EMB22109 - Sistemas Embarcados:

Introdução a SOs e Tipos de Kernel

João Cláudio Elsen Barcellos

Engenheiro Eletricista
Formado na Universidade Federal de Santa Catarina
campus Florianópolis
joaoclaudiobarcellos@gmail.com

5 de Junho de 2025

* *Créditos ao Prof. Emerson Ribeiro de Mello, o qual criou e disponibilizou o template aqui usado, via ShareLaTeX*

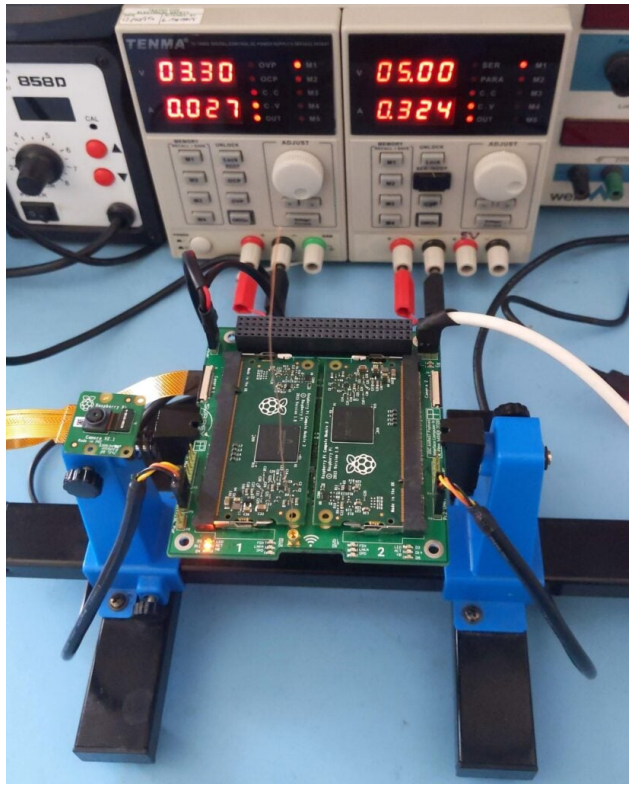
** *Créditos ao Prof. Hugo Marcondes, o qual forneceu parte do conteúdo usado nesta apresentação*

Na aula de hoje veremos...

Sistemas Embarcados

Revisão: Sistemas embarcados

1. São sistemas computacionais desenvolvidos para realizar funções específicas;
2. Usualmente constituem “sistemas maiores”;
3. Usualmente possuem diversas restrições e limitações (e.g., quantidade de memória, capacidade de processamento, gerenciamento de energia);
4. Por vezes, usados em aplicações de *real-time* e de alta confiabilidade (e.g., aplicações automotivas, aeroespaciais);
5. Usualmente têm interação direta com o ambiente físico.



Definição

An embedded system is a computerized system that is purpose built for its application [1].

Revisão: Software embarcado

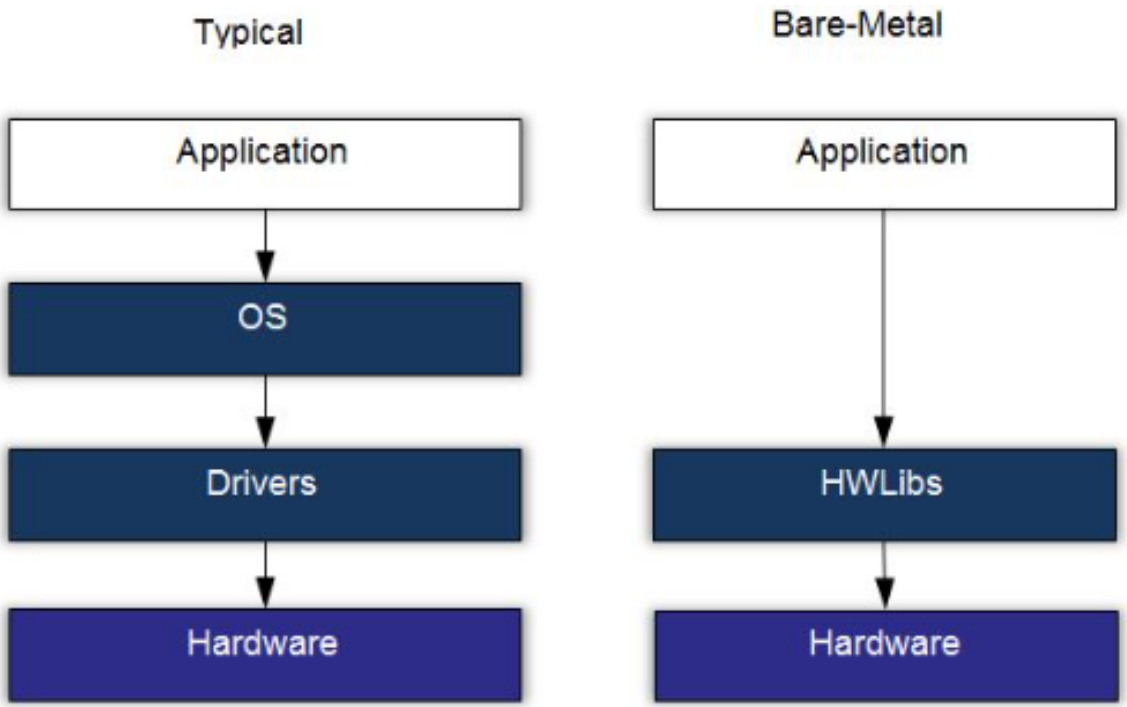
Bare-metal:

1. Controle direto e total do hardware;
2. Ausência de overhead de sistema operacional;
3. Máximo desempenho e otimização de recursos;
4. Maior complexidade e tempo de desenvolvimento;
5. Depuração mais desafiadora.

SO:

1. Camada de abstração de hardware;
2. Gerenciamento de multitarefas e recursos;
3. Facilita o desenvolvimento e aumenta a produtividade;
4. Pequeno overhead de CPU/memória;
5. Maior modularidade e manutenibilidade.

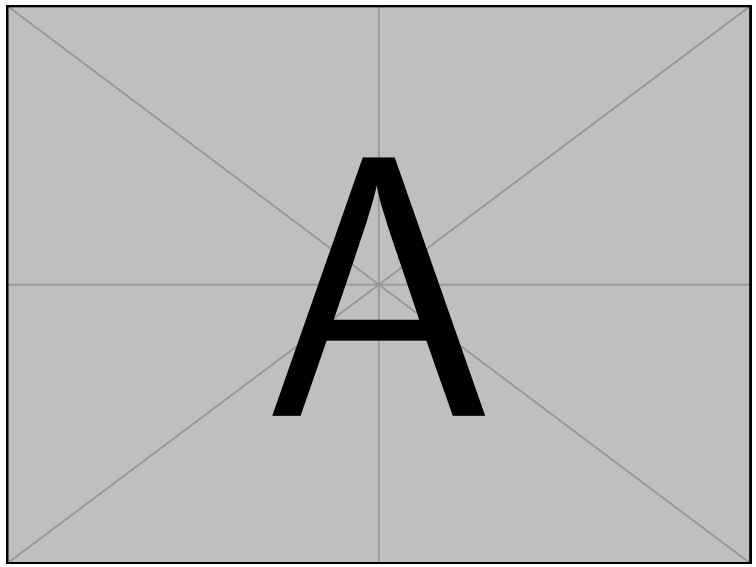
Abordagens de implementação do software embarcado



Introdução a SOs

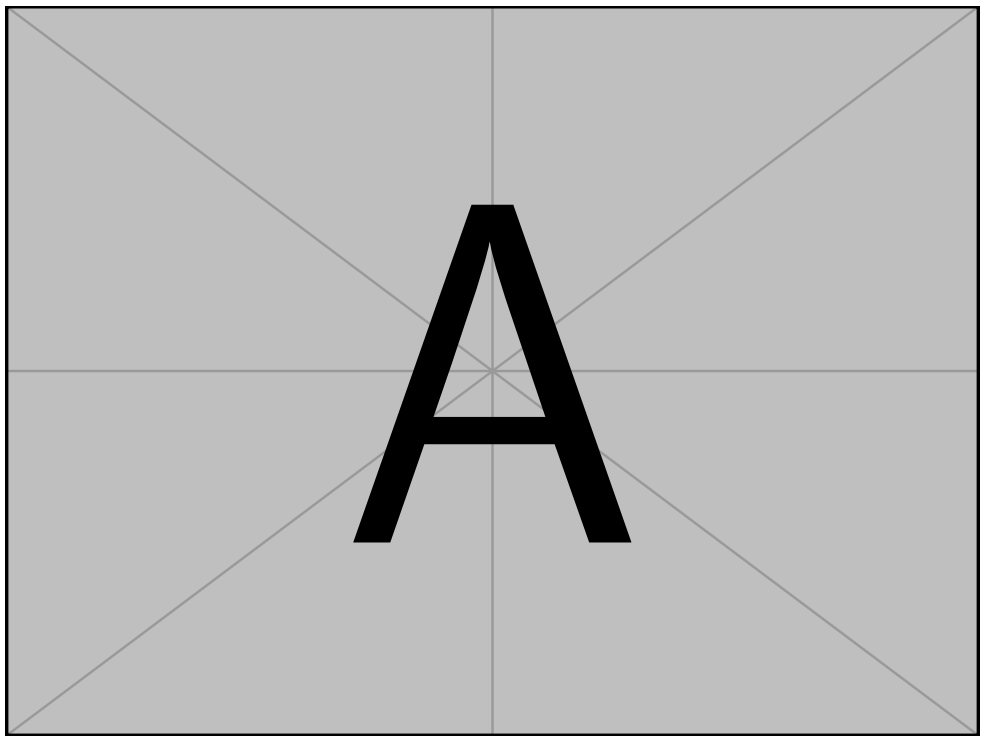
Sistema Operacional

1. Software essencial para FACILITAR o uso de sistemas computacionais;
2. Permite a execução de múltiplos programas (aparentemente) simultaneamente;
3. Gerencia os recursos de hardware do sistema computacional (e.g., CPU, memória, dispositivos de I/O);
4. Portanto, pode ser visto sob duas perspectivas principais: máquina virtual e gerenciador de recursos.



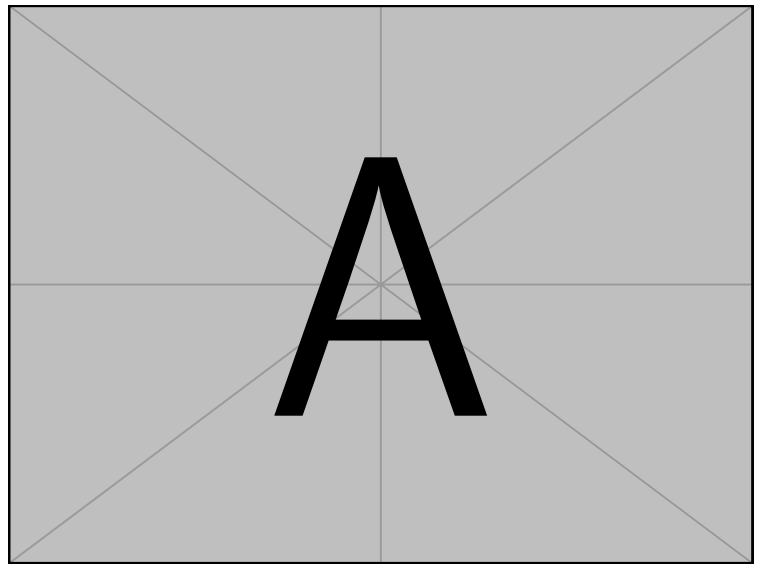
SO como “máquina virtual”

1. A principal técnica utilizada é a “virtualização”;
2. Transforma recursos de hardware (e.g., CPU, memória) em formas virtuais mais gerais, poderosas e fáceis de usar;
3. Abstrai detalhes complexos do hardware, provendo uma interface de mais alto nível;
4. Oferece interfaces (APIs/chamadas de sistema) para que programas acessem funcionalidades e recursos;
5. Contribui para a portabilidade das aplicações.



SO como gerenciador de recursos

1. Gerencia e aloca os recursos do sistema entre múltiplos programas:
   - 1.1 CPU (agendamento de processos/tarefas);
   - 1.2 Memória (alocação e proteção);
   - 1.3 Dispositivos de E/S (compartilhamento);
2. Controla a execução de programas;
3. Previne erros e o uso indevido de recursos;
4. Otimiza o uso dos recursos com base em diversos objetivos.



Kernel de um SO

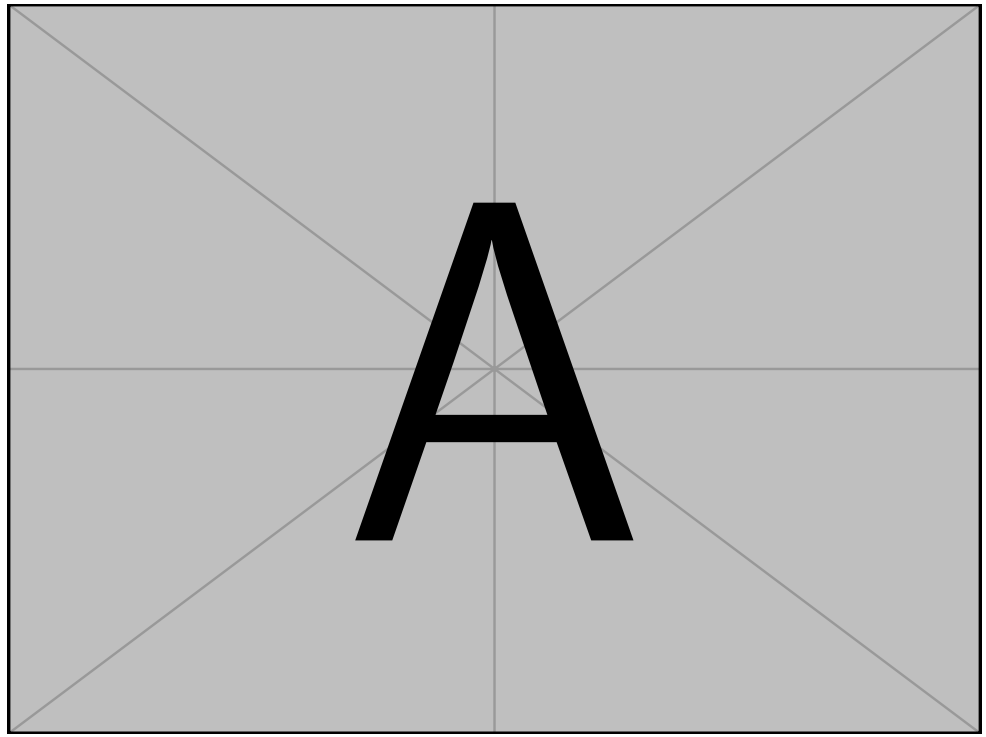
1. **Kernel = Núcleo**
   - 1.1 Parte central e fundamental de um programa/algoritmo

2. **Kernel = Modo Kernel (Supervisor)**
   - 2.1 Parte de um programa que executa em modo Kernel
   - 2.2 Modo Kernel -> Espaço/Modo de Execução
   - 2.3 Suporte do Processador a diferentes espaços ou níveis de execução (modo Kernel e modo usuário)

Kernel de um SO

1. Núcleo do Sistema Operacional
2. Provê o Gerenciamento do Sistema e Funcionalidades Básicas
3. Executa em modo privilegiado (modo kernel)
4. Controla acesso direto ao hardware
5. Implementa serviços fundamentais:
   - 5.1 Gerenciamento de memória
   - 5.2 Gerenciamento de processos
   - 5.3 Gerenciamento de dispositivos
   - 5.4 Comunicação entre processos

Ambientes de Execução



1. Diferentes níveis de acesso ao hardware
2. Proteção contra operações indevidas
3. Separação entre código privilegiado e não-privilegiado

Tipos de Kernel

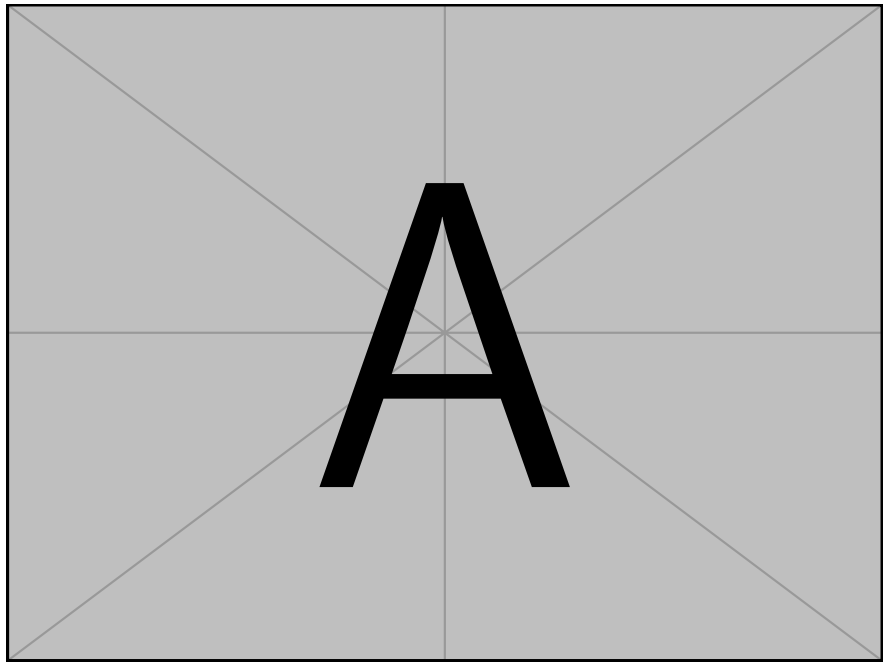
Abordagens de Design de Kernel

1. **Kernel Monolítico**
2. **Microkernel**
   - 2.1 Nanokernel
3. **Kernel Híbrido**
4. **Exokernel**

**Kernel Híbrido é tido como um termo controverso dentre a comunidade científica da área [2].*

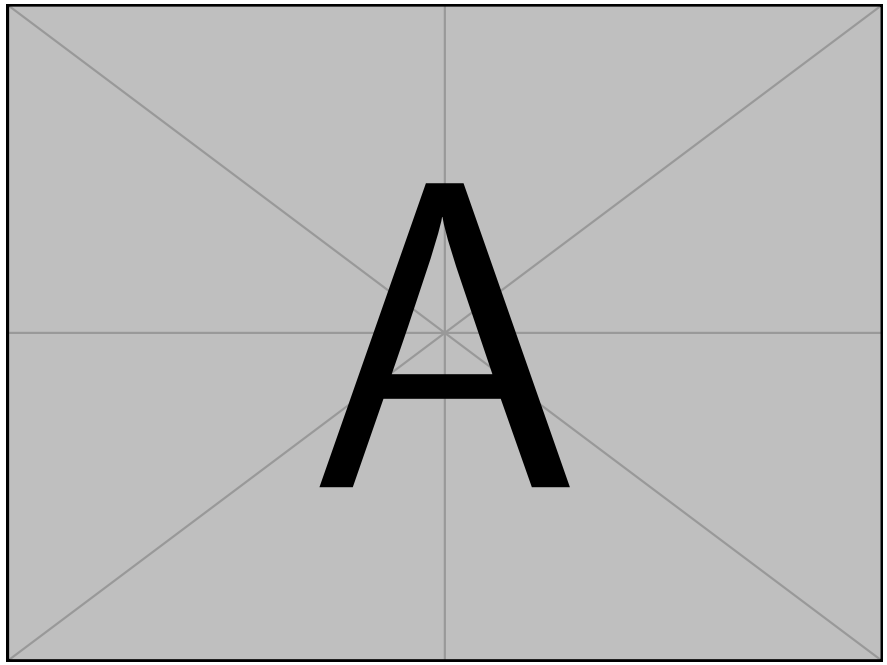
Kernel Monolítico

1. Todo o sistema operacional executa em modo kernel
2. Abordagem tradicional
3. Exemplos:
   - 3.1 Unix tradicional
   - 3.2 Linux
   - 3.3 FreeBSD
4. Vantagens:
   - 4.1 Desempenho
   - 4.2 Acesso direto ao hardware
5. Desvantagens:
   - 5.1 Menor segurança
   - 5.2 Falhas afetam todo o sistema



Microkernel

1. Apenas funções essenciais no kernel
2. Serviços executam como processos em modo usuário
3. Comunicação via IPC (Comunicação Entre Processos)
4. Exemplos:
   - 4.1 MINIX
   - 4.2 QNX
   - 4.3 L4
5. Vantagens:
   - 5.1 Maior segurança
   - 5.2 Falhas isoladas
   - 5.3 Código mais modular



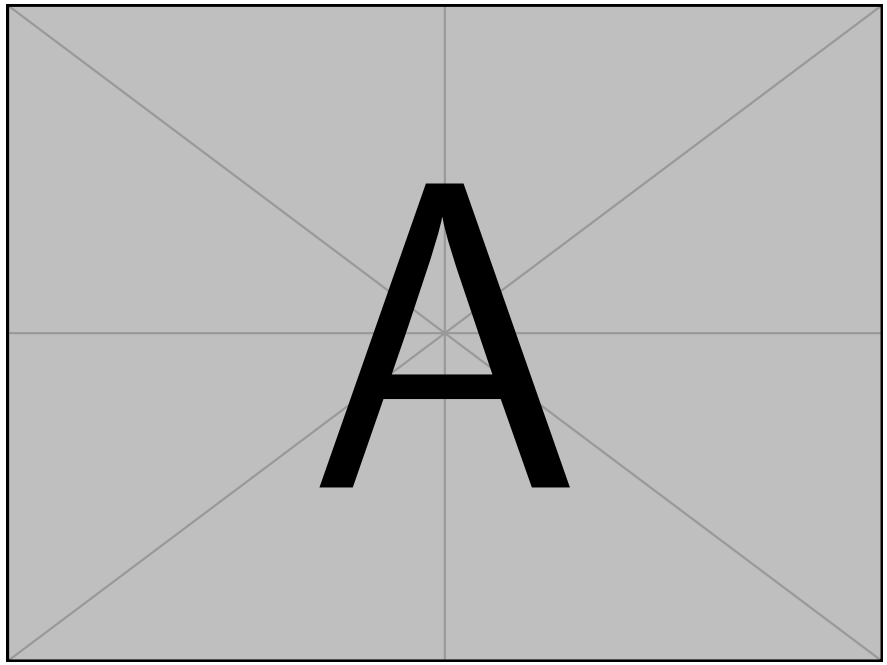
Conceitos e Objetivos do Microkernel

1. **Segurança**
   - 1.1 Menor código possível executando em modo Kernel
   - 1.2 Menos tempo de execução do sistema em modo privilegiado
2. **Separação de Serviços em Servidores**
   - 2.1 Serviços executam como processos independentes
   - 2.2 Isolamento de falhas
3. **Comunicação via IPC**
   - 3.1 Baseada em troca de mensagens
   - 3.2 Mecanismo fundamental para operação do sistema

Abordagem proposta por Liedtke [3] para melhorar a segurança e modularidade

Kernel Híbrido

1. Combina características de kernel monolítico e microkernel
2. Alguns serviços no espaço do kernel, outros como processos de usuário
3. Exemplos:
   - 3.1 Windows NT (Windows 10/11)
   - 3.2 macOS (XNU)
   - 3.3 DragonFly BSD
4. Vantagens:
   - 4.1 Equilíbrio entre desempenho e modularidade
   - 4.2 Flexibilidade de design



Exokernel

1. Abordagem minimalista e radical
2. Kernel fornece apenas proteção e multiplexação de recursos
3. Bibliotecas em nível de usuário (LibOS) implementam abstrações tradicionais
4. Desenvolvido no MIT nos anos 90 [4]
5. Vantagens:
   - 5.1 Máxima flexibilidade para aplicações
   - 5.2 Personalização de abstrações
   - 5.3 Potencial para alto desempenho
6. Desvantagens:
   - 6.1 Complexidade para desenvolvedores de aplicações
   - 6.2 Menor portabilidade
   - 6.3 Poucos sistemas em produção

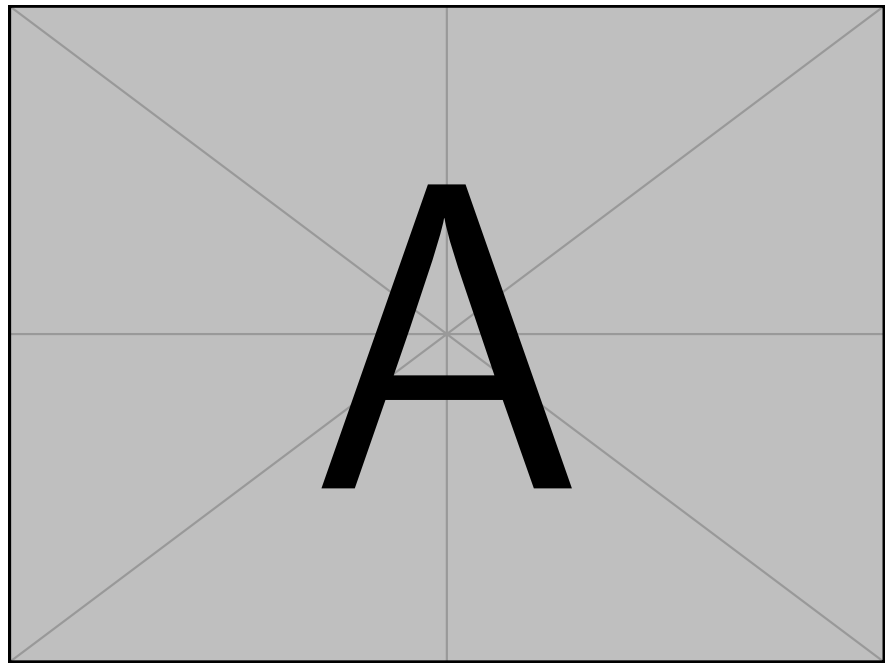
Comparação entre Tipos de Kernel

Característica	**Monolítico**	**Microkernel**	**Híbrido**	**Exokernel**
Tamanho do kernel	Grande	Pequeno	Médio	Mínimo
Desempenho	Alto	Moderado	Alto/Moderado	Potencial alto
Modularidade	Baixa	Alta	Média	Alta
Segurança	Menor	Maior	Média	Variável
Complexidade	Alta	Média	Alta	Baixa (kernel) Alta (LibOS)
Exemplos	Linux, FreeBSD	MINIX, QNX	Windows NT, macOS	MIT Exokernel, Nemesis

Cada tipo de kernel apresenta vantagens e desvantagens específicas [5]

Interação de um Sistema com SO

1. Comunicação entre aplicações e kernel:
   - 1.1 Kernel Monolítico: System Calls e Sinais de Interrupções
   - 1.2 Microkernel: IPC, Kernel Calls e Sinais de Interrupções
2. Diferenças fundamentais:
   - 2.1 Monolítico: chamadas diretas ao kernel
   - 2.2 Microkernel: comunicação via mensagens entre processos



Tendências Atuais em Design de Kernel

1. Sistemas operacionais modernos tendem a adotar abordagens híbridas
2. Linux: kernel monolítico com módulos carregáveis
3. Windows: kernel híbrido com componentes em modo kernel e modo usuário
4. Sistemas embarcados: microkernels ganham popularidade pela segurança
5. Virtualização e containers: novas demandas para design de kernel
6. Sistemas de tempo real: foco em previsibilidade e determinismo

A escolha do tipo de kernel para sistemas embarcados depende dos requisitos específicos da aplicação [6]

Resumo

1. Sistemas Embarcados podem ser implementados:
   - 1.1 Bare Metal: programação direta no hardware
   - 1.2 Com Sistema Operacional: camada de abstração
2. Sistemas Operacionais são camadas de software que gerenciam recursos de hardware
3. Podem ser vistos como Máquinas Virtuais ou Gerenciadores de Recursos
4. O Kernel é o núcleo do sistema operacional, executando em modo privilegiado
5. Principais tipos de kernel:
   - 5.1 Monolítico: todo o SO em modo kernel (Linux)
   - 5.2 Microkernel: mínimo em modo kernel, serviços em modo usuário (MINIX)
   - 5.3 Híbrido: combinação das abordagens anteriores (Windows NT)
   - 5.4 Exokernel: apenas proteção e multiplexação de recursos

Referências

References I

[1] Elecia White. *Making Embedded Systems: Design Patterns for Great Software*. O'Reilly Media, Inc., 2024.

[2] Raoni Fassina Firmino, Glauber Módolo Cabral e Aleksey Victor Trevelin Covacevice. *Design de Kernels: Microkernel, Exokernel e novos Sistemas Operacionais*. Universidade Estadual de Campinas - UNICAMP, Instituto de Computação - IC. 2007.

[3] Jochen Liedtke. "On micro-kernel construction". Em: *ACM SIGOPS Operating Systems Review*. Vol. 29. 5. ACM. 1995, pp. 237–250.

[4] Dawson R. Engler, M. Frans Kaashoek e James O'Toole Jr. "Exokernel: An operating system architecture for application-level resource management". Em: *ACM SIGOPS Operating Systems Review*. Vol. 29. 5. ACM. 1995, pp. 251–266.

References II

[5] Andrew S. Tanenbaum e Herbert Bos. *Modern Operating Systems*. 4ª ed. Pearson, 2015.

[6] William Stallings. *Operating Systems: Internals and Design Principles*. 9ª ed. Pearson, 2018.