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

8 de Junho de 2025

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

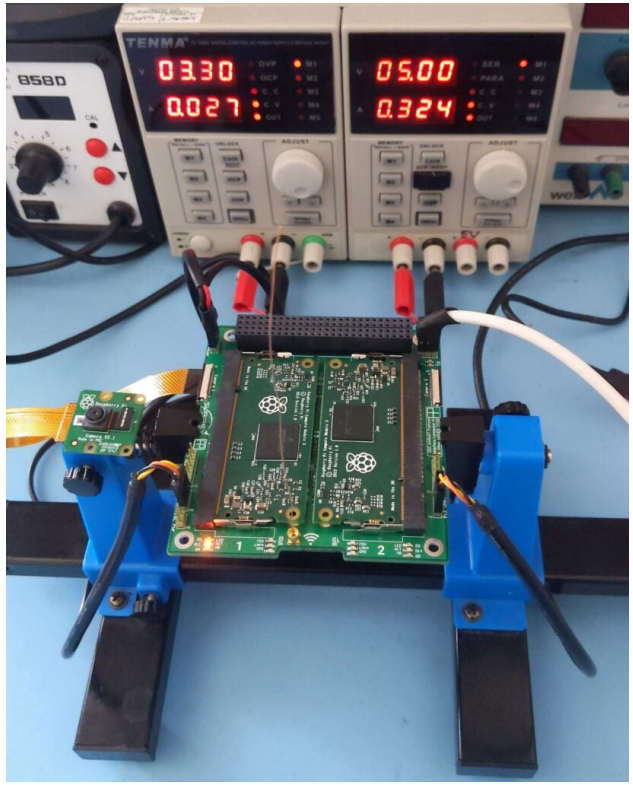
** *Créditos ao Prof. Hugo Marcondes, o qual forneceu parte do conteúdo usado nesta apresentação*

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.



Fonte: Retirado de [1]

Definição

An embedded system is a computerized system that is purpose-built for its application [2].

Revisão: software embarcado

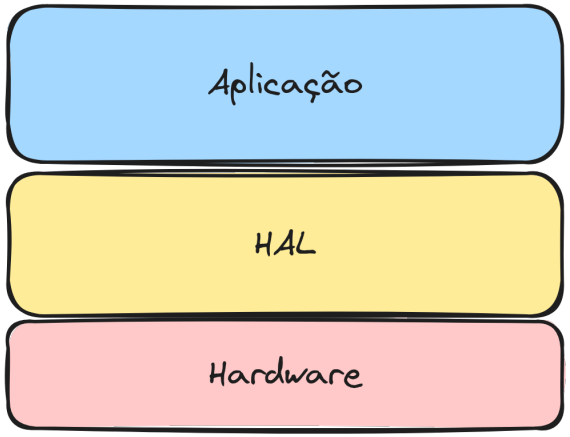
Bare-metal:

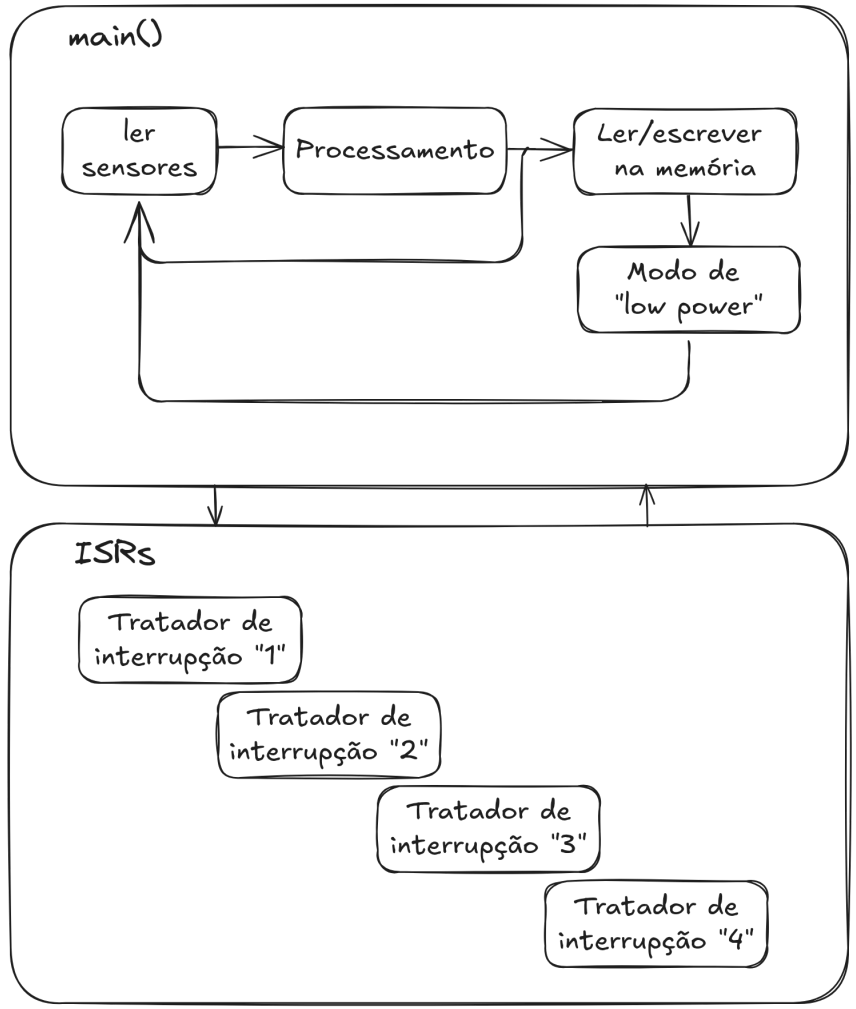
1. Controle direto e total do hardware;
2. Ausência de overhead de sistema operacional;
3. Máximo desempenho e otimização de recursos;
4. Maior complexidade e tempo de desenvolvimento;
5. Depuração mais desafiadora.

SO:

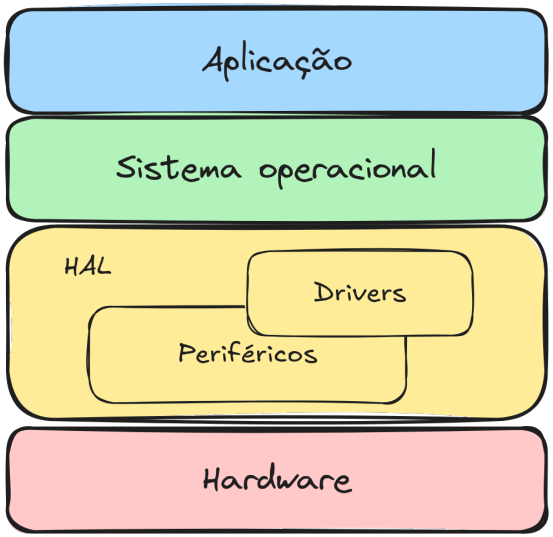
1. Camada de abstração de hardware;
2. Gerenciamento de multitarefas e recursos;
3. Facilita o desenvolvimento e aumenta a produtividade;
4. Pequeno overhead de CPU/memória;
5. Maior modularidade e manutenibilidade.

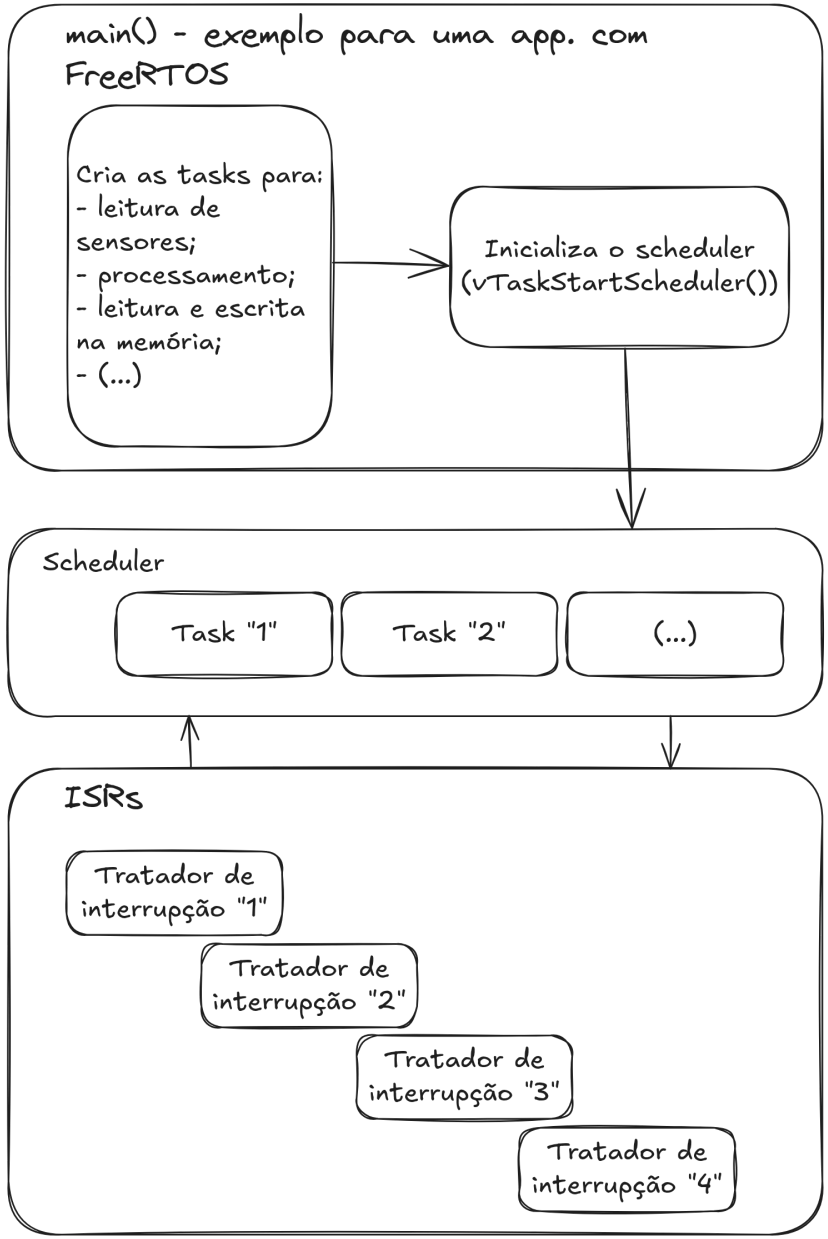
Revisão: software embarcado





Revisão: software embarcado

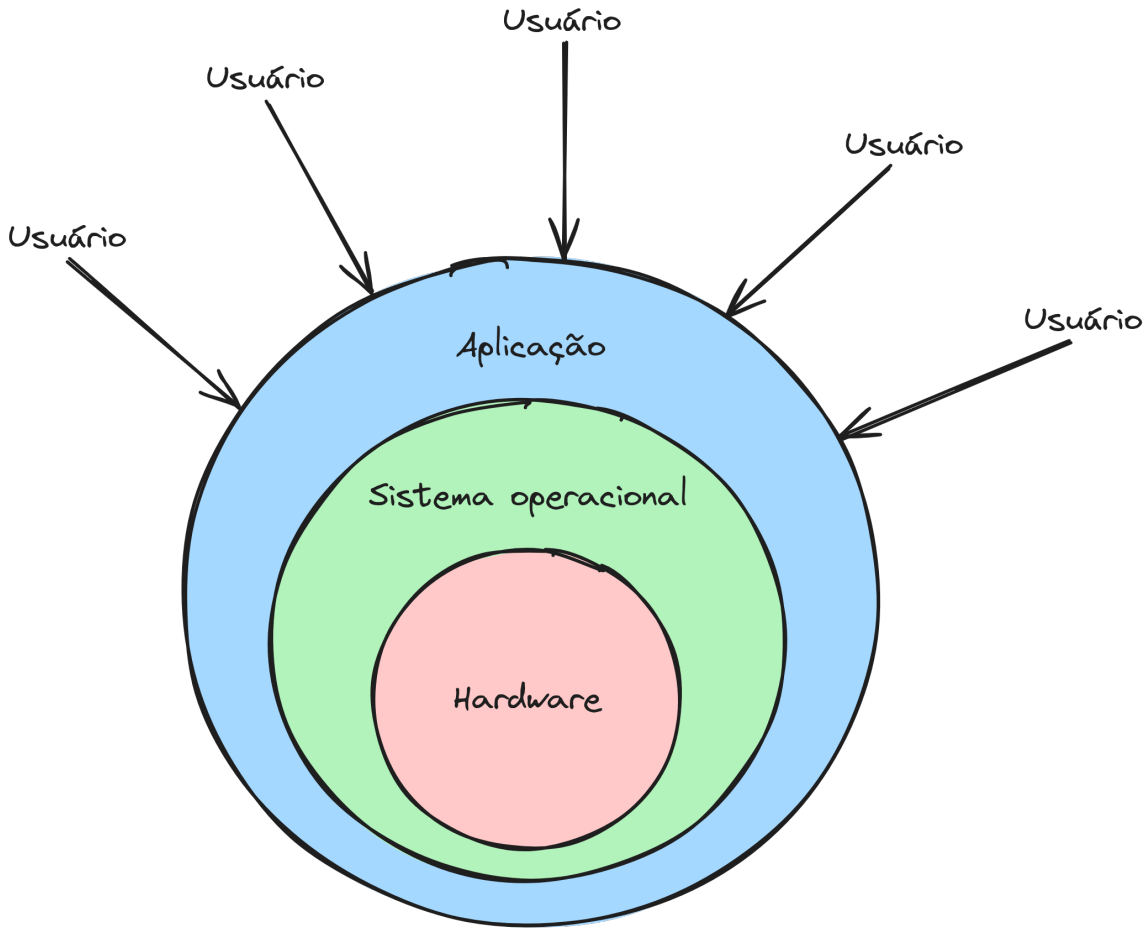




Introdução a SOs

Sistema Operacional

1. É o software que facilita 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: como sendo uma "máquina virtual" e um 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.

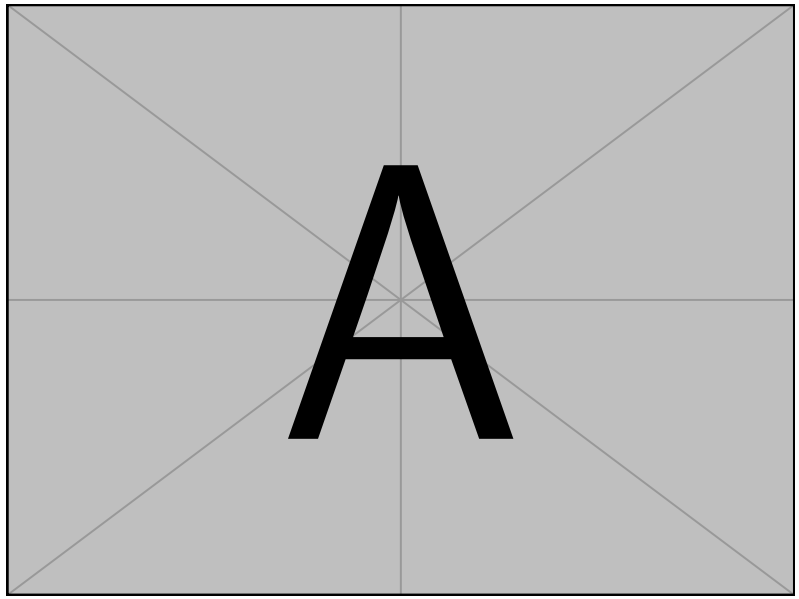
SO como gerenciador de recursos

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.

O núcleo de um SO: kernel

Kernel e modos de execução

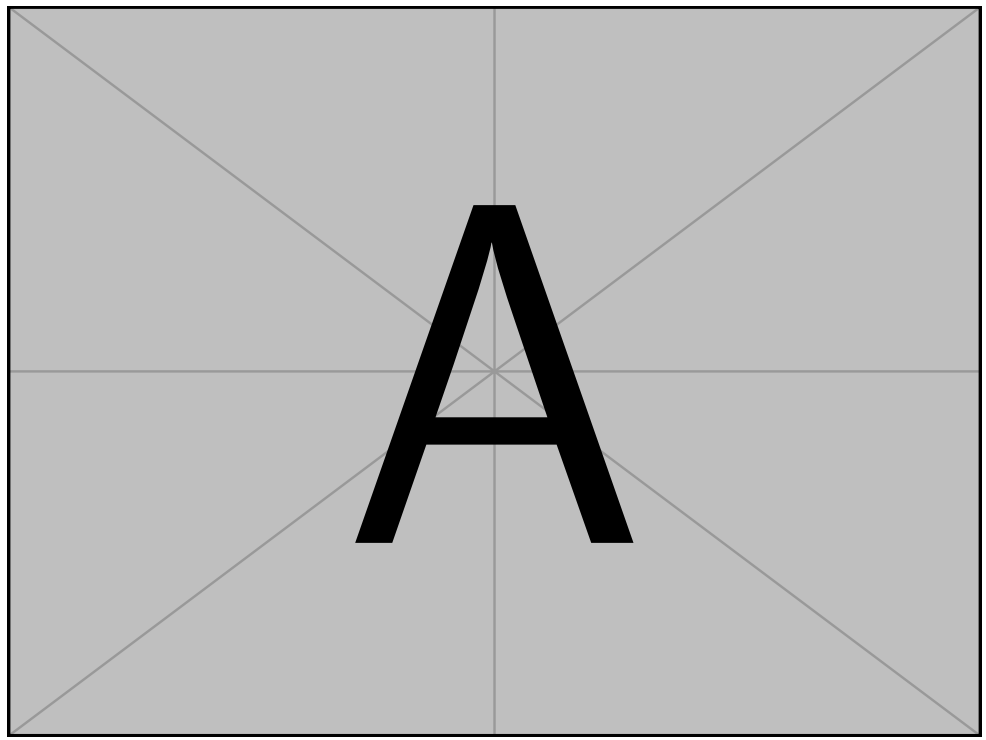
1. O kernel é o núcleo do sistema operacional:
   - 1.1 Primeira parte carregada durante o boot;
   - 1.2 Executa em **modo privilegiado** (*kernel mode*);
   - 1.3 Responsável por gerenciar recursos do sistema.
2. O processador opera em dois modos:
   - 2.1 **Modo Usuário**: instruções restritas, sem acesso direto ao hardware;
   - 2.2 **Modo Kernel**: acesso total a todos os recursos do sistema.



Transições entre modos ocorrem via interrupções e chamadas de sistema (syscalls)

Funções Típicas do Kernel

1. Gerenciamento de processos (criação, escalonamento, finalização);
2. Gerenciamento de memória (alocação, paginação, proteção);
3. Gerenciamento de dispositivos (drivers, interrupções);
4. Sistema de arquivos (organização, acesso, permissões);
5. Comunicação entre processos (IPC);
6. Segurança (controle de acesso, isolamento).



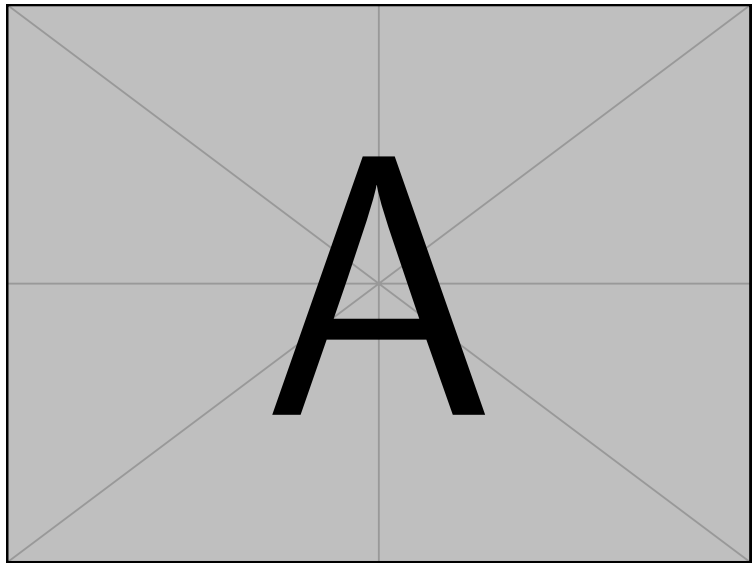
Espaço de Endereçamento e Isolamento

O kernel opera em um espaço de memória separado (espaço kernel);

Aplicativos operam em **modo usuário**, sem acesso direto ao hardware;

A troca entre modos é mediada pelo hardware (MMU + privilégios);

Falhas no modo kernel afetam todo o sistema; no modo usuário, apenas o processo.



Drivers no kernel: maior desempenho, maior risco.

Tipos de kernel

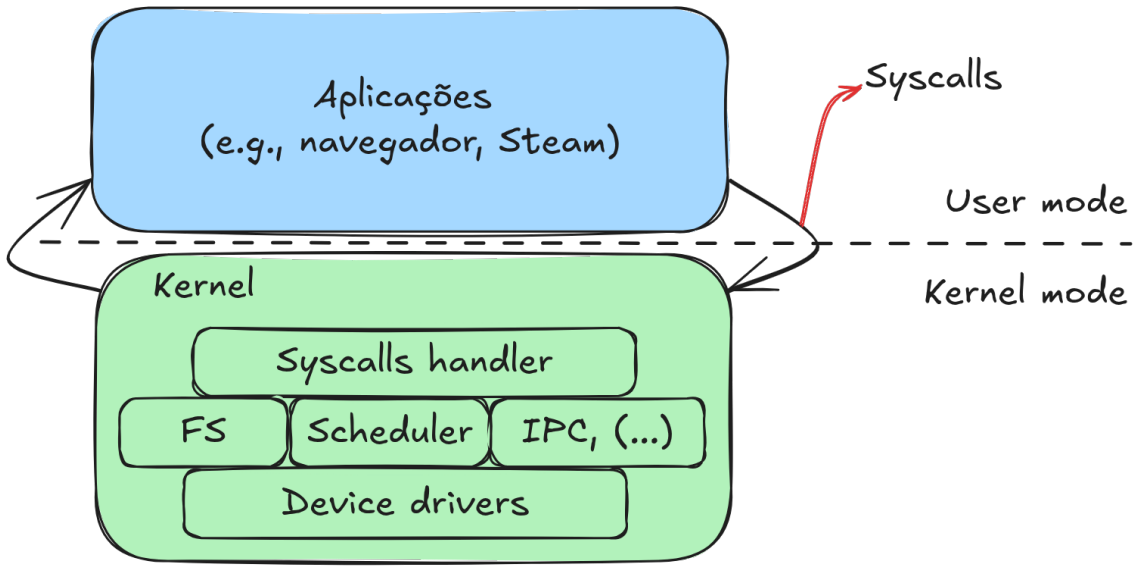
Abordagens de design de kernel

1. **Monolítico**: kernel grande e unificado, todos os serviços no modo kernel.
2. **Microkernel**: apenas funções essenciais no kernel; serviços externos no modo usuário.
3. **Híbrido**: mistura características dos dois modelos anteriores.
4. **Exokernel**: abstrai o mínimo; expõe recursos diretamente às aplicações.

** O termo "kernel híbrido" é discutido na literatura [3].*

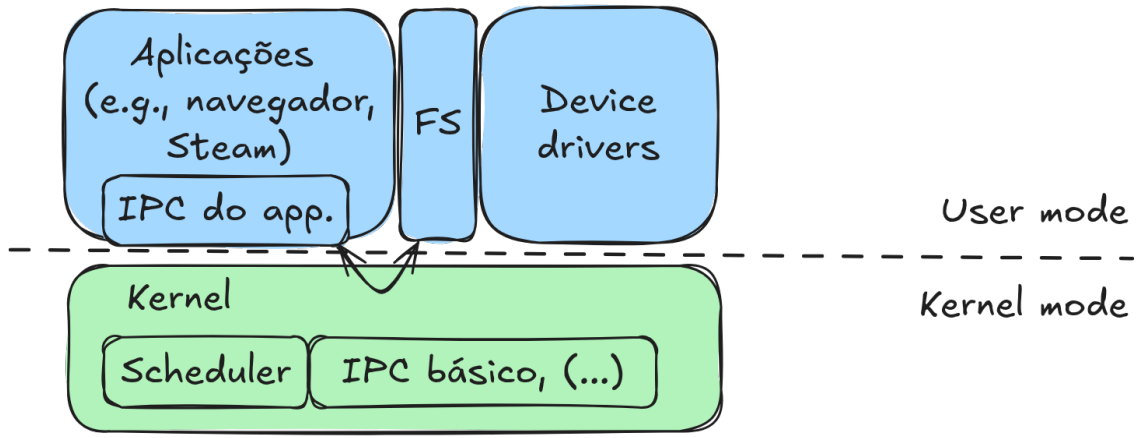
Kernel Monolítico

1. Todos os serviços operacionais rodam em modo kernel (drivers, arquivos, rede, etc.);
2. Vantagens: alto desempenho, comunicação interna eficiente;
3. Desvantagens: menor isolamento, falhas podem comprometer todo o sistema;
4. Exemplos: Linux, FreeBSD, Unix tradicional.



Microkernel

1. Mantém no kernel apenas funções essenciais: IPC, gerenciamento de processos e interrupções;
2. Drivers, sistema de arquivos e rede rodam como serviços em modo usuário;
3. Alta modularidade e isolamento de falhas;
4. Comunicação via troca de mensagens (IPC);
5. Exemplos: MINIX, QNX, seL4.

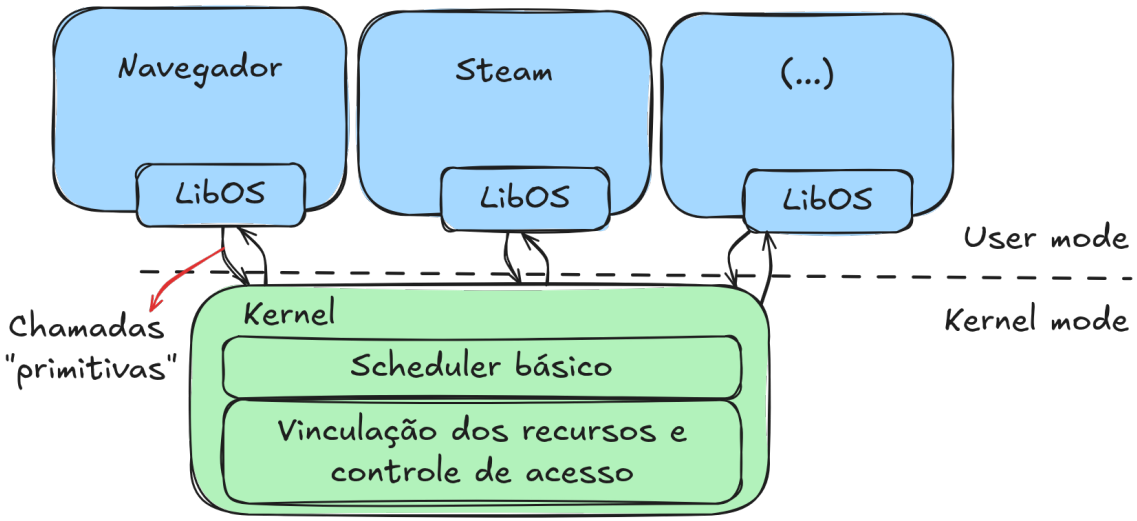


Kernel Híbrido

1. Combina características de microkernel e monolítico;
2. Alguns serviços rodam em modo kernel para ganho de desempenho;
3. Outros rodam isoladamente em modo usuário;
4. Exemplos: Windows NT, macOS (XNU), DragonFly BSD;
5. Equilíbrio entre segurança, desempenho e manutenção.

Exokernel

1. Não abstrai recursos; apenas controla acesso seguro e eficiente;
2. Aplicações escolhem como usar CPU, memória, disco, rede;
3. Abstrações tradicionais (sockets, arquivos, processos) são delegadas a bibliotecas no espaço do usuário;
4. Projetado para máxima flexibilidade e desempenho;
5. Exemplo: ExOS (MIT), Nemesis.



Aplicações podem experimentar novas abstrações diretamente sobre o hardware.

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 [4]

Referências

References I

[1] Rosemary Hattersley. *CubeSat dual-redundant flight computer*. https://magazine.raspberrypi.com/articles/cubesat-dual-redundant-flight-computer. 2021.

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

[3] 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.

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