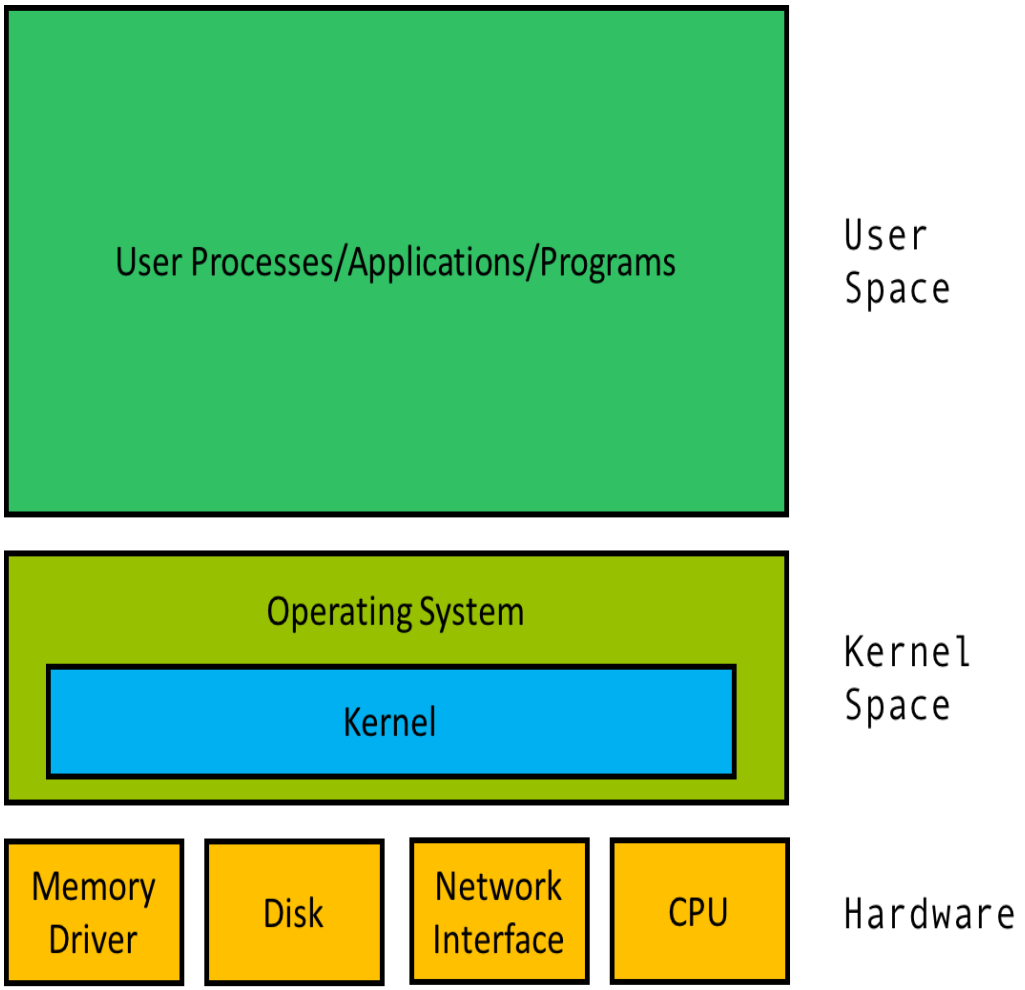
Evolução dos sistemas operacionais

Núcleo

- Principal componente
- Gerencia os recursos
- Quando o computador é ligado, executa a inicialização



Núcleo - principais funções

- Tratamento de interrupções e exceções
- Gerenciamento, sincronização, comunicação e escalonamento de processos e *threads*
- Gerenciamento da memória
- Gerenciamento dos sistemas de arquivos
- Gerenciamento dos dispositivos de entrada/saída
- Auditoria e segurança do sistema

SO

- Limita as ações dos programas
  - Ex.: quem acessa a memória e como
- Modos de acesso aos serviços do núcleo
  - Referem-se aos privilégios de execução de um programa
    - Ex.: garante que a memória não seja invadida por outro programa
- Os modos de acesso são realizados de duas formas
  - Usuário
    - Ações sem privilégios, como leitura de um arquivo
  - Kernel
    - Acesso pode ser em modo usuário ou em modo com privilégio total

- Quando um programa é executado
  - O kernel é consultado para saber se o acesso será em modo usuário ou kernel
- Os acessos aos serviços do núcleo são realizados através de chamadas ao sistema
  - Interface entre o sistema operacional e os programas dos usuários
  - Solicitação de serviços de acesso ao disco para a criação e execução de processos
  - etc.

Chamada ao sistema

- O SO recebe um comando e seus parâmetros
  - Ex.: abrir um arquivo
- Como resposta à chamada, retorna um código
  - Sucesso (se a abertura do arquivo aconteceu com sucesso)
  - Falha (se houve erro na abertura do arquivo)
  - O resultado do próprio comando (o arquivo é aberto para o usuário)

Usuários

- Conseguem identificar claramente quais são as funções do núcleo?
- É possível diferenciar das funções do software aplicativo (editores de texto, Internet Explorer, entre outros)?

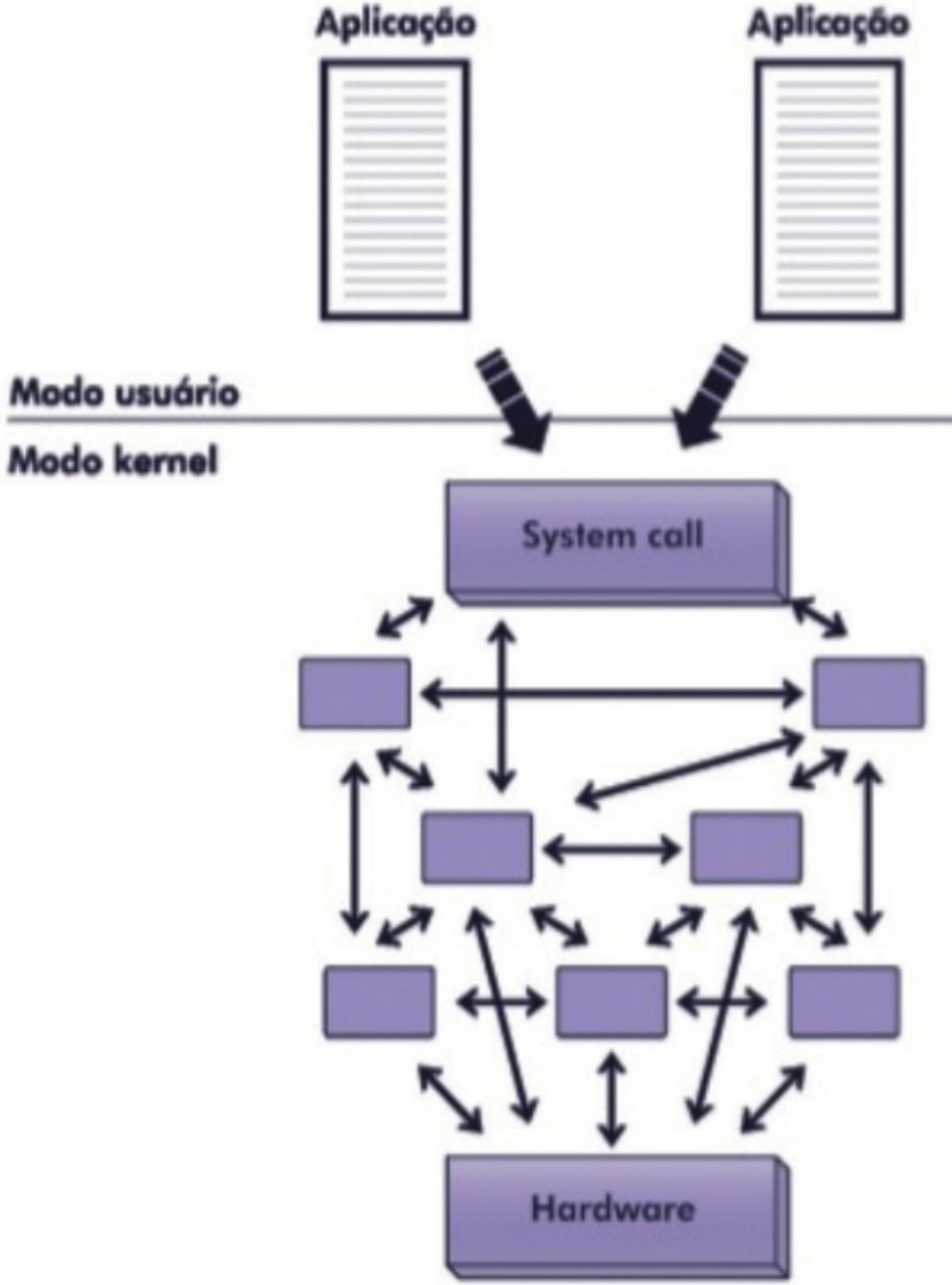
Estrutura do SO

- Maneira como o código do sistema é organizado e o inter-relacionamento entre seus diversos componentes
- Varia conforme a concepção do projeto
- Modelos das principais arquiteturas
  - Sistemas Monolíticos
  - Sistemas em Camadas
  - Máquinas Virtuais
  - Modelo Cliente-Servidor

Monolítico

Ex.: Linux (kernel)

- Conjunto de módulos compilados separadamente e agrupados em um único lugar
- Podem existir variações, porém a ideia é agrupar
- Compacto, simples, eficiente, integrado
- Problema: manutenção e extensibilidade



Sistemas em camadas

Ex.: Linux (módulos carregáveis), IBM OS/2, Multics

- Hierarquia de camadas construídas umas sobre as outras
- Cada camada oferecendo recursos às camadas superiores

Camada	Função
5	O operador
4	Programas do usuário
3	Gerenciamento de entrada/saída
2	Comunicação operador-processo
1	Gerenciamento da memória e do tambor magnético
0	Alocação de processador e multiprogramação

Máquina virtual (VM)

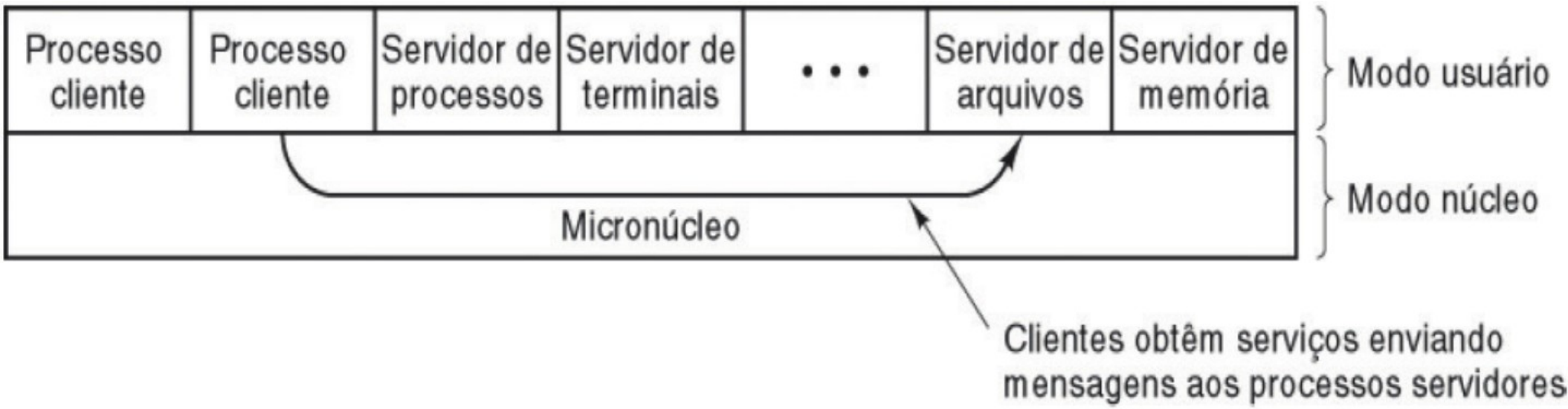
Ex.: Virtualbox, etc.

- Nível intermediário entre o SO e o hardware
- Cada VM é como se fosse um novo computador (virtual)
- Permite rodar vários SOs na mesma máquina, em VMs
  - Windows, Linux, macOS, etc., rodando lado a lado
- As VMs podem ser salvas e rodar em outro computador
- Quando acontece um problema em uma VM, basta fechá-la
- Até um certo ponto é possível configurar o hardware visível
- Existem vários tipos de VM

Modelo cliente-servidor

Ex.: Windows NT

- Implementa a maior parte das funções em modo usuário
- O kernel cuida da comunicação entre cliente e servidor
- Os servidores executam em modo usuário
- Separação entre servidores permite isolar problemas



Classificação dos SOs

- Monoprogramáveis / monotarefa
- Multiprogramáveis / multitarefa
- Multiprocessadores

Monotarefa

- Primeiros computadores
  - Pouca capacidade de hardware (comparado com os atuais)
- Um único programa por vez
- Todos os recursos são alocados ao programa
  - Mesmo que não seja utilizado
- Implementação simples do SO
- Ex.: MS-DOS

Multitarefa

- Vários programas rodando ao mesmo tempo
  - Máquinas mais poderosas
- Divisão dos recursos entre os programas
  - Reduz ociosidade
- Monousuários ou Multiusuários
- Classificação em relação à maneira de gerenciar aplicações
  - Em lote
  - Tempo compartilhado
  - Tempo real

Sistemas em lote (*batch*)

- Primeiros SOs multitarefa
  - Usavam cartões perfurados
- Sem interação do usuário com a aplicação (tela, teclado, etc.)
- Ainda usado, p. ex., em supercomputadores
- Filas de execução

Sistemas de tempo compartilhado (*time-sharing*)

- Ex.: uma única CPU rodando vários programas
  - O tempo de CPU é dividido em intervalos
- Existe a comutação entre os intervalos de tal forma que para o usuário pode parecer que os programas estão sendo executados de forma paralela
- Intervalos: fatia de tempo (time-slice)
- Cada usuário possui um ambiente de trabalho próprio
  - Impressão de que o sistema está totalmente dedicado a ele

Ex.: 2 programas rodando

- Um programa começa a executar dentro de uma fatia de tempo
- No final da fatia, o OS para o programa e passa a executar o outro, dentro da fatia deste outro
- No final desta fatia comuta para o primeiro de novo, e assim por diante
- A comutação é rápida e o tempo de resposta ao usuário é relativamente alta
  - Ficaram conhecidos como sistemas “*on-line*”

Sistemas de tempo real (*real-time*)

- O tempo é o principal parâmetro
- Um programa utiliza os recursos, até aparecer outro com maior prioridade
  - O de maior prioridade não precisa ficar esperando
- Os prazos e tempos são rigorosos
- Aplicação em processos industriais
- Roteadores de Internet residenciais
- Aplicações onde o tempo de resposta é importante

Sistemas com múltiplos processadores

- 2 ou mais CPUs interligadas
  - Processadores modernos têm mais de uma CPU
  - Cada CPU é capaz de executar programas de forma paralela
- Programas pode ser subdivididos e executados em paralelo
- Maior capacidade de processamento e de controle das CPUs
- Complexos e gastam uma parte do tempo gerenciando
- Classificação dos sistemas:
  - Fortemente acoplados
  - Fracamente acoplados

Sistemas fortemente acoplados

- Compartilham a mesma memória e dispositivos
- Controlados por um único SO
- Sistemas que usam intensivamente as CPUs
- Voltado à solução de um problema
- Classificados em
  - Simétricos
  - Assimétricos

Simétricos

- Processadores compartilham uma única memória e SO
- Técnicas de paralelismo
  - Divisão em partes
  - Rodam concorrentemente
  - Precisam que as CPUs estejam disponíveis
  - É possível distribuir o processamento entre CPUs
    - Em caso de problemas outra CPU pode ser usada
    - Problemas podem não impactar
  - Windows e Linux suportam sistemas simétricos

Assimétricos

- Processadores têm funções específicas e não compartilham o mesmo nível de acesso ao SO
- Um processador mestre controla e gerencia os demais
- A coordenação centralizada é essencial para o funcionamento
- Tarefas especializadas são distribuídas para otimizar o desempenho
- Comum em sistemas embarcados e dispositivos especializados
- Alta dependência do processador central

Sistemas fracamente acoplados

- Funcionam de forma independente
- Cada um com seu SO e gerenciamentos
- Dependem de rede para distribuir as atividades de processamento
- Programas podem depender de bibliotecas de comunicação
- Classificados em
  - Sistemas Operacionais de Rede
  - Sistemas Operacionais Distribuídos

SO de Rede

- Independentes, conectados por uma rede, programas não rodam de forma distribuída
- Na rede cada estação de trabalho ou nó possui um SO
  - Tem a capacidade de rodar os seus programas
- Permitem o compartilhamento de recursos como
  - Impressora
  - Diretórios
  - Cópia de arquivos
  - etc

SO Distribuído

- Permite rodar programa de forma distribuída
- Dividido em partes que rodam em nós
- Usuários veem como um sistema centralizado, e não como um sistema de rede
- Ex.: *clusters*
- Pode exigir bibliotecas para poder distribuir