Arquitetura de Sistemas Operacionais

LICENCIATURA EM COMPUTAÇÃO

Tipos de Arquitetura de SOs

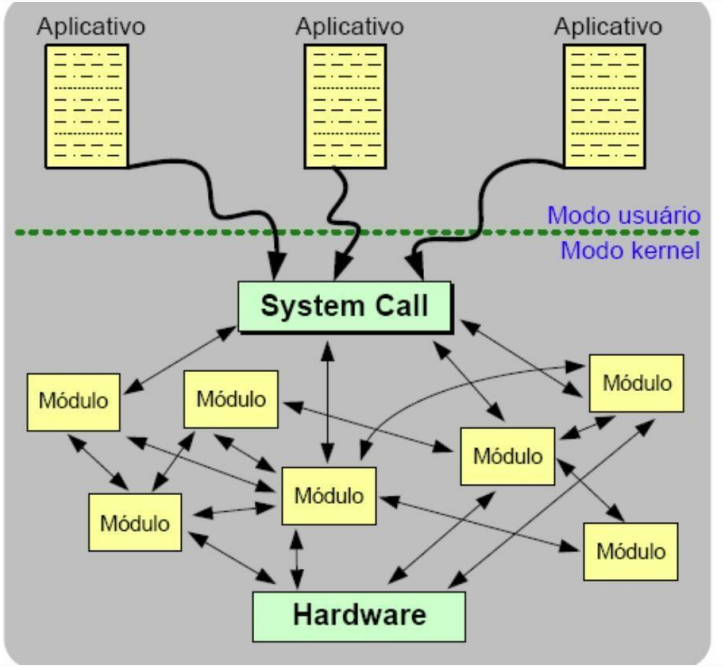
- Arquitetura Monolítica
- Arquitetura de Micronúcleo (cliente-servidor)
- Arquitetura em Camadas
- Arquitetura de Máquina Virtual

Arquitetura Monolítica

- SO é um "bloco maciço" de código que opera em **modo núcleo**, com acesso a todos os recursos do hardware e sem restrições de acesso à memória.
  - Os componentes internos do sistema operacional podem se **relacionar entre si** conforme suas necessidades.
  - **Vantagem:** desempenho → qualquer componente do núcleo pode acessar os demais componentes, áreas de memória ou mesmo dispositivos periféricos diretamente, pois não há barreiras impedindo esses acessos.
  - A interação direta entre componentes leva a **sistemas mais rápidos e compactos**, pois não há necessidade de utilizar **mecanismos específicos de comunicação** entre os componentes do núcleo.
  - Pode levar a **problemas de robustez** do sistema.
    - Como todos os componentes do SO têm acesso privilegiado ao *hardware*, caso um componente perca o controle devido a algum erro (acessando um ponteiro inválido ou uma posição inexistente em um vetor, por exemplo), esse erro pode se propagar rapidamente por todo o núcleo, levando o sistema ao **colapso** (travamento, reinicialização ou funcionamento errático).
  - **Outro problema:** processo de desenvolvimento. Como os componentes do sistema têm **acesso direto uns aos outros, podem ser fortemente interdependentes**, tornando a **manutenção e evolução do núcleo mais complexas**. Como as dependências e pontos de interação entre os componentes podem ser pouco evidentes, pequenas alterações na estrutura de dados de um componente podem ter um impacto inesperado em outros componentes, caso estes acessem aquela estrutura diretamente.

Arquitetura Monolítica

- Sistemas UNIX antigos e o MS-DOS seguiam esse modelo.
- Núcleo do Linux é monolítico, mas seu código vem sendo gradativamente estruturado e modularizado desde a versão 2.0.

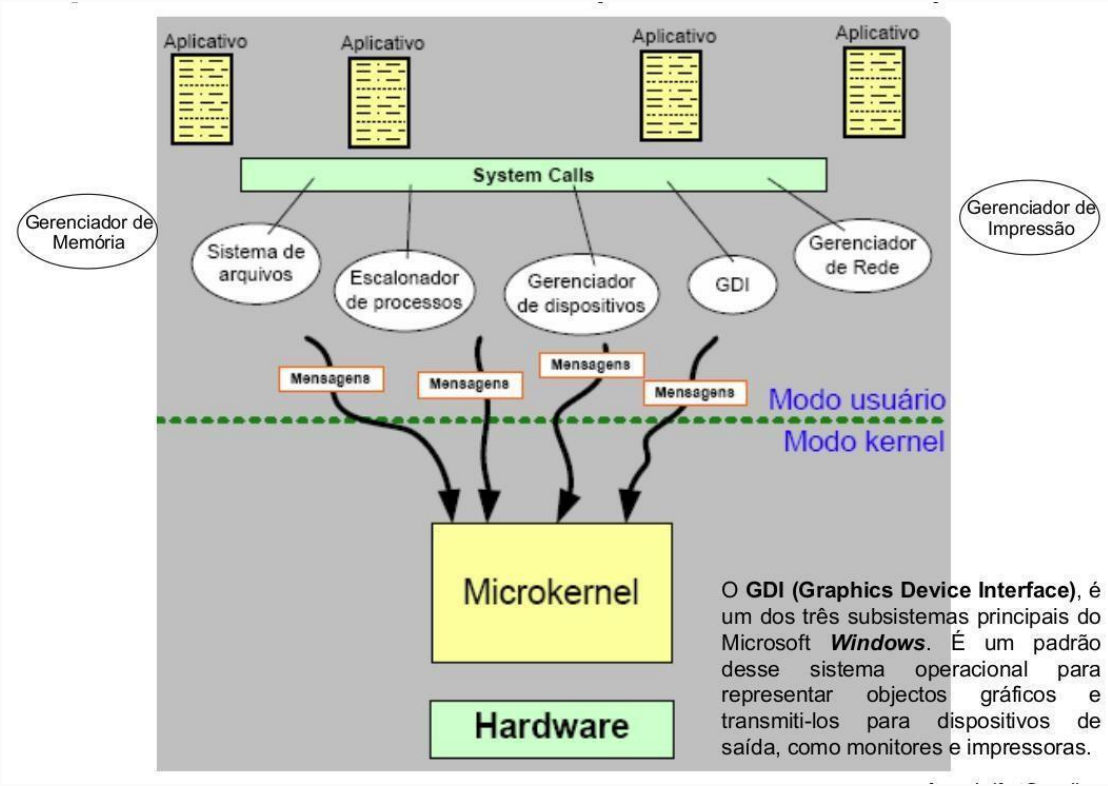


Arquitetura de Micronúcleo (cliente-servidor)

- Retirar do núcleo todo o código de alto nível, normalmente associado às abstrações de recursos, deixando no núcleo somente o **código de baixo nível** necessário para **interagir com o hardware** e criar algumas **abstrações básicas.**
- Restante do código seria transferido para **programas** separados **no espaço de usuário**, denominados **serviços**.
- **Vantagens:** maior modularidade, pois cada serviço pode ser desenvolvido de forma independente dos demais; mais flexibilidade, pois os serviços podem ser carregados e desativados conforme a necessidade; e mais **robustez**, pois caso um serviço falhe, somente ele será afetado, devido ao confinamento de memória entre os serviços.
- Um micronúcleo implementa somente a **noção de tarefa**, os **espaços de memória protegidos** para cada aplicação, a **comunicação entre tarefas** e as **operações de acesso às portas de entrada/saída** (para acessar os dispositivos).
    - Todos os aspectos de alto nível, como políticas de uso do processador e da memória, o sistema de arquivos, o controle de acesso aos recursos e até mesmos os drivers são implementados fora do núcleo, em processos que se comunicam usando o mecanismo de comunicação provido pelo núcleo.

Arquitetura de Micronúcleo (cliente-servidor)

- **Exemplo: Minix 3**



O **GDI (Graphics Device Interface)**, é um dos três subsistemas principais do Microsoft ***Windows***. É um padrão desse sistema operacional para representar objectos gráficos e transmiti-los para dispositivos de saída, como monitores e impressoras.

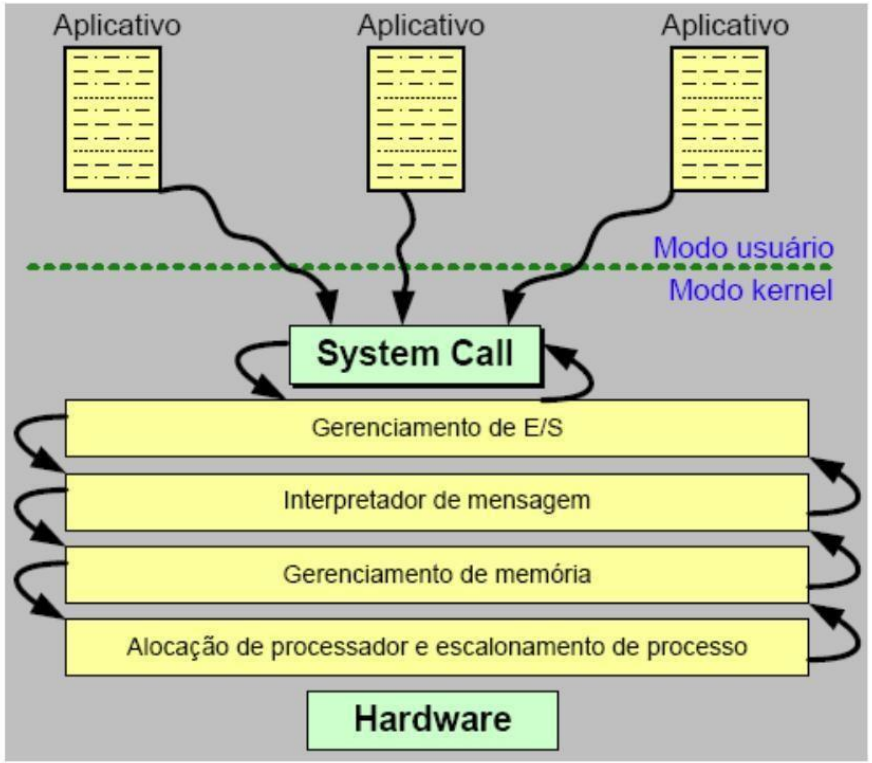
Arquitetura em Camadas

- Forma elegante de estruturar um SO:
    - Camada mais baixa realiza a interface com o ***hardware***.
    - Camadas intermediárias proveem **níveis de abstração** e gerência cada vez mais sofisticados.
    - Camada superior define a **interface do núcleo para as aplicações** (as chamadas de sistema).
    - Camadas têm níveis de privilégio decrescentes: a camada inferior tem acesso total ao hardware, enquanto a superior tem acesso bem mais restrito.

Arquitetura em Camadas

- Alguns inconvenientes **limitam a aplicação do modelo em camadas** de forma intensiva nos sistemas operacionais:
    - 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**.
    - Nem sempre a divisão de funcionalidades do sistema em camadas é óbvia, pois muitas dessas funcionalidades são interdependentes e não teriam como ser organizadas em camadas.

Arquitetura em Camadas



Arquitetura em Camadas

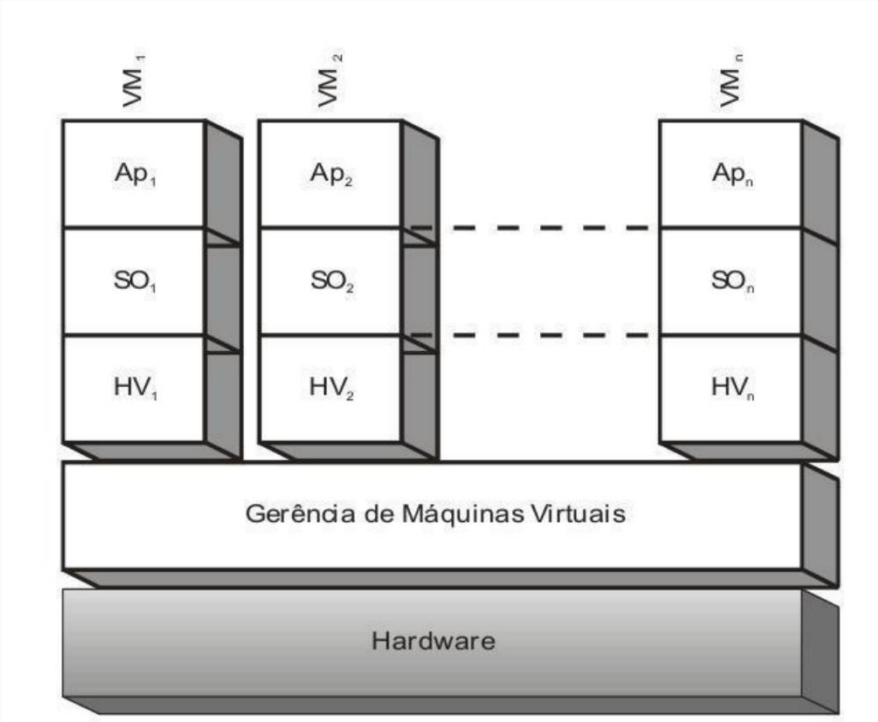
- Resumindo...
  - Organiza as funções similares em camadas.
  - Uma camada comunica-se com as camadas vizinhas.
  - Uma camada presta serviço à camada de cima.
  - Facilita a construção, uso e manutenção do SO.
  - Requisição de serviço pode atravessar várias camadas.
    - Invocando métodos adicionais de passagem de parâmetros.
    - Degradando o desempenho.

  - Exemplo:
    - Multics

Arquitetura de Máquina Virtual

- Camada intermediária entre o SO e o *hardware* – responsável por gerenciar as máquinas virtuais.
- Diversas máquinas virtuais independentes (VM).
- **C**ópia virtual completa do *hardware*.
- Pode-se ter SOs diferentes em cada máquina virtual.
- Isolamento total entre máquinas virtuais e respectivas falhas de sistemas.
- **Crítica:** O *hardware* se torna o **ponto único de falha.**

Arquitetura de Máquina Virtual



Exemplos de SOs

- Linux:
  - **Monolítico** - modular
- Windows:
  - Microkernel - **monolítico**

Atividade

- **Atividade no AVA.**