **Operações de E/S** - Um programa em execução pode requerer E/S, que pode envolver um arquivo ou dispositivo de E/S.
 - **Manipulação do Sistema de Arquivos** - Os programas necessitam ler e escrever em arquivos e diretórios, criar e removê-los, pesquisá-los, listar informações de arquivos e gerenciar permissões.
- 

- **Comunicação**– Processos podem trocar informações, no mesmo computador ou entre computadores numa rede. A comunicação pode ser entre **memória compartilhada** ou através de **passagem de mensagens** (pacotes movidos pelo SO).
- **Detecção de Erros** – O SO necessita estar constantemente informado de possíveis erros. Podem ocorrer no hardware da CPU ou memória, em dispositivos de E/S, ou um programa do usuário.

Para cada tipo de erro, o SO deve tomar a ação adequada para assegurar a computação correta e consistente

Facilidades de depuração podem melhorar significativamente a capacidade de usuários e programadores usarem o sistema de forma eficiente.

- **Alocação de Recursos** – Quando vários usuários ou vários jobs executam concorrentemente, recursos devem ser alocados para cada um deles. Muitos tipos de recursos – Alguns (Ciclos de CPU, memória principal e armazenamento de arquivos) podem ter código de alocação especial, enquanto outros (tais como dispositivos de E/S) podem ter código de pedido e liberação muito mais geral.
- **Contabilização (*Accounting*)** – Para manter informação sobre quais usuários usaram **quanto** e **quais** tipos de recursos do computador.

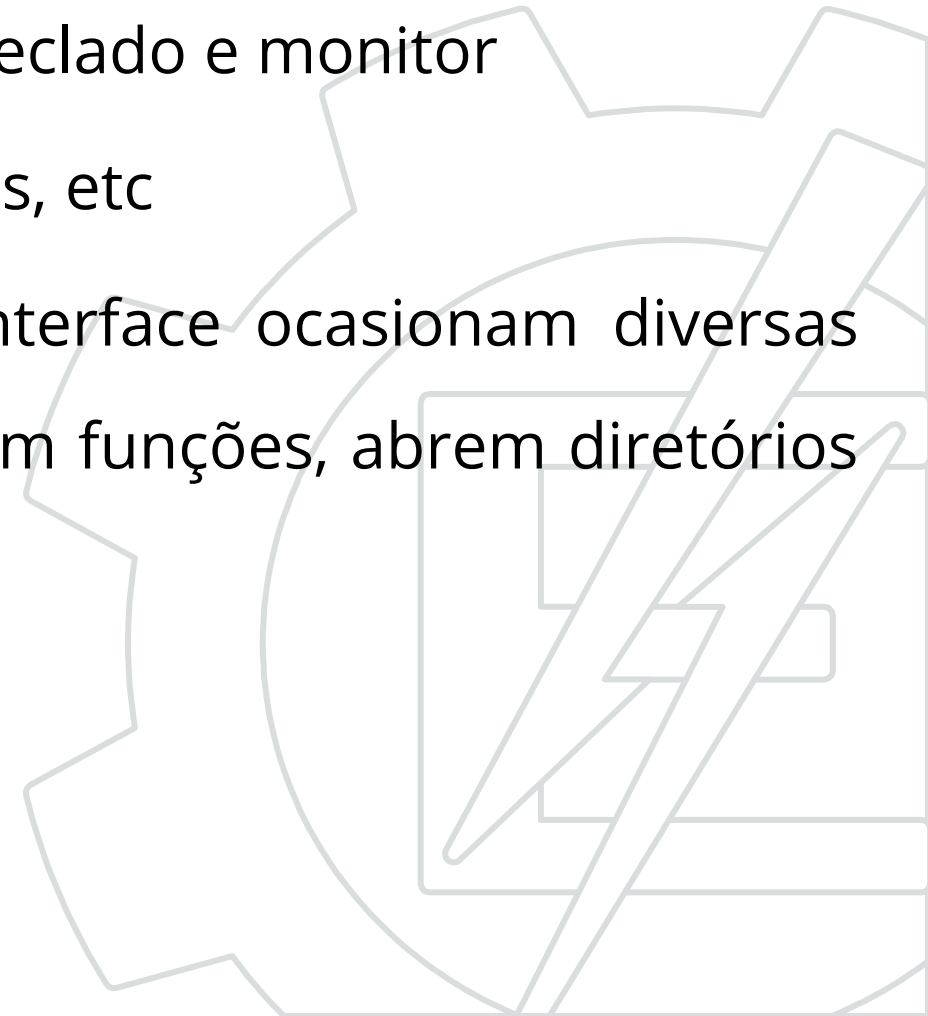


- **Proteção** envolve a garantia de que todo acesso a recursos do sistema é controlado.
- **Segurança** do sistema contra acesso de pessoas estranhas requer autenticação de usuário, estende-se à defesa dos dispositivos de E/S e a manutenção do registro de todas conexões para detecção de invasões.
- Se um sistema deve ser protegido e seguro, precauções devem ser instituídas como um todo.
- *"Uma corrente é tão forte quanto o seu elo mais fraco".*

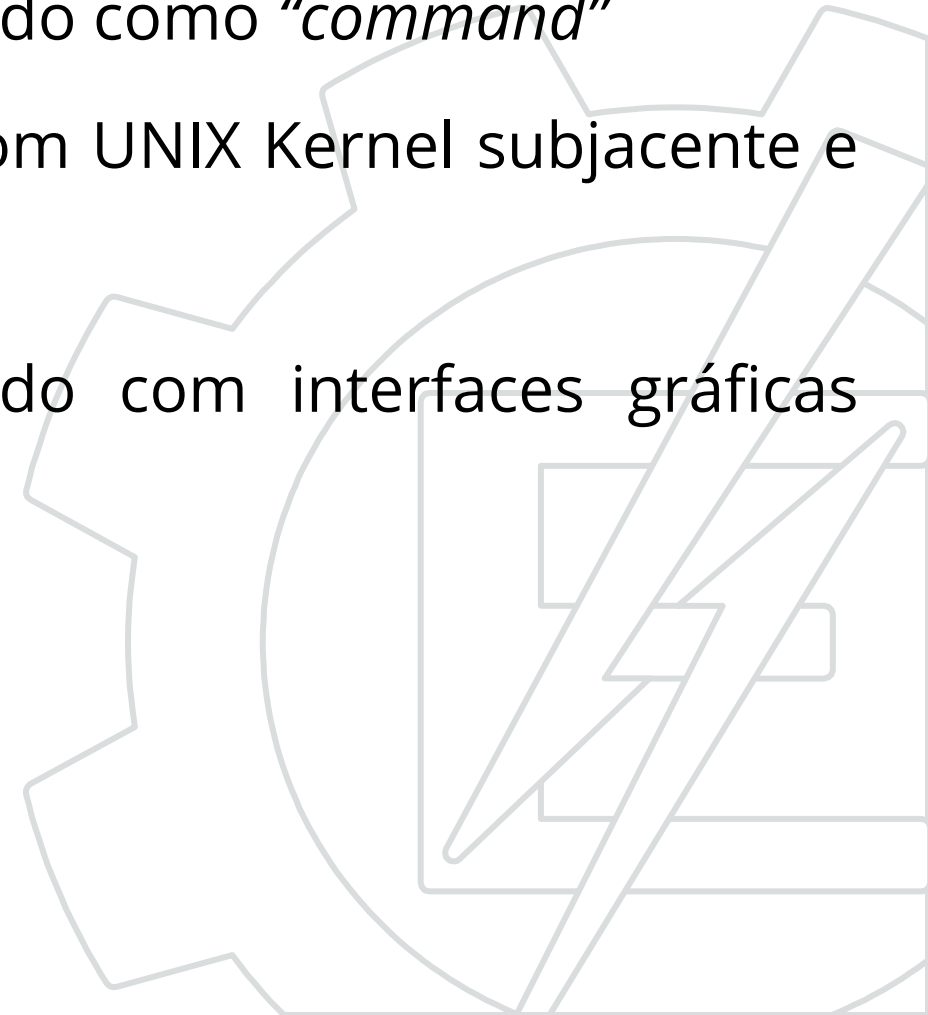
- Uma interface de Linha de Comando ou **interpretador de comandos** permite a entrada direta de comandos
 - Algumas vezes implementadas no kernel, outras vezes por programas do sistema
 - Algumas vezes, implementado em de múltiplas maneiras – **shells**
 - Aceita um comando do usuário e o executa

Se implementado como programa do sistema, a adição de novas funcionalidades não requer modificação do shell

- Interface **desktop** amigável ao usuário (*user-friendly*)
 - Normalmente, permite a utilização de mouse, teclado e monitor
 - **Ícones** representam arquivos, programas, ações, etc
 - Vários botões do mouse sobre objetos na interface ocasionam diversas ações (fornecem informações, opções, executam funções, abrem diretórios (também conhecidos como **pastas**))
 - Inventada no Xerox PARC



- Muitos sistemas atuais incluem tanto interfaces de linha de comando quanto GUI
 - Microsoft Windows possui GUI e o *shell* conhecido como “*command*”
 - Apple Mac OS X possui Interface GUI “Aqua” com UNIX Kernel subjacente e shells disponíveis
 - Solaris possui interface de linha de comando com interfaces gráficas opcionais (Java Desktop, KDE)



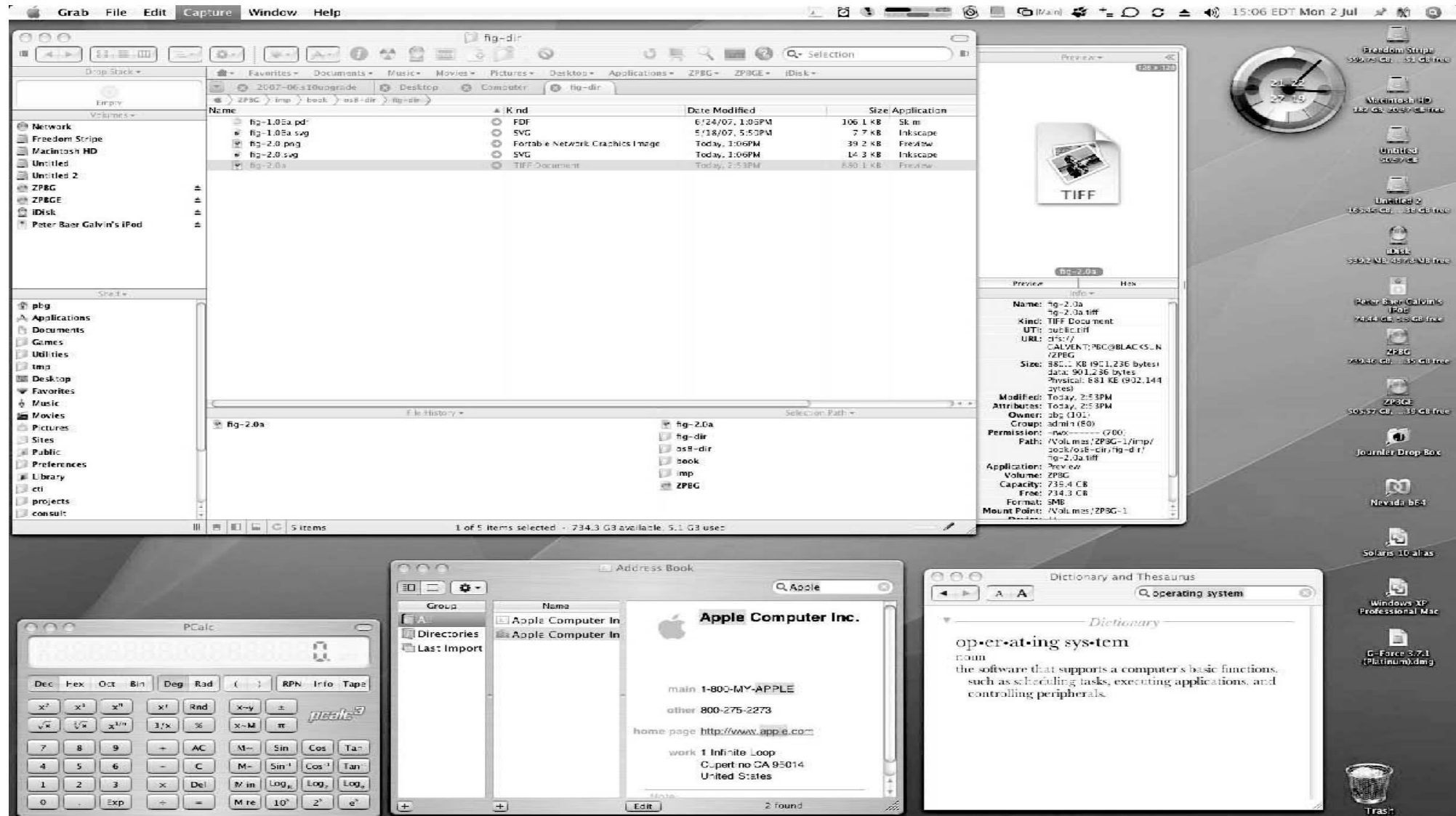
Sistema Operacional

Interpretador de comandos - *bash*

```
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-vn)-(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-vn)-(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-vn)-(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-vn)-(14/pts)-(16:07 02-Jul-2007)-(global)
-(/var/tmp/system-contents/scripts)#
```

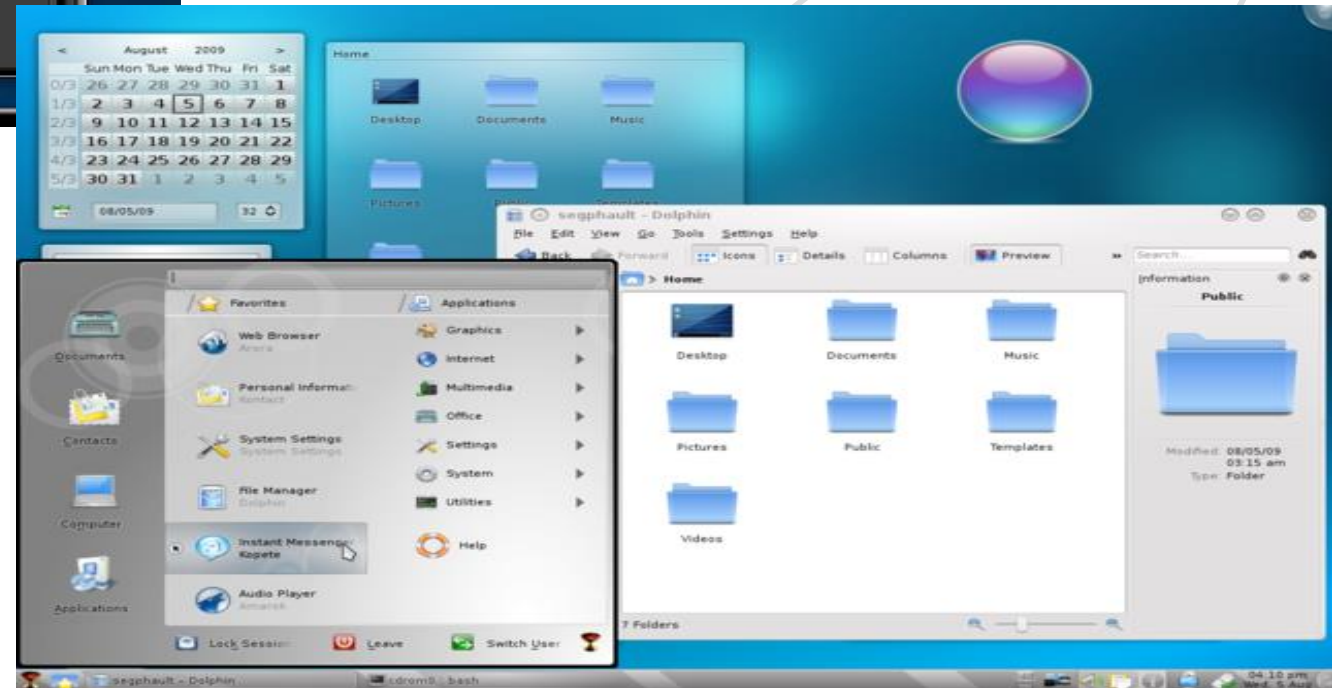
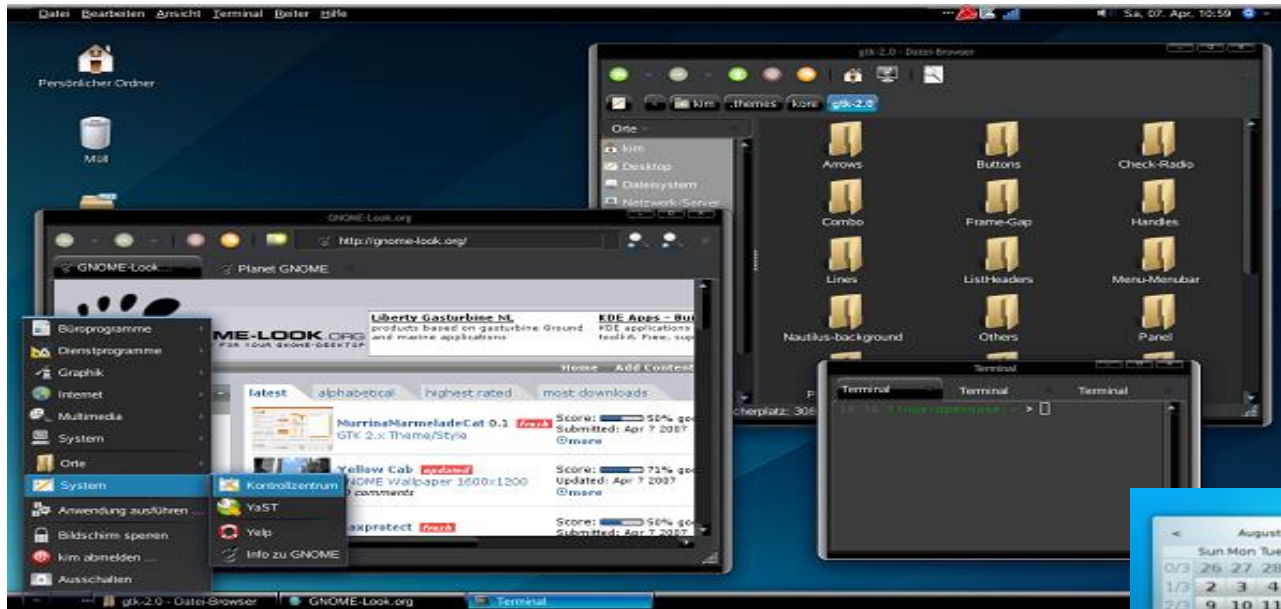
Sistema Operacional

Interface gráfica do MAC OS X



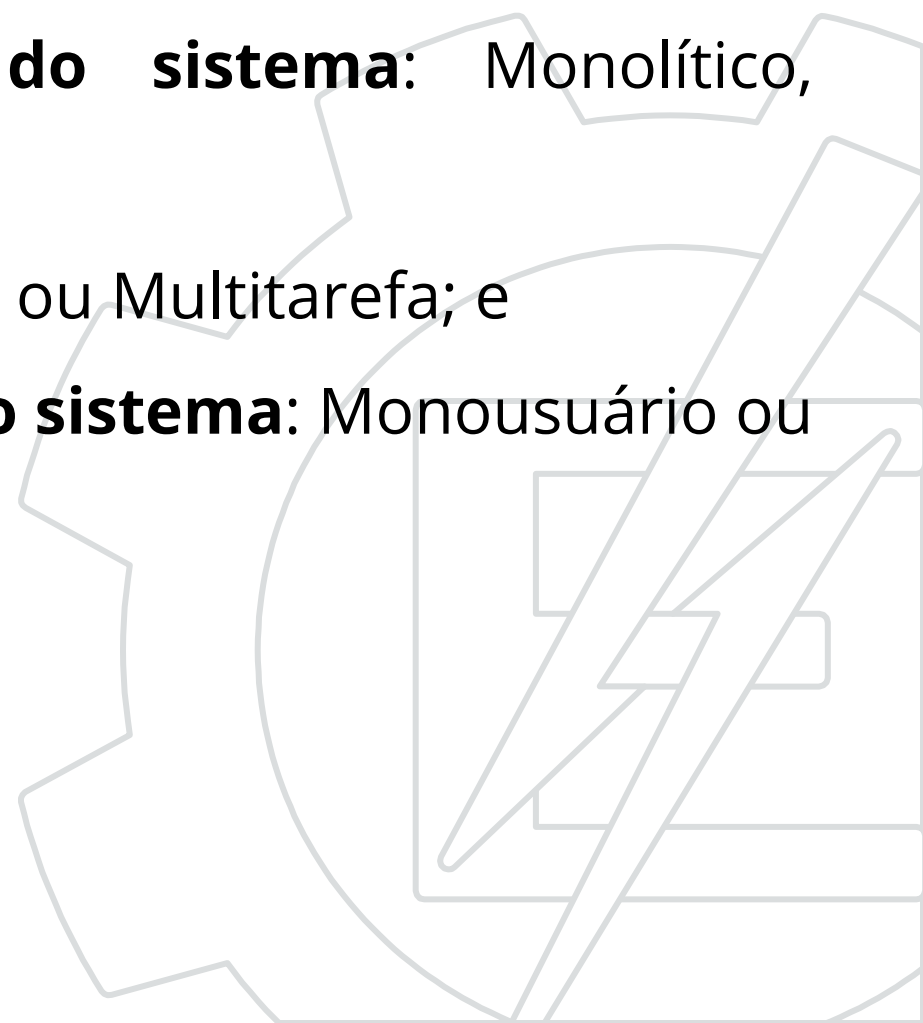
Sistema Operacional

Interfaces gráficas GNOME e KDE



Podem ser classificados de acordo com:

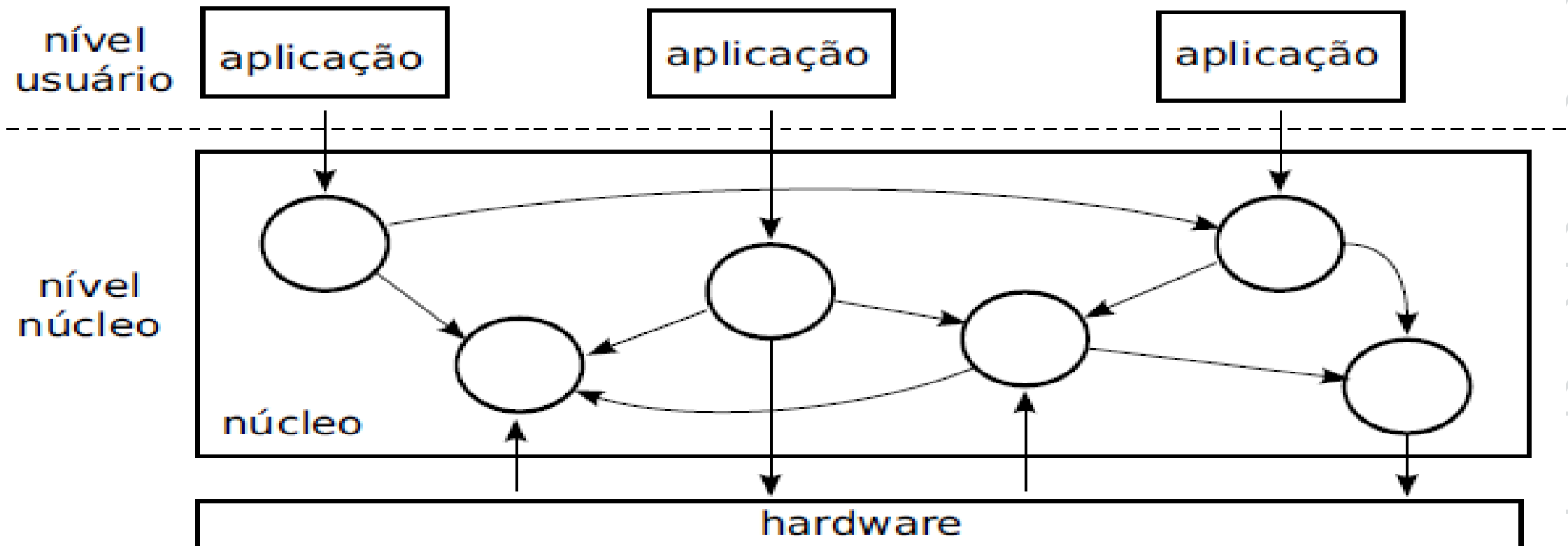
- 1) **Características básicas de arquitetura do sistema:** Monolítico, *Microkernel*, em Camadas ou Máquina virtual;
- 2) **Capacidade de executar tarefas :** Monotarefa ou Multitarefa; e
- 3) **Quantidade de usuários que podem operar o sistema:** Monousuário ou Multiusuário.



Sistema Operacional

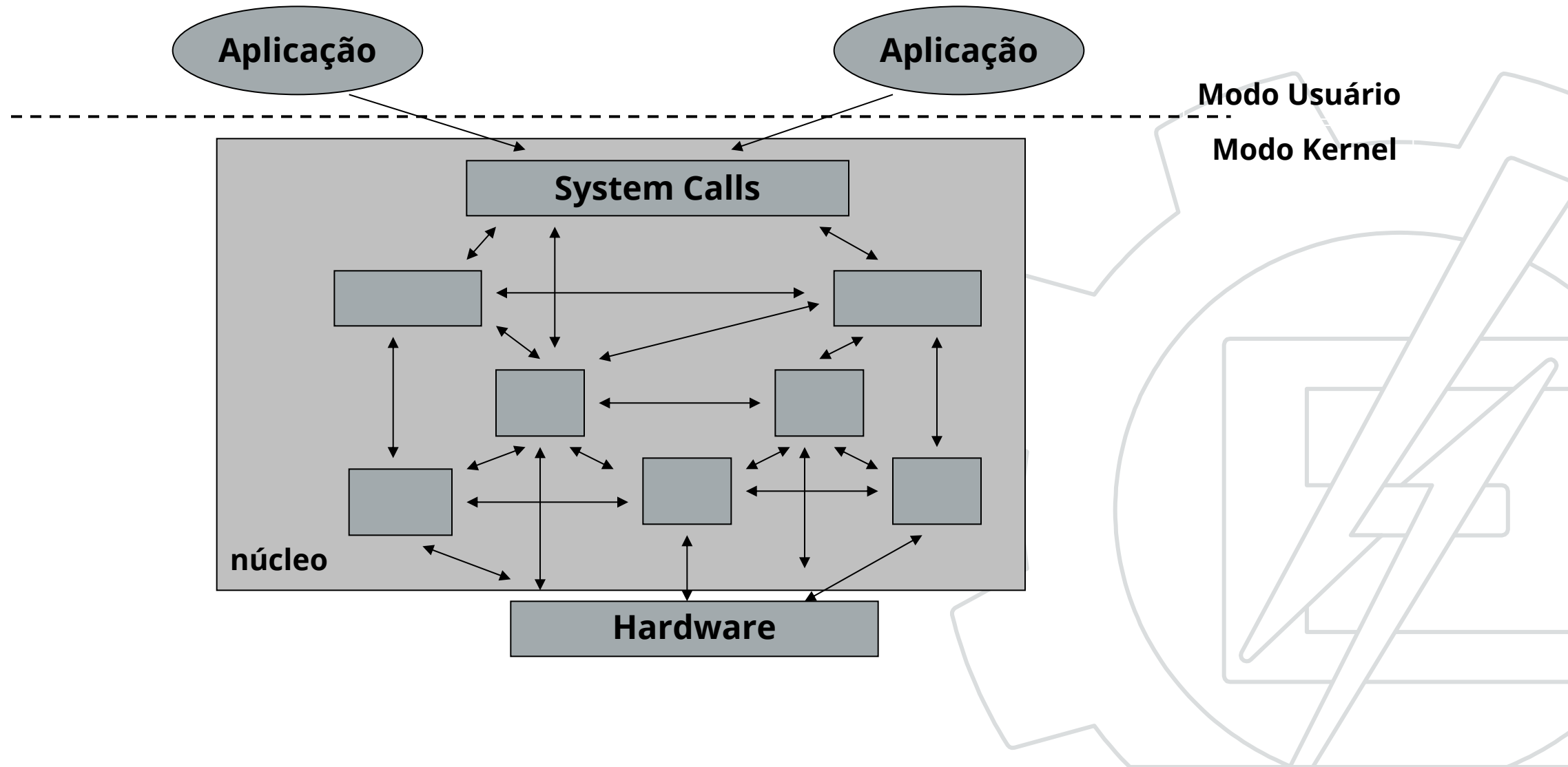
Classificação – Arquitetura: Monolítico

- Todos os componentes do núcleo operam em **modo núcleo** e se inter-relacionam sem restrições de acesso entre si.



Sistema Operacional

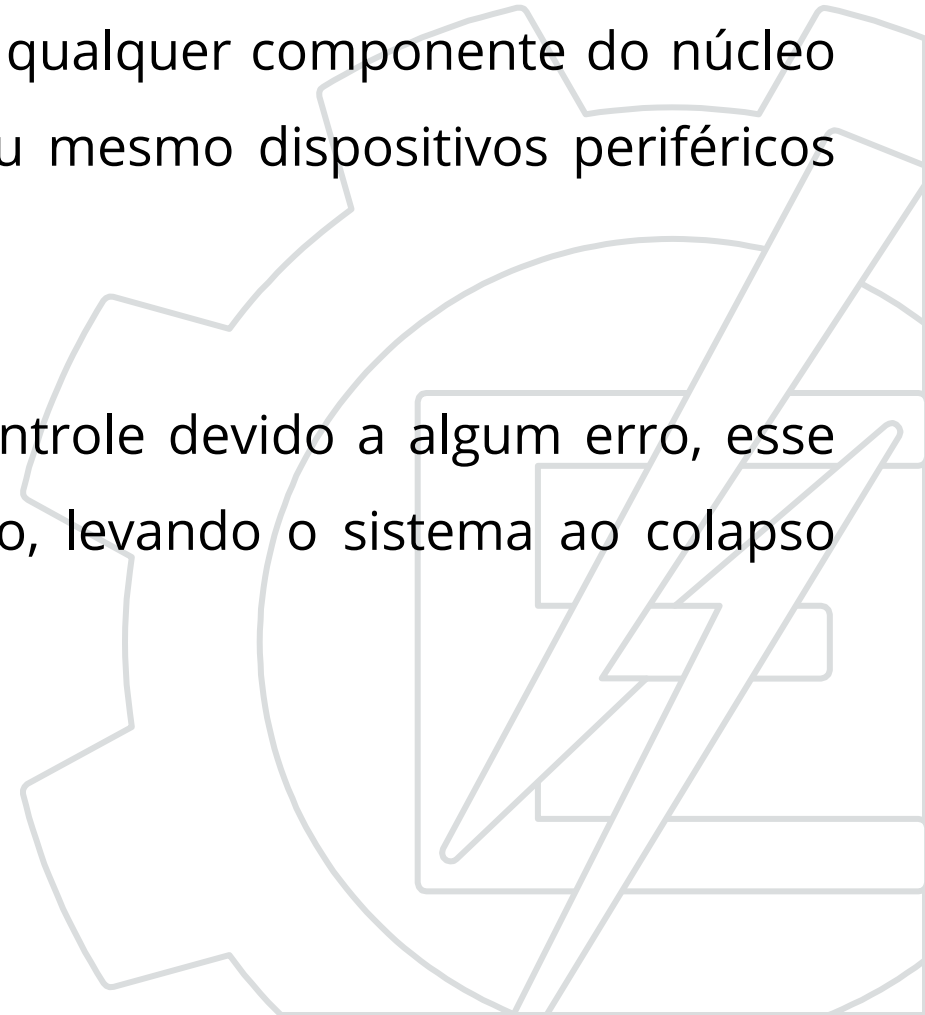
Classificação – Arquitetura: Monolítico



Sistema Operacional

Classificação – Arquitetura: **Monolítico**

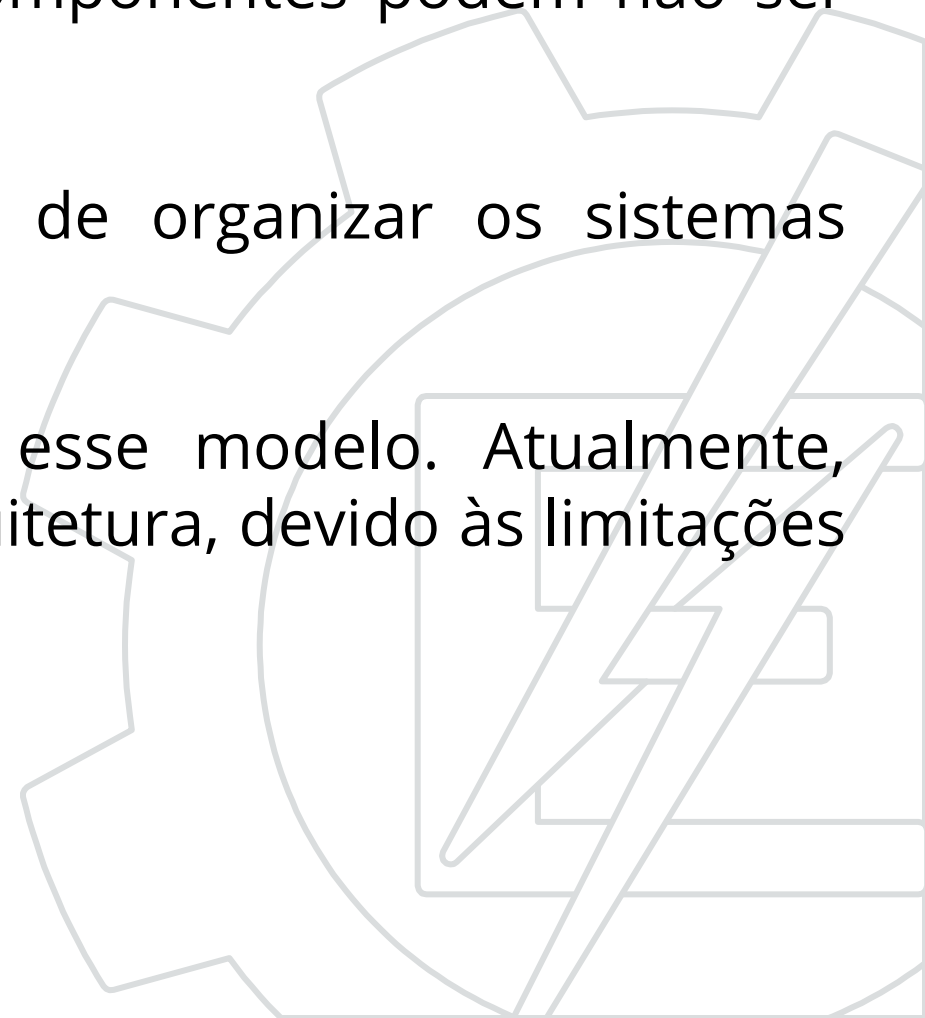
- É construído como uma **coleção de rotinas** onde cada uma pode chamar qualquer outra.
- A grande vantagem dessa arquitetura é seu **desempenho**: qualquer componente do núcleo pode acessar os demais componentes, toda a memória ou mesmo dispositivos periféricos diretamente, pois não há barreiras impedindo esse acesso.
- Desvantagem: caso um componente do núcleo perca o controle devido a algum erro, esse problema pode se alastrar rapidamente por todo o núcleo, levando o sistema ao colapso (travamento, reinicialização ou funcionamento errático).



Sistema Operacional

Classificação – Arquitetura: Monolítico

- A **manutenção** e **evolução** do núcleo se tornam mais complexas, porque as dependências e pontos de interação entre os componentes podem não ser evidentes.
- A arquitetura monolítica foi a primeira forma de organizar os sistemas operacionais.
- Sistemas UNIX antigos e o MS-DOS seguiam esse modelo. Atualmente, sistemas operacionais embutidos usam essa arquitetura, devido às limitações do *hardware* sobre o qual executam.



Sistema Operacional

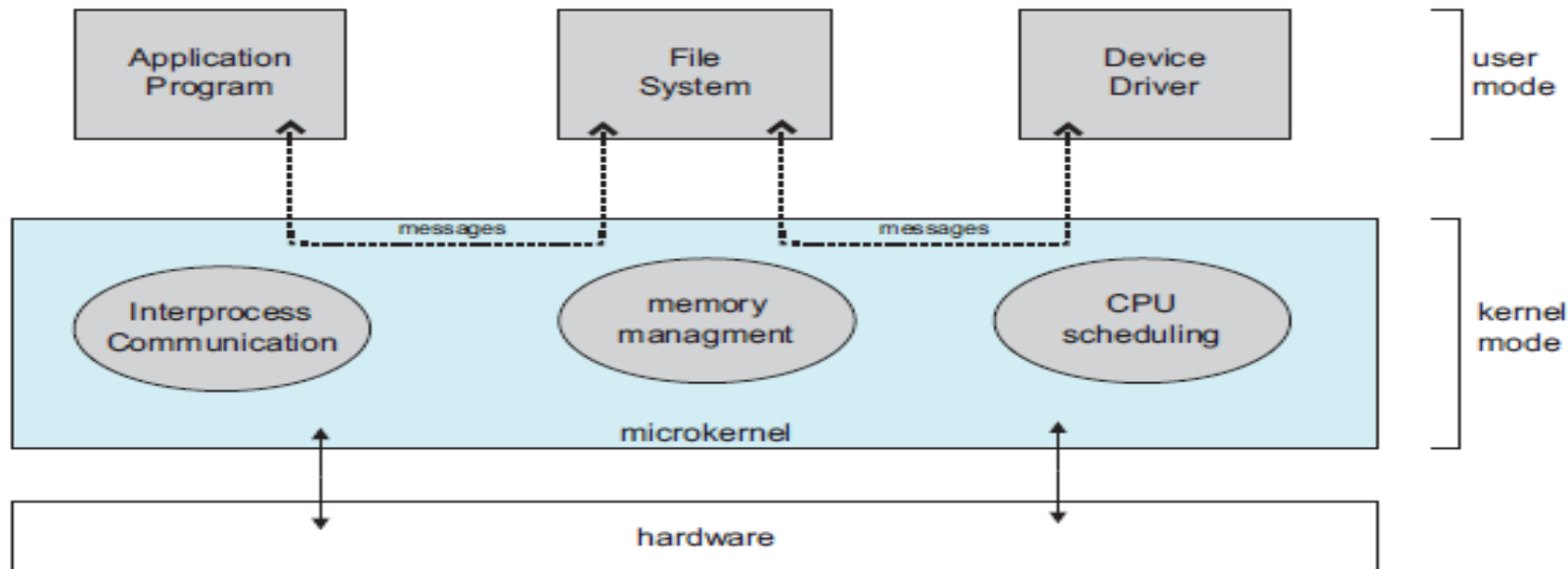
Classificação – Arquitetura: *Microkernel*

- Possui separação de funções chamadas **Servidores** - Consiste em retirar do núcleo todo o código de alto nível (normalmente associado às políticas de gerência de recursos), deixando no núcleo somente o código de baixo nível necessário para interagir com o *hardware* e criar as abstrações fundamentais (como a noção de atividade).
- Por exemplo, usando essa abordagem o código de acesso aos blocos de um disco rígido seria mantido no núcleo, enquanto as abstrações de arquivo e diretório seriam criadas e mantidas por um código fora do núcleo, executando da mesma forma que uma aplicação do usuário.
- Por fazer os núcleos de sistema ficarem menores, essa abordagem foi denominada micro-núcleo (ou *microkernel*).

Sistema Operacional

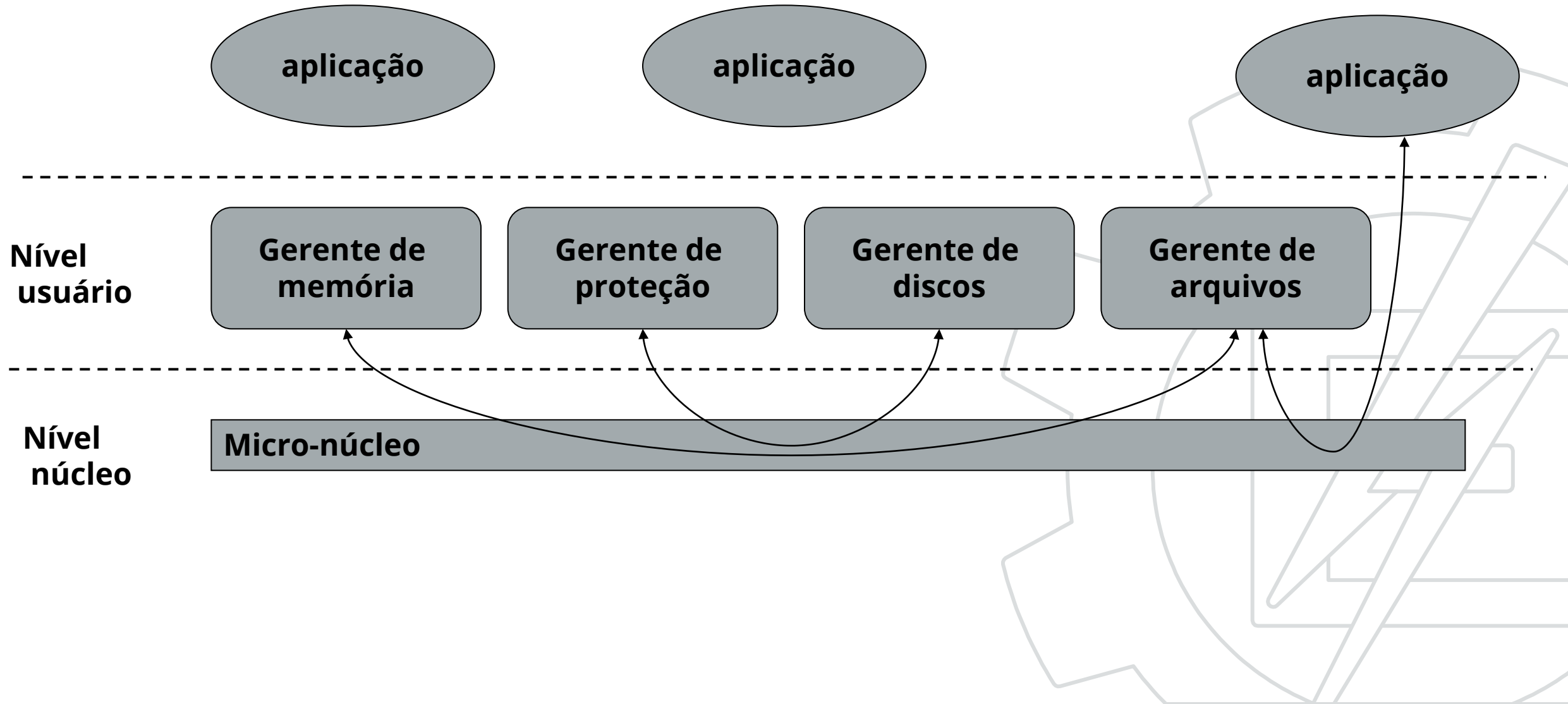
Classificação – Arquitetura: *Microkernel*

- As interações entre componentes e aplicações são feitas através de **trocas de mensagens**.
- Os processos não podem se comunicar diretamente, devido às restrições impostas pelos **mecanismos de proteção do hardware**.
- Todas as mensagens são transmitidas através de serviços do micronúcleo. Como os processos têm de solicitar “serviços” uns dos outros, para poder realizar suas tarefas, essa abordagem também foi denominada **cliente-servidor**.



Sistema Operacional

Classificação – Arquitetura: *Microkernel*



Sistema Operacional

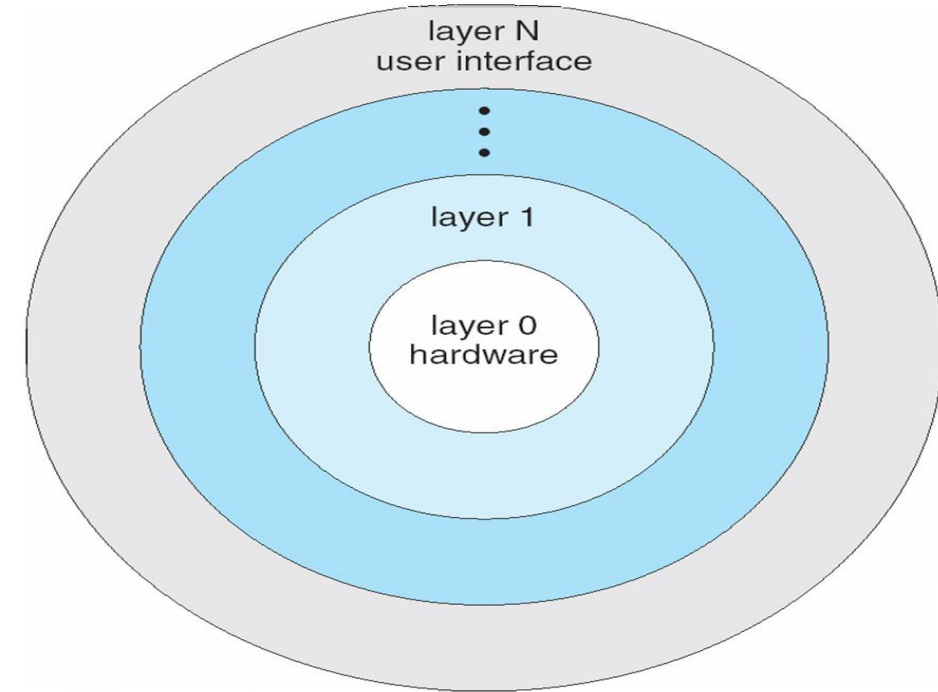
Classificação – Arquitetura: *Microkernel*

- O micronúcleos foram muito investigados durante os anos 80. Dois exemplos clássicos dessa abordagem são os sistemas **Mach** [Rashid et al., 1989] e **Chorus** [Rozier et al., 1992].
- As principais vantagens dos sistemas micro-núcleo são sua **robustez e flexibilidade**: caso um sub-sistema tenha problemas, os mecanismos de proteção de memória e níveis de privilégio irão confiná-lo, impedindo que a instabilidade se alastre ao restante do sistema.
- ▪ Vários sistemas operacionais atuais adotam parcialmente essa estruturação; por exemplo, o **MacOS X** da Apple tem suas raízes no sistema Mach, ocorrendo o mesmo com o **Digital UNIX**. Assim como o Minix e o Symbian.
- O **QNX** é um dos poucos exemplos de micronúcleo amplamente utilizado, sobretudo em sistemas embarcados e de tempo-real.

Sistema Operacional

Classificação – Arquitetura: em Camadas

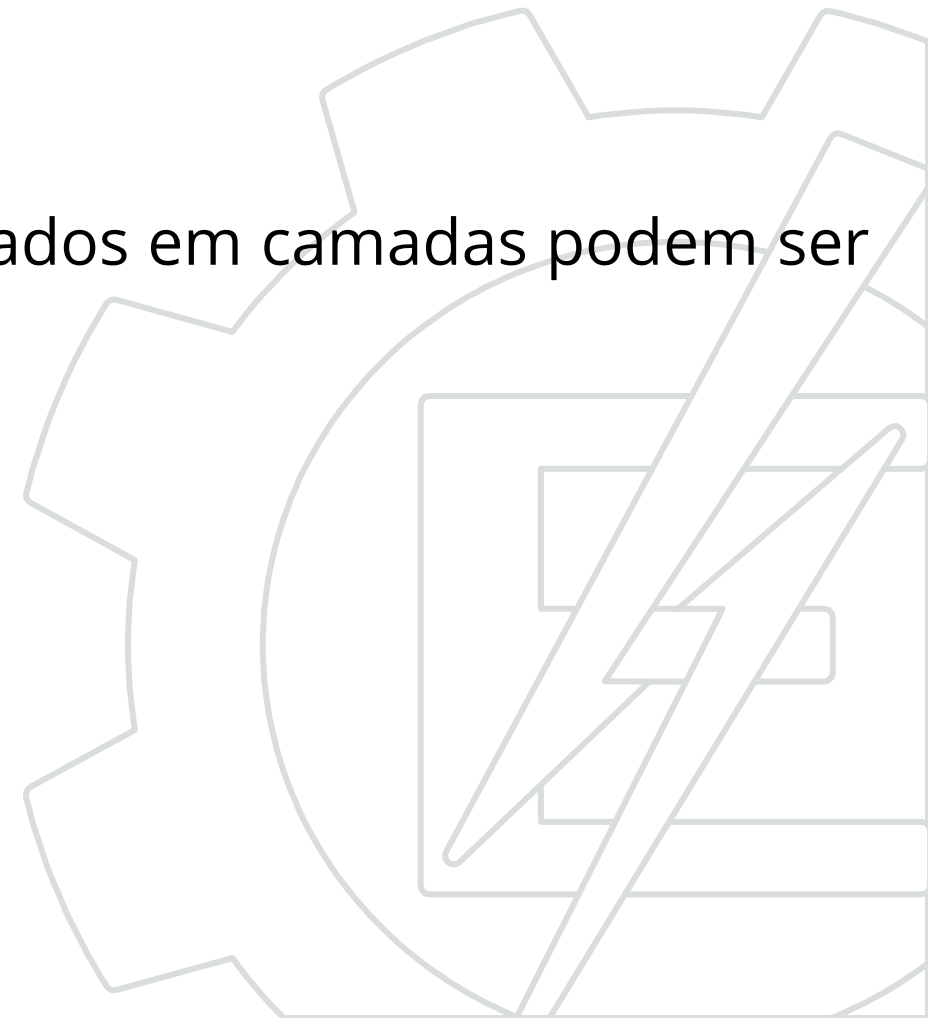
- O sistema operacional é dividido em um número de camadas (níveis), cada uma construída no topo das outras camadas. A camada inferior (camada 0), é o *hardware*; a camada mais alta (camada N) é a interface com o usuário.
- Com modularização, as camadas são selecionadas de tal forma que cada uma use as funções (operações) e serviços das camadas de nível inferior.



- Essa abordagem de estruturação de software fez muito sucesso no domínio das redes de computadores, através do modelo de referência **OSI** (***Open Systems Interconnection***), e também seria de se esperar sua adoção no domínio dos sistemas operacionais.
- Muitos sistemas implementam uma camada inferior de abstração do *hardware* para interagir com os dispositivos (a camada **HAL** – *Hardware Abstraction Layer*, implementada no Windows NT e seus sucessores), e também organizam em camadas alguns subsistemas como a gerência de arquivos e o suporte de rede (seguindo o modelo OSI).

- No entanto, alguns inconvenientes limitam sua aceitação nesse contexto:
 - O empilhamento de várias camadas de software faz com que cada pedido de uma aplicação demore mais tempo para chegar até o dispositivo periférico ou recurso a ser acessado, prejudicando o desempenho do sistema.
 - Não é óbvio como dividir as funcionalidades de um núcleo de sistema operacional em camadas horizontais de abstração crescente, pois essas funcionalidades são interdependentes, embora tratem muitas vezes de recursos distintos.

- Em decorrência desses inconvenientes, a estruturação em camadas é apenas parcialmente adotada hoje em dia.
- Como exemplos de sistemas fortemente estruturados em camadas podem ser citados o OpenVMS, IBM OS/2 e o MULTICS.

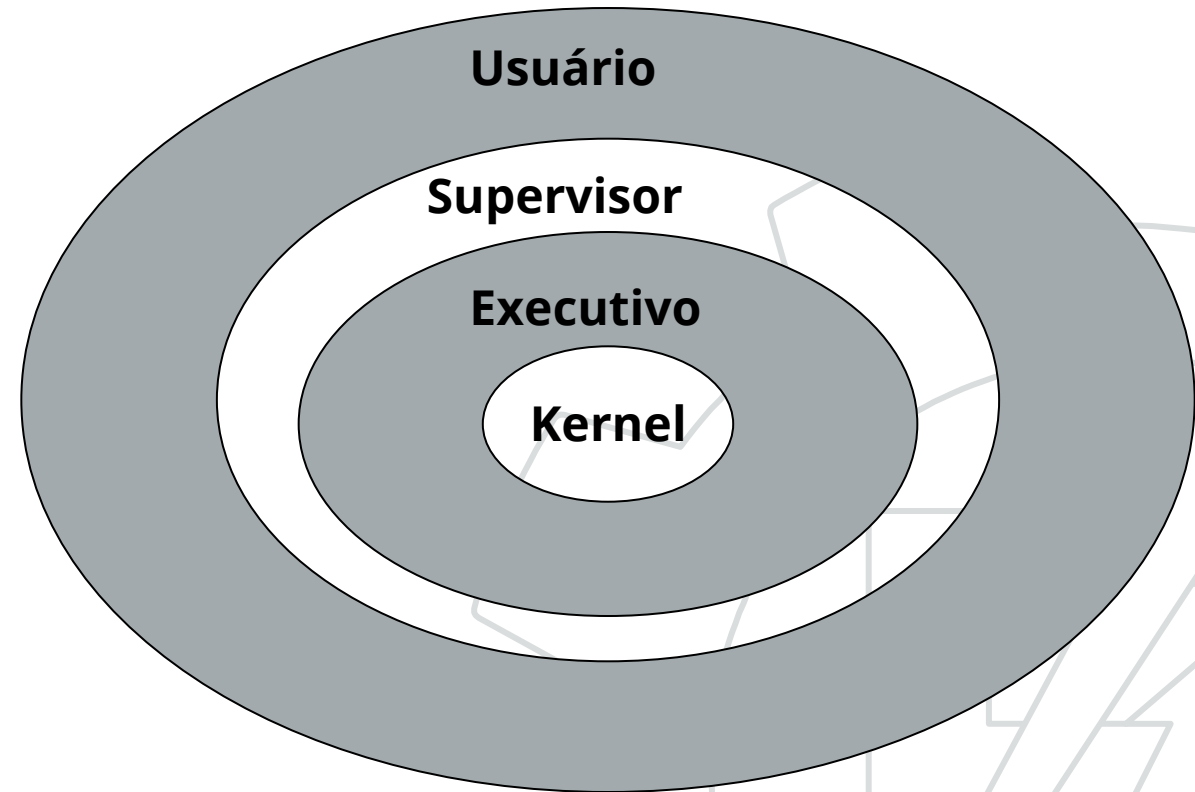


Sistema Operacional

Classificação – Arquitetura: em Camadas

Sistema Multics

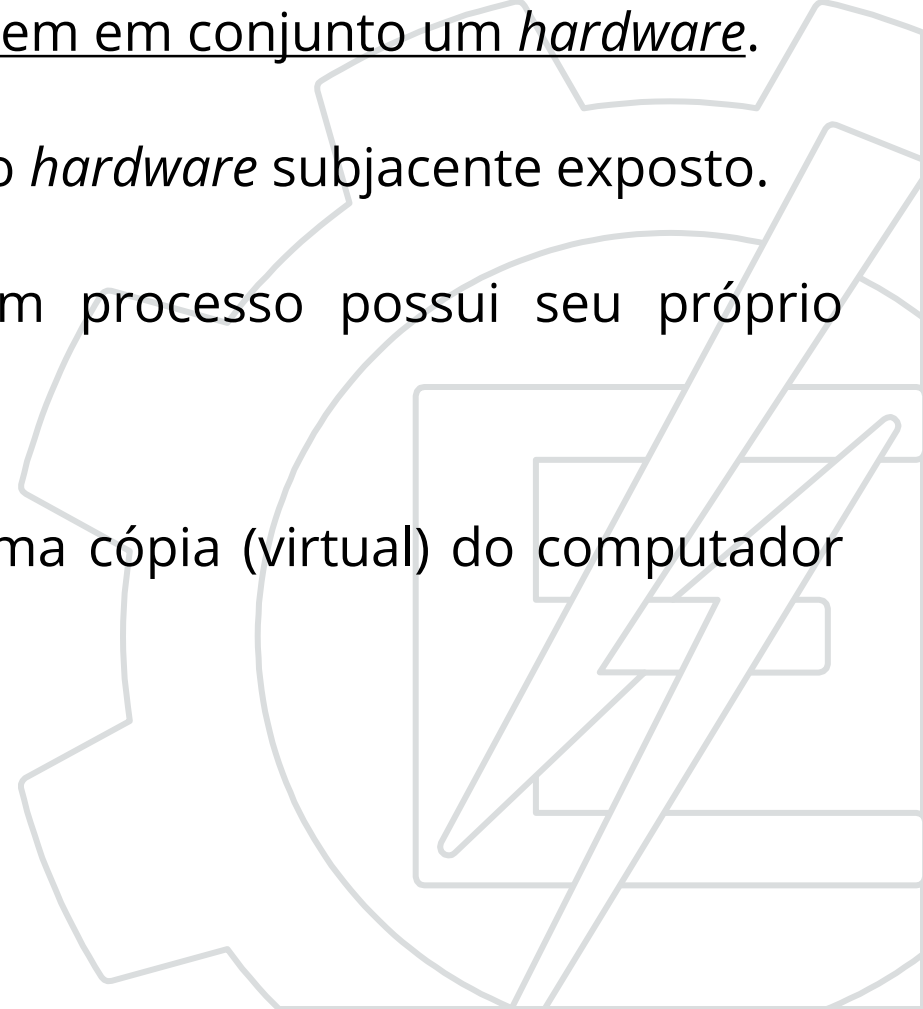
5	Operador
4	Programas de Usuário
3	Entrada/Saída
2	Comunicação
1	Gerência de Memória
0	Multiprogramação



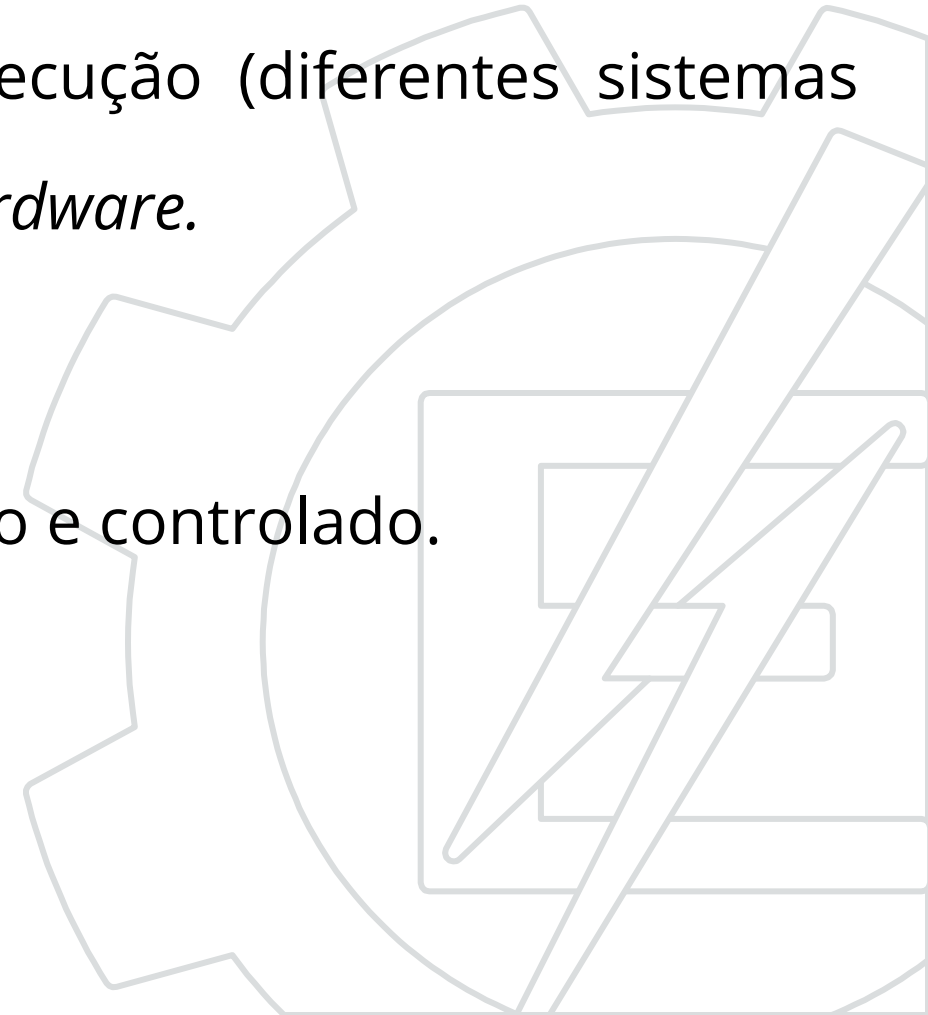
Sistema VMS

Sistema Operacional

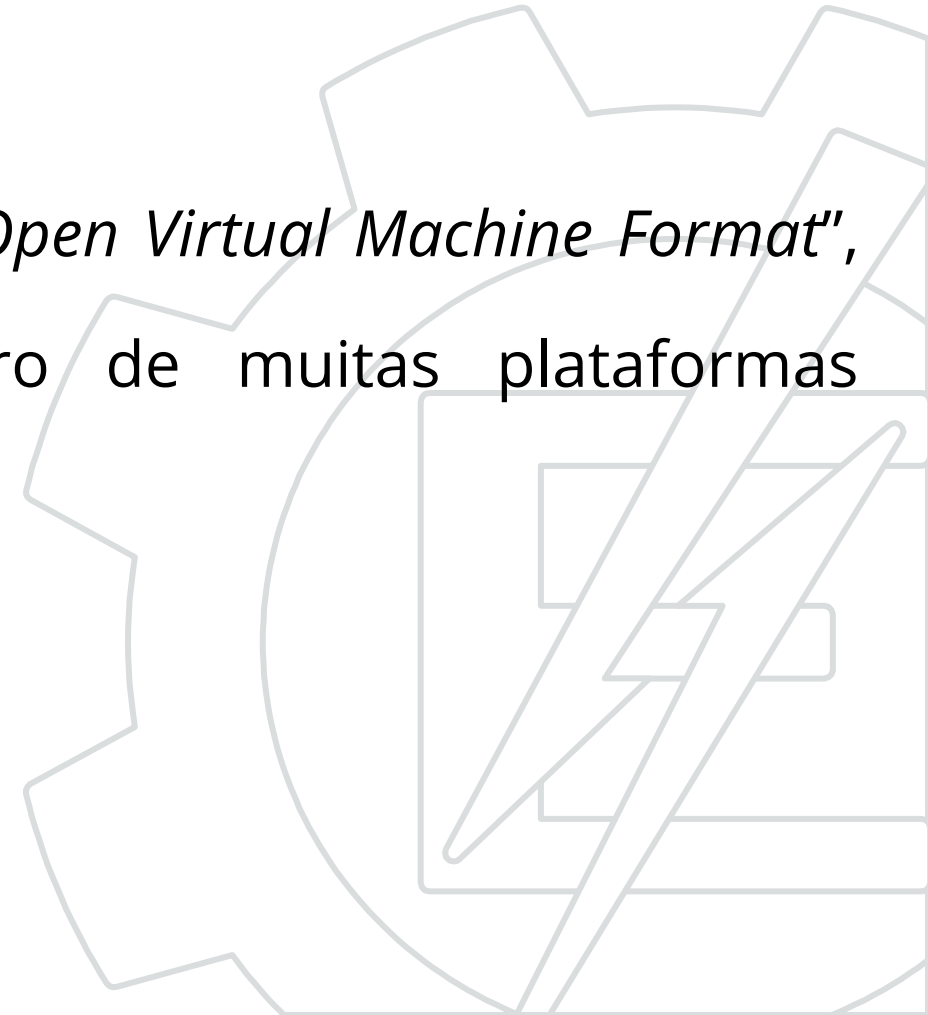
Classificação – Arquitetura: Máquina virtual

- Uma máquina virtual leva a abordagem em camadas para sua conclusão lógica. Ela trata o *hardware* e o *kernel* do sistema operacional como se eles fossem em conjunto um *hardware*.
 - Uma máquina virtual pode fornecer uma interface idêntica ao *hardware* subjacente exposto.
 - O Sistema operacional hospedeiro cria a ilusão que um processo possui seu próprio processador e sua própria memória (virtual).
 - Cada sistema operacional convidado disponibilizado com uma cópia (virtual) do computador subjacente (ou partes dele).
- 

- Apareceram comercialmente em *mainframes* IBM em 1972.
- Fundamentalmente, múltiplos ambientes de execução (diferentes sistemas operacionais) podendo compartilhar o mesmo *hardware*.
- Proteção mútua entre os sistemas operacionais.
- Compartilhamento de arquivos pode ser permitido e controlado.

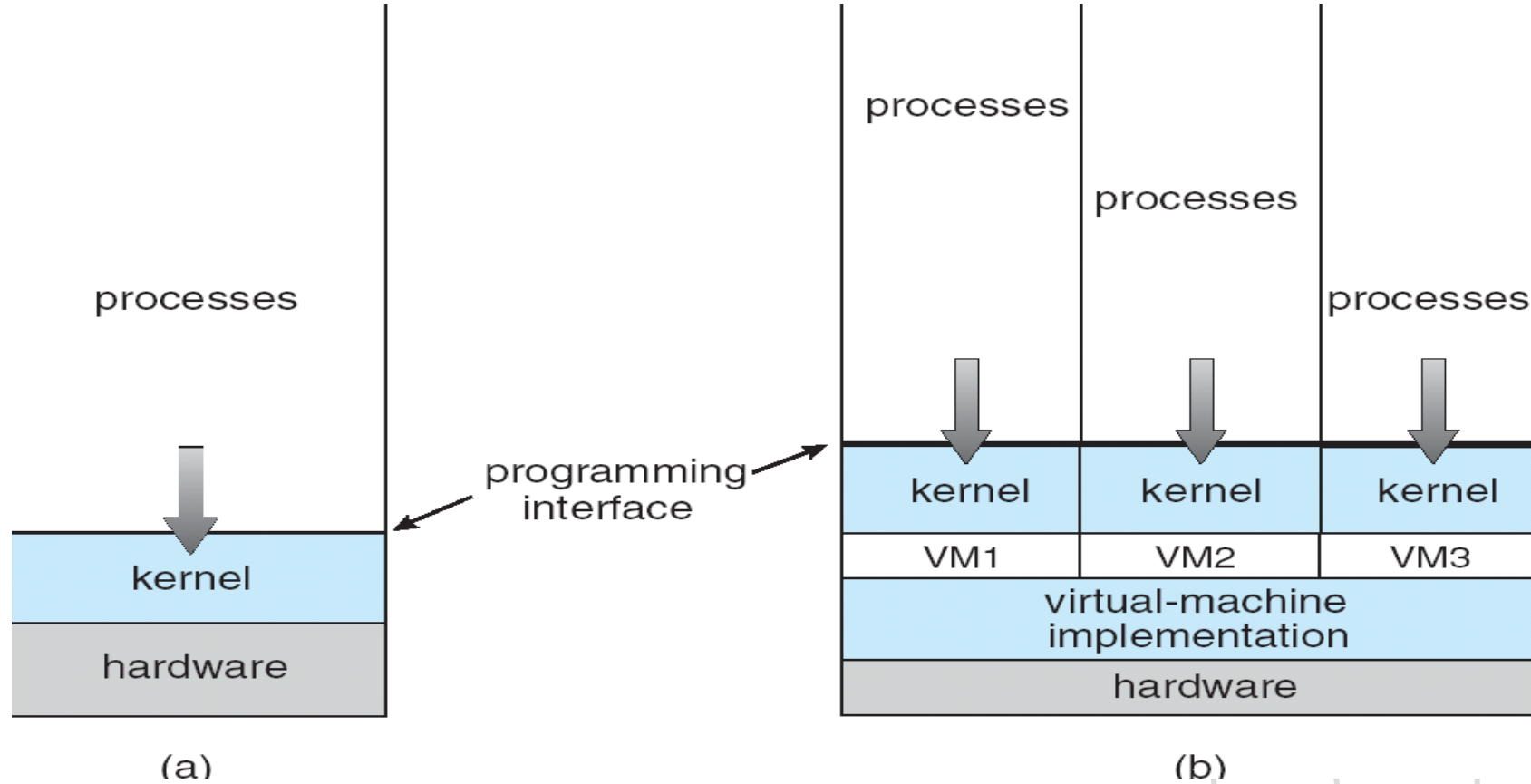


- Comunicam-se umas com as outras, inclusive sistemas físicos via rede.
- Útil para desenvolvimento e testes.
- O padrão “Formato de Máquina Virtual Aberta - *Open Virtual Machine Format*”, permite uma máquina virtual executar dentro de muitas plataformas diferentes de máquinas virtuais (*host*).



Sistema Operacional

Classificação – Arquitetura: Máquina virtual

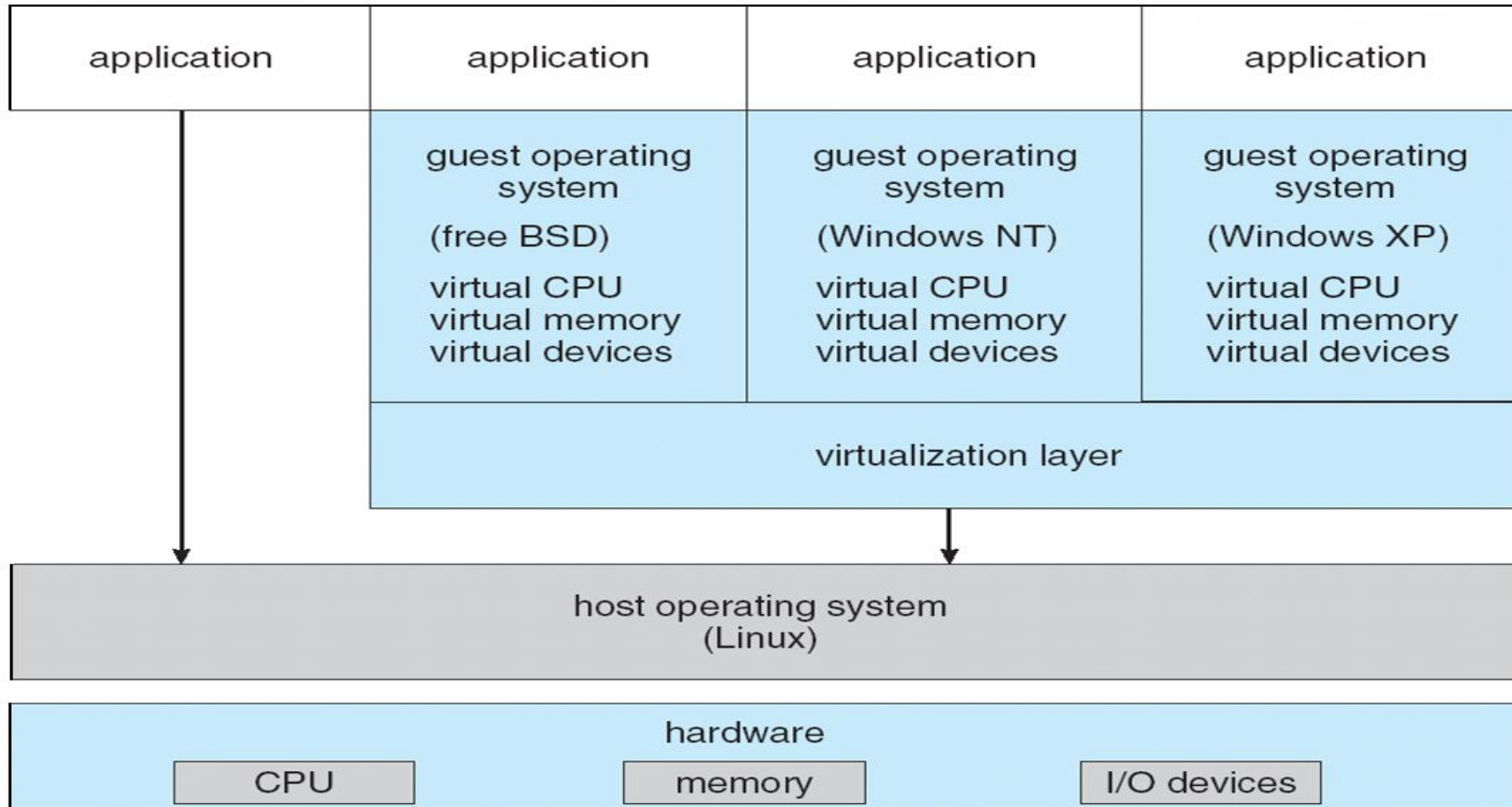


- (a) Máquina não-virtual

- (b) máquina virtual

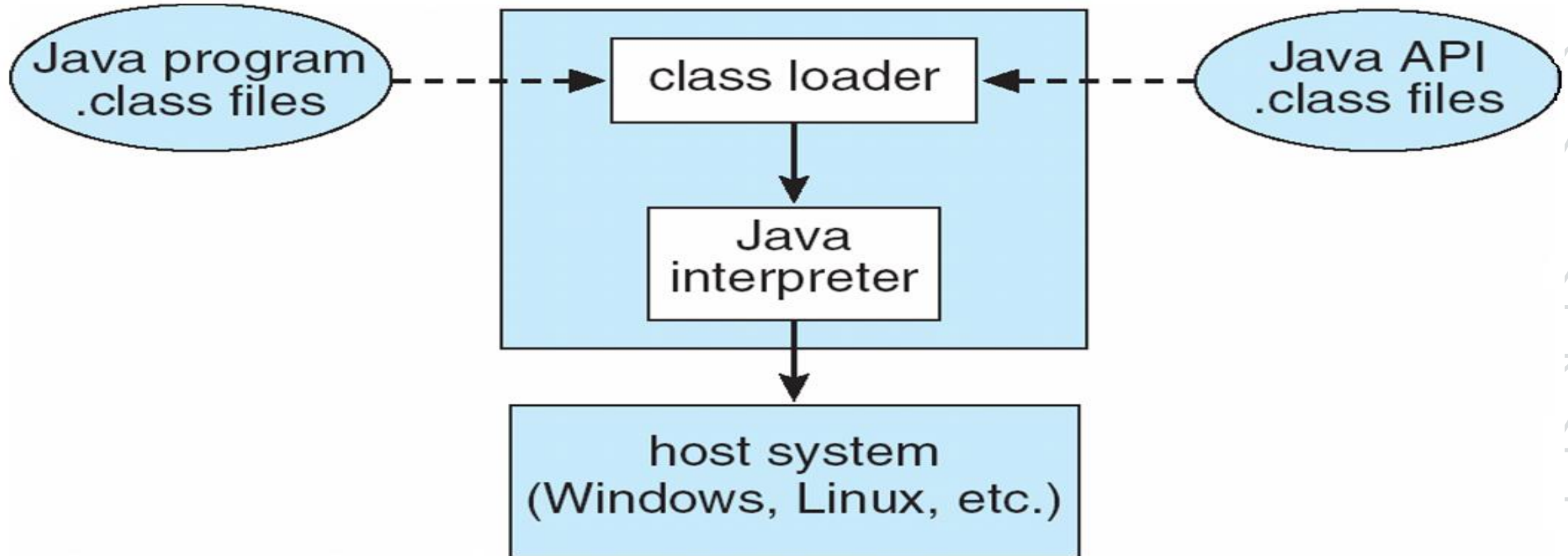
Máquinas Virtuais

Arquitetura VMWare



Máquinas Virtuais

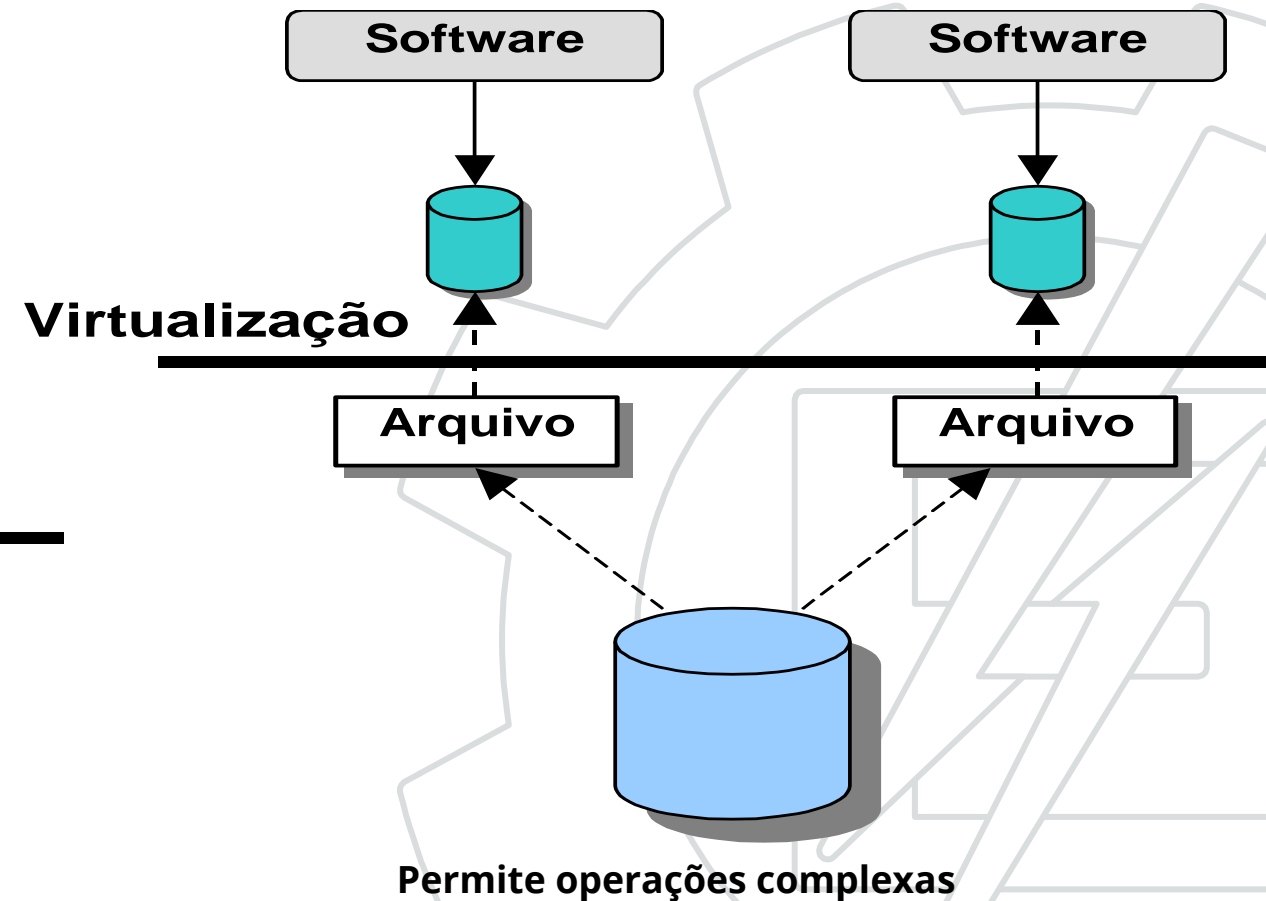
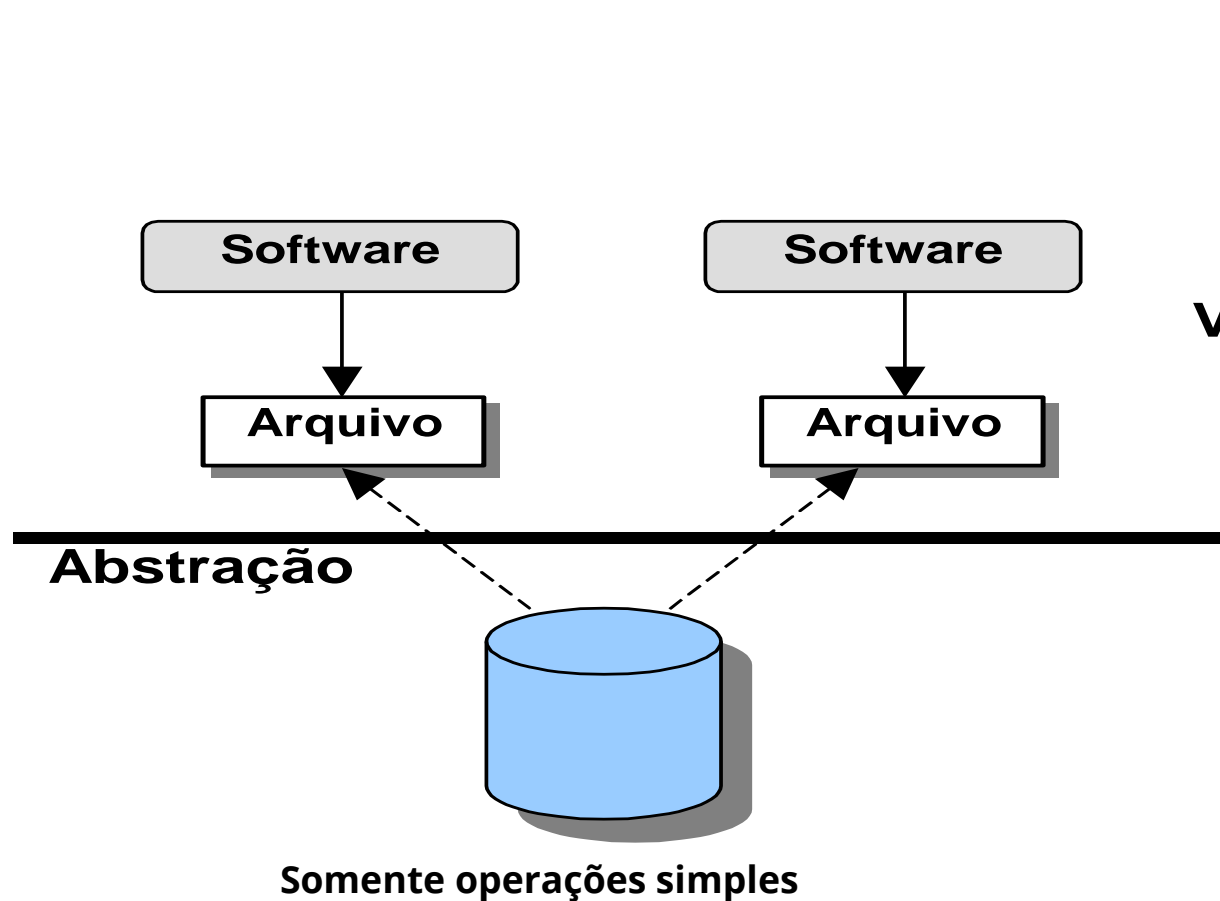
JVM



Sistema Operacional

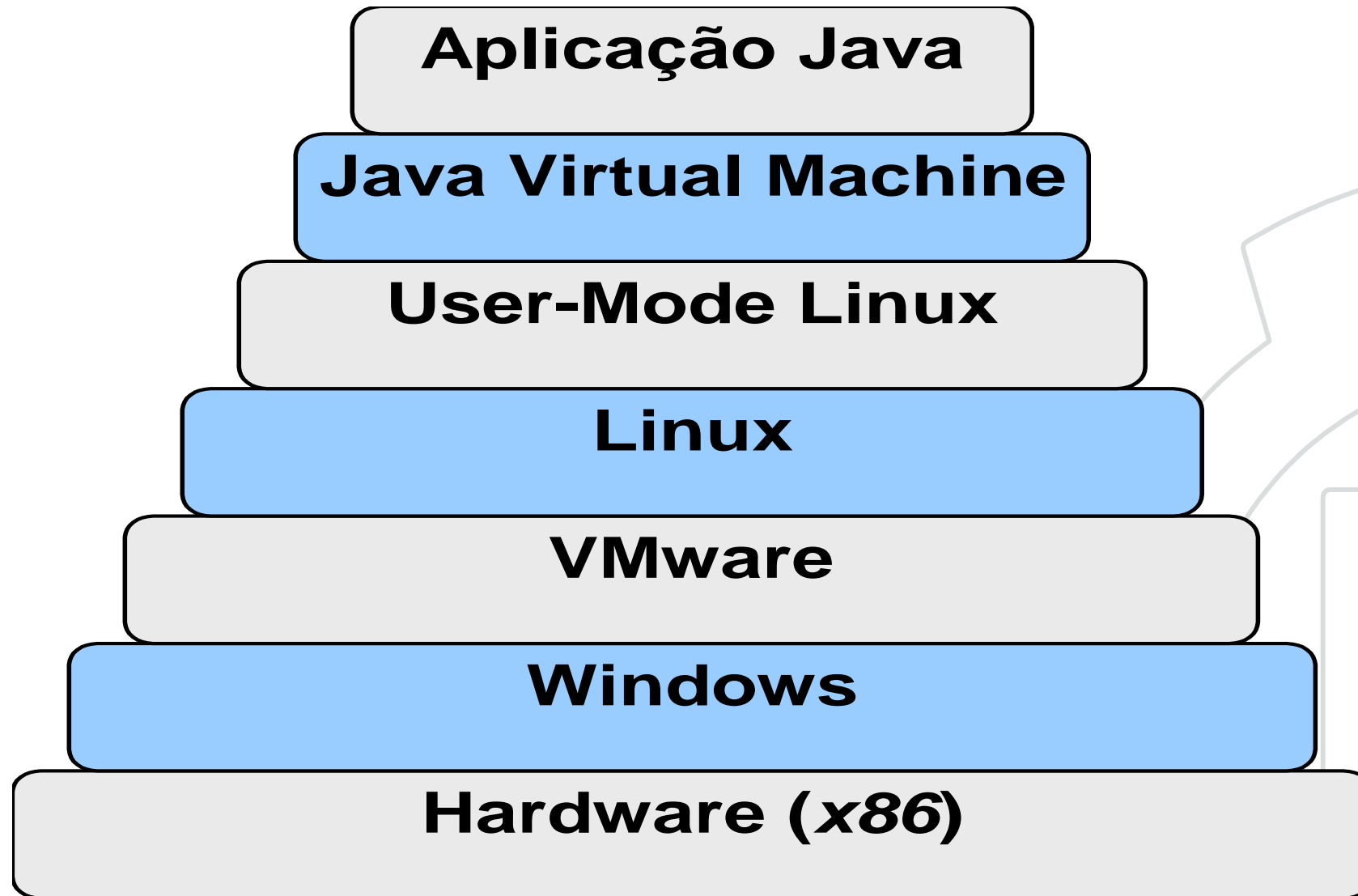
Classificação – Arquitetura: Máquina virtual

A **abstração** é uma forma simples de prover alguns recursos específicos de *hardware* para um *software*, enquanto a **virtualização** provê um conjunto completo de recursos.



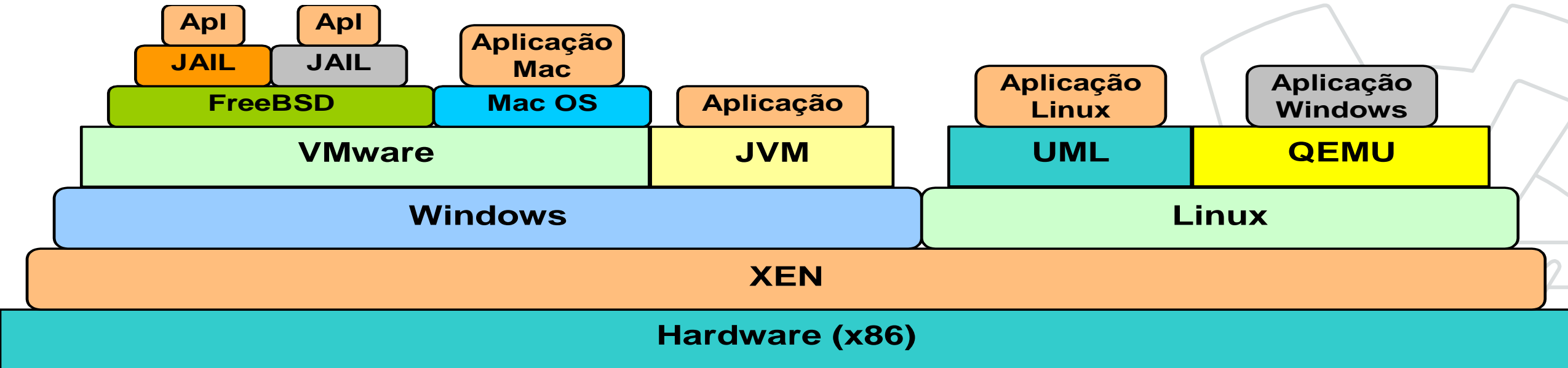
Sistema Operacional

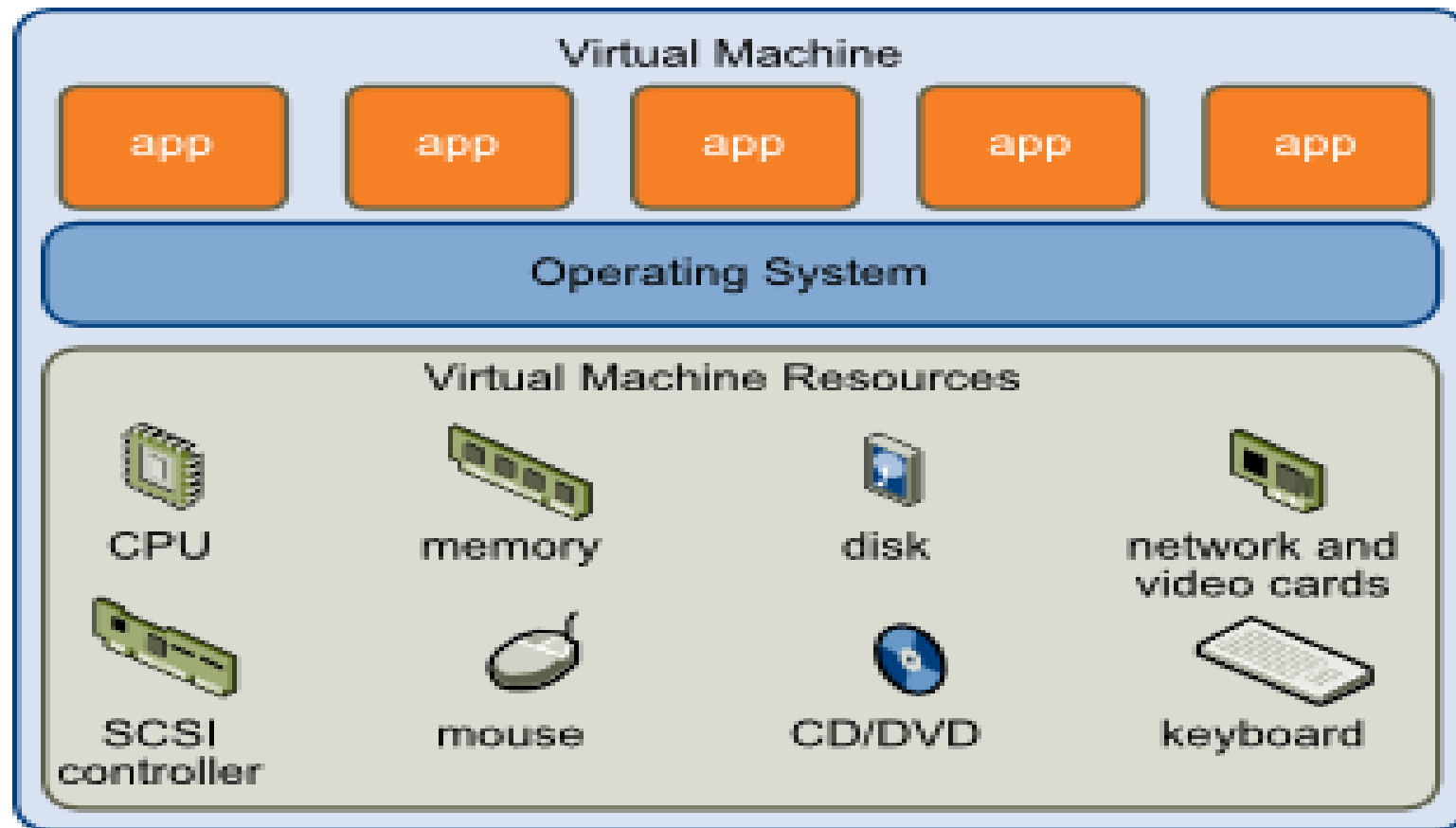
O poder da virtualização

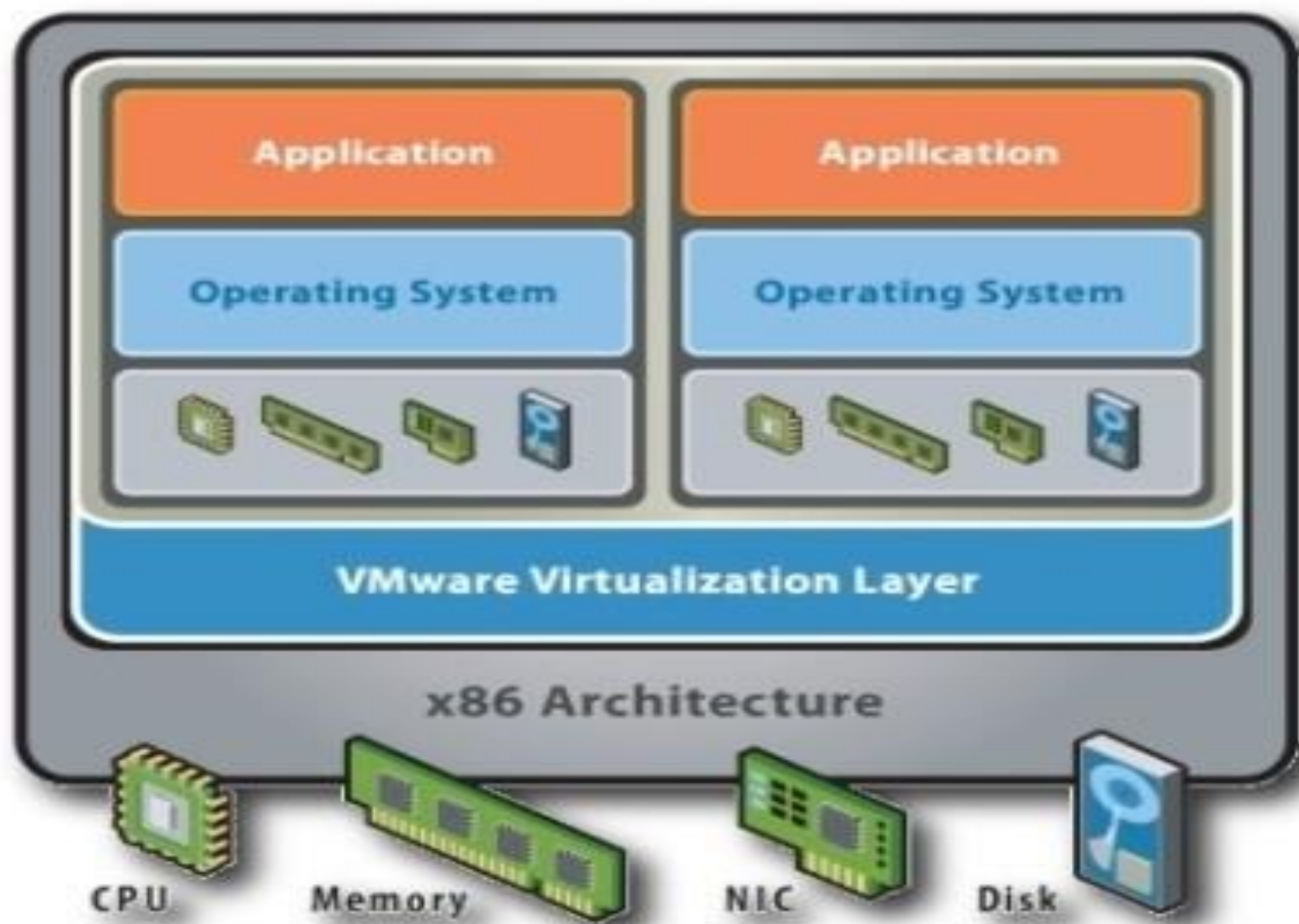


Sistema Operacional

O poder da virtualização

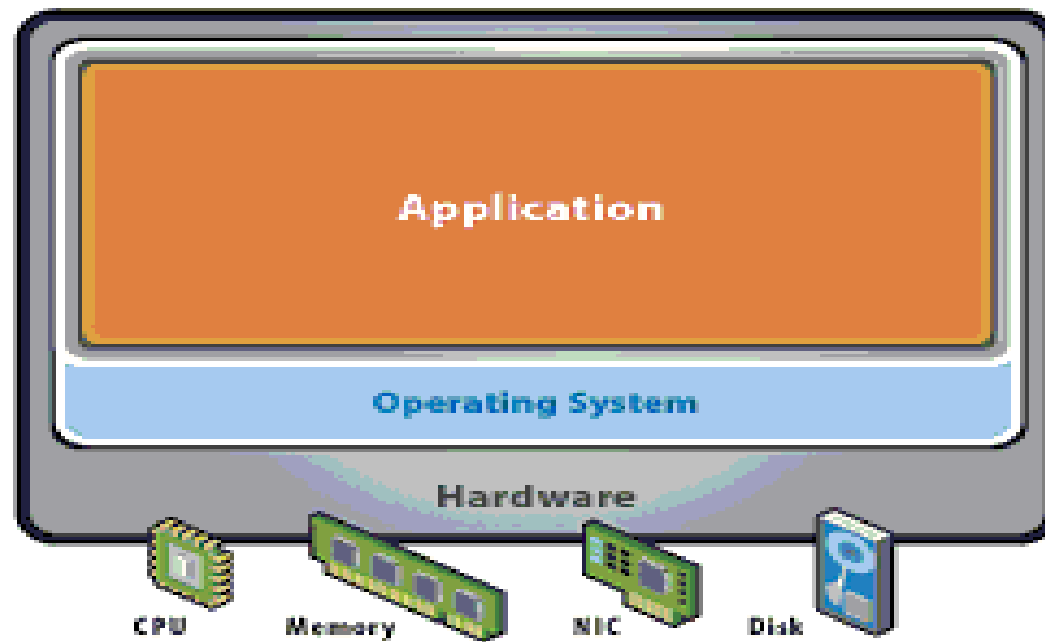




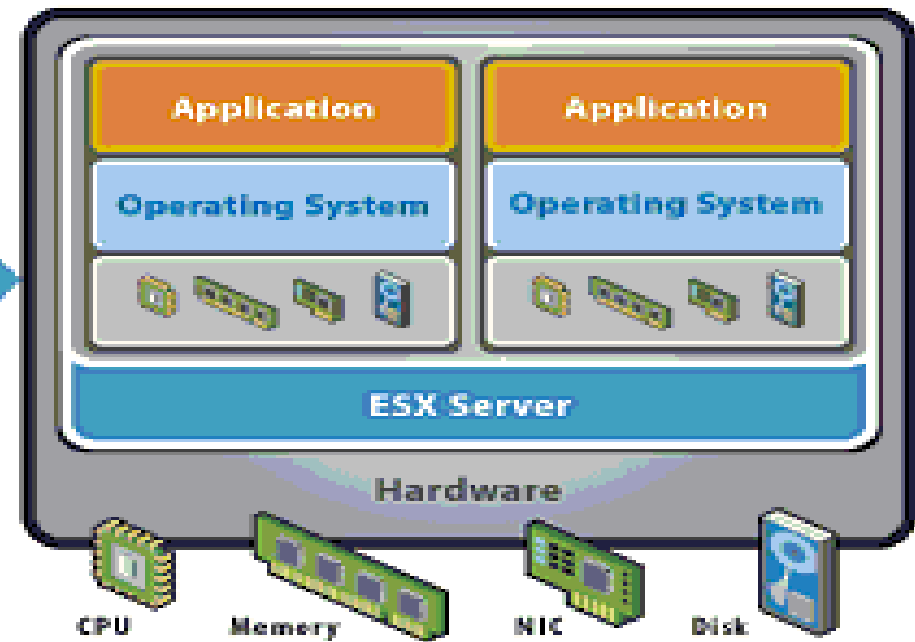




Without Virtualization



With Virtualization





VirtualBox



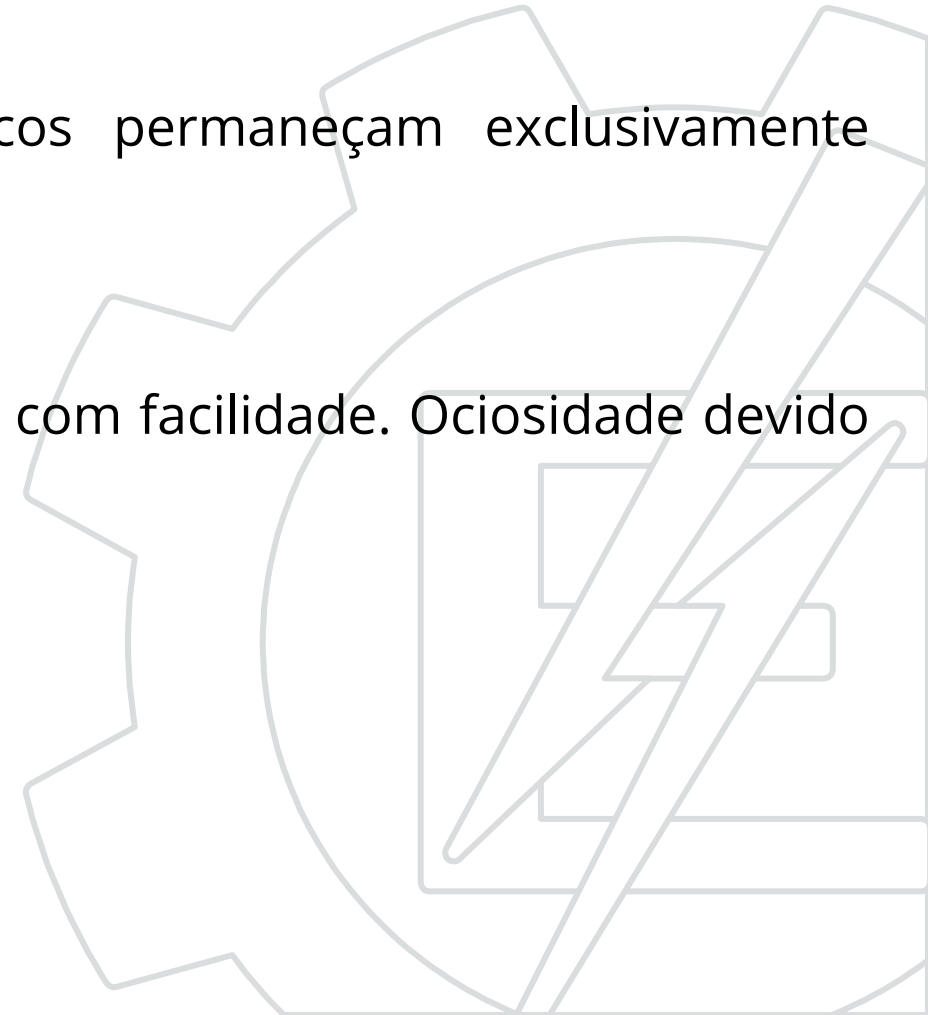
QEMU



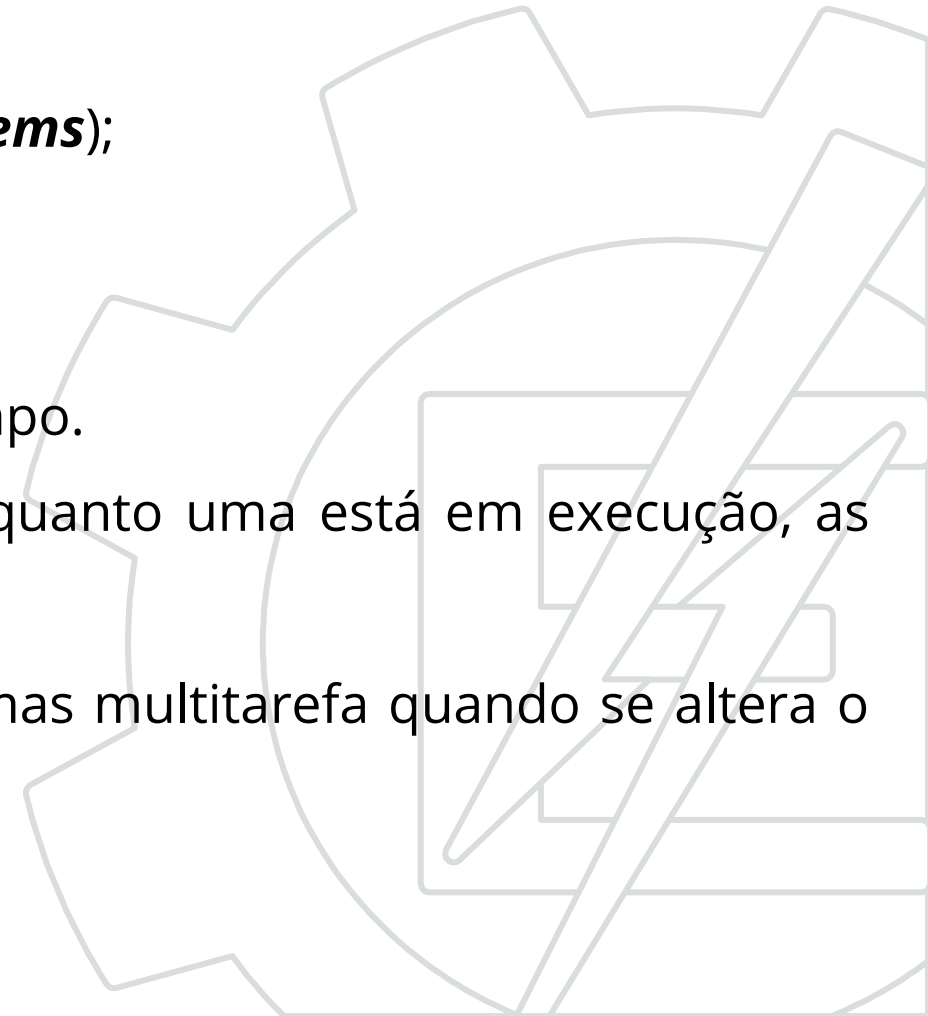
vmware

Parallels™
Optimized Computing

- Um sistema **monotarefa ou monoprogramado** executa uma tarefa de cada vez (serial).
- Permite que o processador, a memória e os periféricos permaneçam exclusivamente dedicados à execução de um único programa.
- Recursos são mal utilizados, entretanto são implementados com facilidade. Ociosidade devido a I/O-bound ou CPU-bound.
- Possíveis estados dos processos: *Novo - Execução – Término*.



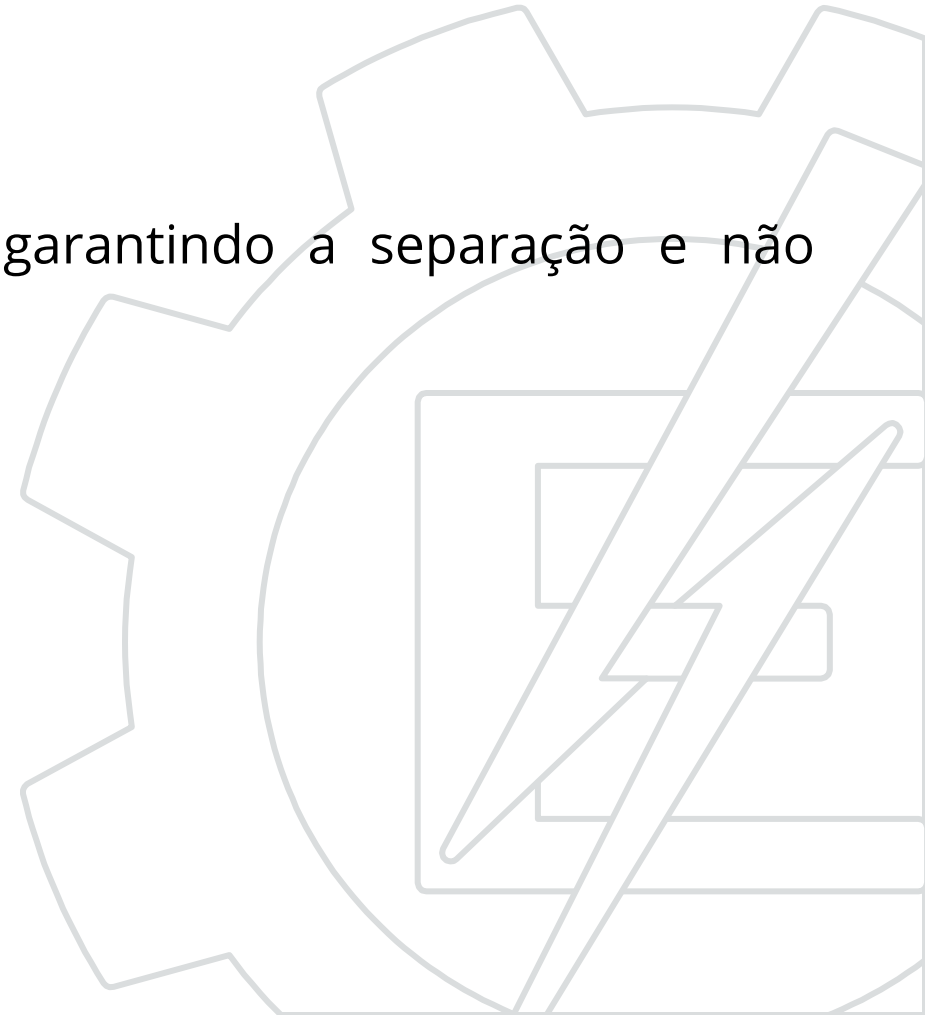
- Um sistema **multitarefa ou multiprogramado** executa várias tarefas simultaneamente (concorrente) e subdivide-se em:
 - a) Sistemas de tempo compartilhado (***time sharing systems***);
 - b) Sistemas de tempo real.
- Pode manter vários programas em memória ao mesmo tempo.
- Execução de várias tarefas em um único processador: enquanto uma está em execução, as outras estão em espera (*I/O-bound*).
- **Troca de contexto** é uma operação que ocorre nos sistemas multitarefa quando se altera o programa que está sendo executado.



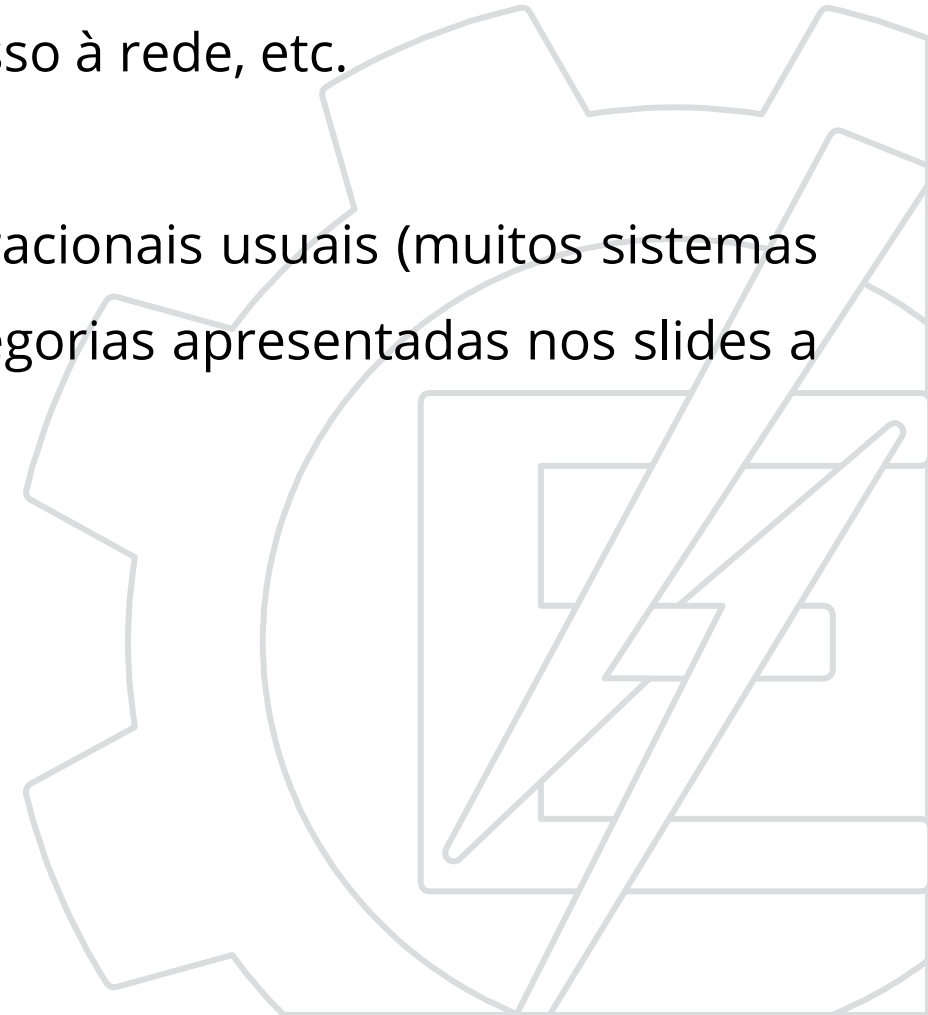
Sistema Operacional

Classificação – Usuários: **Monousuário** *versus* **Multiusuário**

- Um sistema **monousuário** não faz distinção entre os usuários, foi projetado para ser utilizado por um único usuário.
- Um sistema **multiusuário** permite diversos usuários garantindo a separação e não interferência entre as tarefas de cada um.



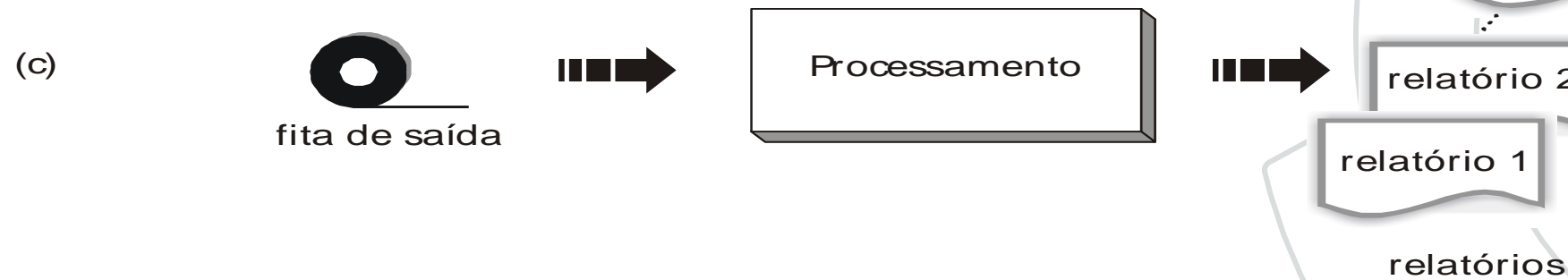
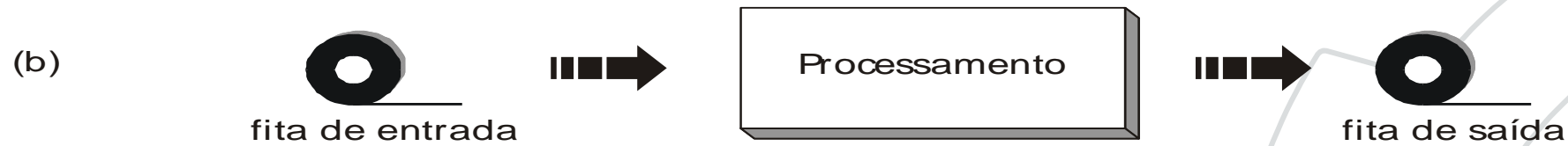
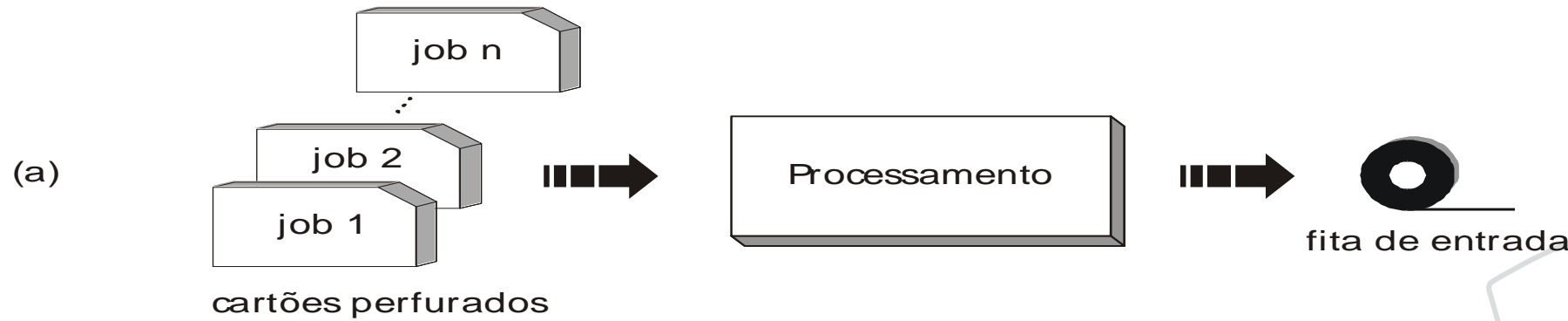
- Outro tipo de classificação está relacionada a diversos parâmetros e perspectivas, como tamanho, velocidade, suporte a recursos específicos, acesso à rede, etc.
- A seguir são apresentados alguns tipos de sistemas operacionais usuais (muitos sistemas operacionais se encaixam bem em mais de uma das categorias apresentadas nos slides a seguir).



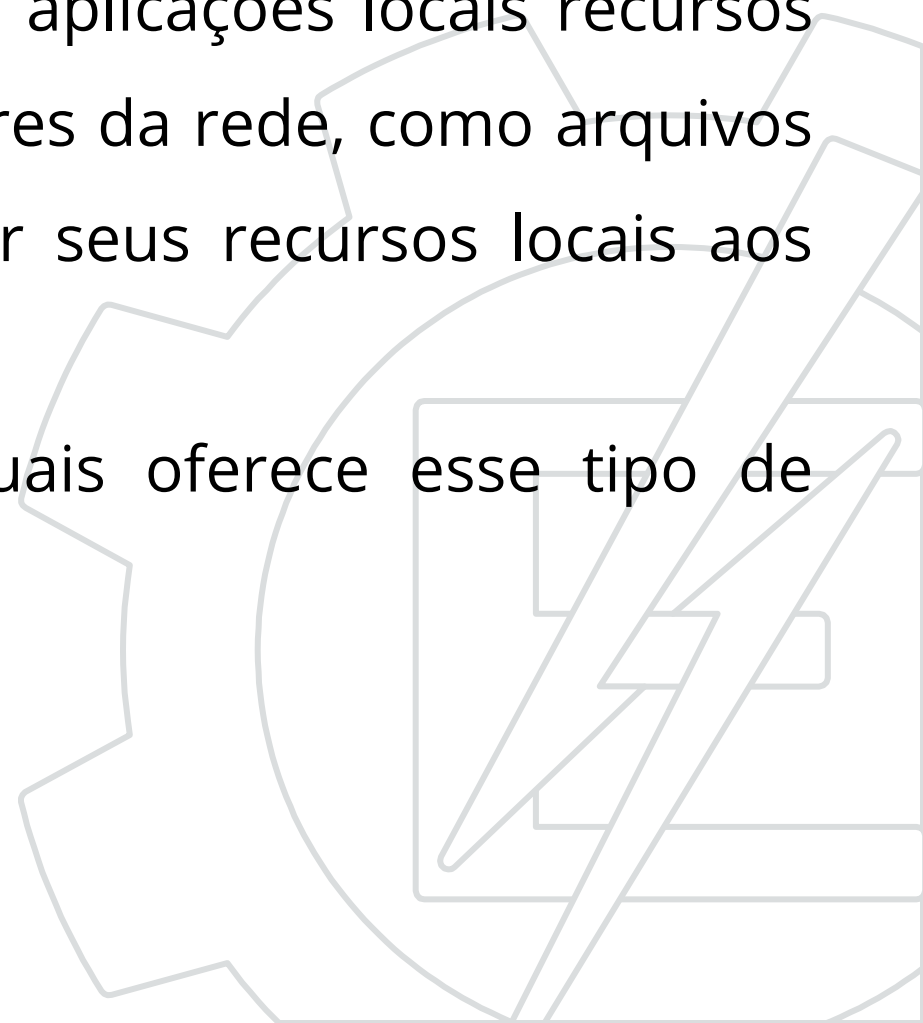
- **Batch** (de lote): os sistemas operacionais mais antigos trabalhavam “por lote”, ou seja, todos os programas a serem executados eram colocados em uma fila, com seus dados e demais informações para a execução. O processador recebia os programas e os processava sem interagir com os usuários, o que permitia um alto grau de utilização do sistema.
 - Atualmente, este conceito se aplica a sistemas que processam tarefas sem interação direta com os usuários, como os sistemas de processamento de transações em bancos de dados.

Sistema Operacional

Outras classificações

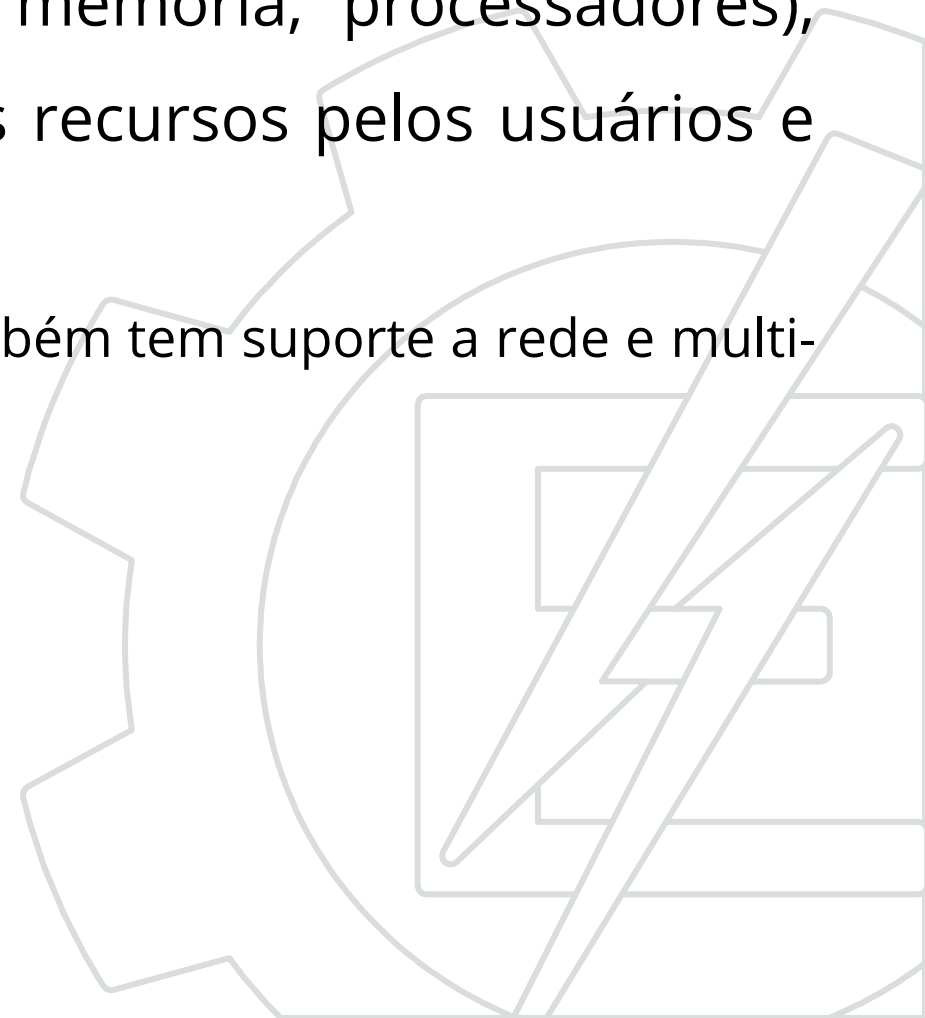


- **De rede:** um sistema operacional de rede deve possuir suporte à operação em rede, ou seja, a capacidade de oferecer às aplicações locais recursos que estejam localizados em outros computadores da rede, como arquivos e impressoras. Ele também deve disponibilizar seus recursos locais aos demais computadores, de forma controlada.
 - A maioria dos sistemas operacionais atuais oferece esse tipo de funcionalidade.



- **Distribuído:** em um sistema operacional distribuído, os recursos de cada máquina estão disponíveis globalmente, de forma transparente aos usuários. Ao lançar uma aplicação, o usuário interage com sua janela, mas não sabe onde ela está executando ou armazenando seus arquivos: o sistema é quem decide, de forma transparente.
 - Os sistemas operacionais distribuídos já existem há tempos (**Amoeba** e **Clouds**, por exemplo), mas ainda não são uma realidade de mercado.

- **Servidor:** um sistema operacional servidor deve permitir a gestão eficiente de grandes quantidades de recursos (disco, memória, processadores), impondo prioridades e limites sobre o uso dos recursos pelos usuários e seus aplicativos.
 - Normalmente um sistema operacional servidor também tem suporte a rede e multi-usuários.



- **Embarcado:** um sistema operacional é dito embarcado ou embutido (*embedded*) quando é construído para operar sobre um *hardware* com poucos recursos de processamento, armazenamento e energia. Aplicações típicas desse tipo de sistema aparecem em telefones celulares, controladores industriais e automotivos, equipamentos eletrônicos de uso doméstico (leitores de DVD, TVs, fornos-micro-ondas, centrais de alarme, etc.). Muitas vezes um sistema operacional embutido se apresenta na forma de uma biblioteca a ser ligada ao programa da aplicação (que é fixa).
 - Exemplos de sistemas operacionais embutidos são o **C/OS**, **LynxOS** e **VxWorks**.

- **Tempo real:** ao contrário da concepção usual, um sistema operacional de tempo real não precisa ser necessariamente ultra-rápido; sua característica essencial é ter um **comportamento temporal previsível** (ou seja, seu tempo de resposta deve ser conhecido no melhor e pior caso de operação).
 - A estrutura interna de um sistema operacional de tempo real deve ser construída de forma a minimizar esperas e latências imprevisíveis, como tempos de acesso a disco e sincronizações excessivas.

- **Tempo real** (cont.): Existem duas classificações de sistemas de tempo real:
 - ***Soft real-time systems***: nos quais a perda de prazos implica na degradação do serviço prestado. Um exemplo seria o suporte à gravação de CDs ou à reprodução de músicas.
 - ***Hard real-time systems***: a perda de prazos pelo sistema pode perturbar o objeto controlado, com graves consequências humanas, econômicas ou ambientais. Por exemplo, o controle de funcionamento de uma turbina de avião a jato ou de uma caldeira industrial.
- Exemplos de sistemas de tempo real incluem o **QNX**, **RT-Linux** e **VxWorks**.

Bibliografia

- TANENBAUM, Andrew S; BOS, Herbert. Sistemas operacionais modernos. 4a ed. São Paulo: Pearson Education do Brasil, 2016.

Capítulo 1.

<https://plataforma.bvirtual.com.br/Acervo/Publicacao/1233>

- DEITEL, H.M; DEITEL, P.J; CHOFFNES,D.R. Sistemas Operacionais. 3a ed. São Paulo: Pearson Prentice Hall, 2005. **Capítulo 1.**

<https://plataforma.bvirtual.com.br/Acervo/Publicacao/315>



Sistemas Operacionais

Prof. Otávio Gomes

otavio.gomes@unifei.edu.br

