# SISTEMAS OPERACIONAIS

## Arquitetura Sistemas Operacionais

**Andreza Leite**
andreza.leite@univasf.edu.br

# Plano de Aula

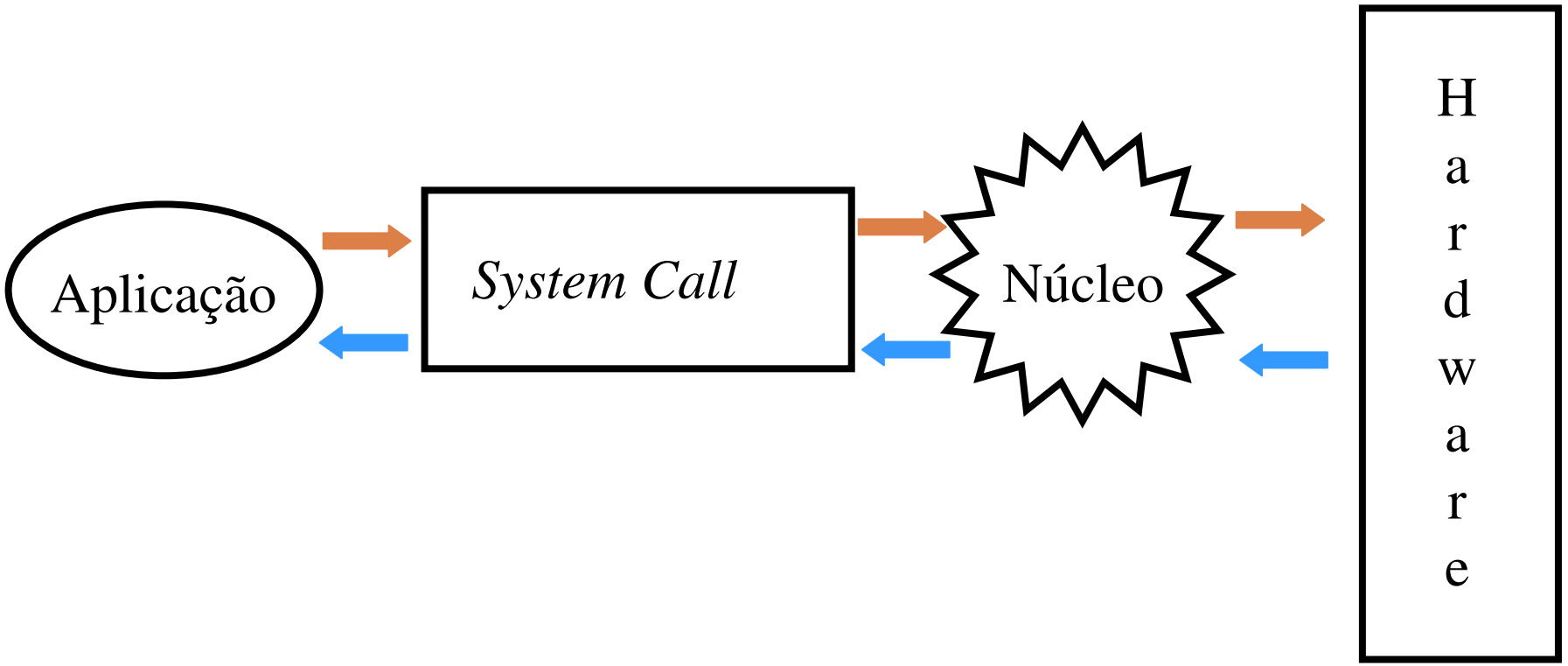
- Sistemas monolíticos
- Sistemas em camadas
- Sistemas micro-núcleo
- Modelo Cliente-Servidor
- Máquinas virtuais
- Exonúcleo

# *SYSTEM CALLS*

- Mecanismo de proteção ao núcleo do sistema e de acesso aos seus serviços.

- O usuário (ou aplicação), quando deseja solicitar algum serviço do sistema, realiza uma chamada a uma de suas rotinas (ou serviços) através da ***system calls*** (chamadas ao sistema).

# SYSTEM CALL



# Modo Kernel e Usuário

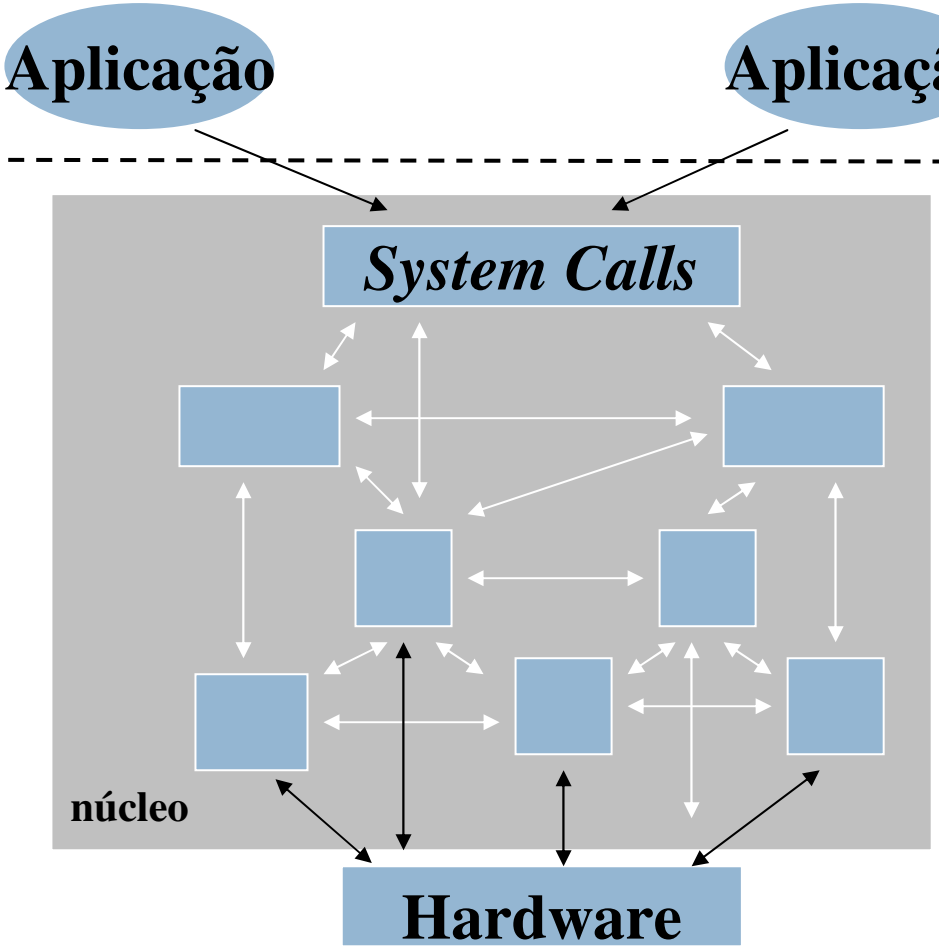
- SO roda em **Modo kernel, supervisor ou núcleo** → protege o *hardware* da ação direta do usuário.

- Os demais programas rodam em **modo usuário** e fazem chamadas ao kernel para terem acesso aos dispositivos.

# Sistema Monolítico

- Nesta abordagem o SO inteiro é executado como um único programa no modo núcleo.
- A organização mais comum é aquela que estrutura o sistema como um conjunto de rotinas que podem interagir livremente umas com as outras.
- Pode ser comparada com uma aplicação formada por vários procedimentos que são compilados separadamente e depois linkados, formando um grande e único programa executável.
  - Grande desempenho
  - Uma falha pode paralisar todo o núcleo. O sistema pode parar por causa de um erro.
  - As interfaces e níveis de funcionalidade não são bem separados nem estão unificados. O excesso de liberdade torna o sistema vulnerável
  - Ex: Linux e FreeBSD

# Sistema Monolítico




1. Um programa principal que invoca a rotina do serviço requerido.
2. Um conjunto de rotinas de serviço que executam as chamadas de sistema.
3. Um conjunto de rotinas de utilidade que auxiliam as rotinas de serviço.

# Sistema em Camadas

- Divide o sistema operacional em sistemas sobrepostos. Cada módulo oferece um conjunto de funções que pode ser usado por outros módulos.

- A vantagem da estruturação em camadas é isolar o sistema operacional, facilitando sua alteração e depuração, além de criar uma hierarquia de níveis de modos, protegendo as camadas mais internas.

# Sistema em Camadas

- 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 dividir as funcionalidades de um núcleo de sistema operacional em camadas horizontais de abstração crescente, pois essas funcionalidades são inter-dependentes, embora tratem muitas vezes de recursos distintos.

# Sistema em Camadas

| Camada | Função |
|---|---|
| 5 | O operador |
| 4 | Programas de usuário |
| 3 | Gerenciamento de entrada/saída |
| 2 | Comunicação operador–processo |
| 1 | Memória e gerenciamento de tambor |
| 0 | Alocação do processador e multiprogramação |

**Tabela 1.3** Estrutura do sistema operacional THE.

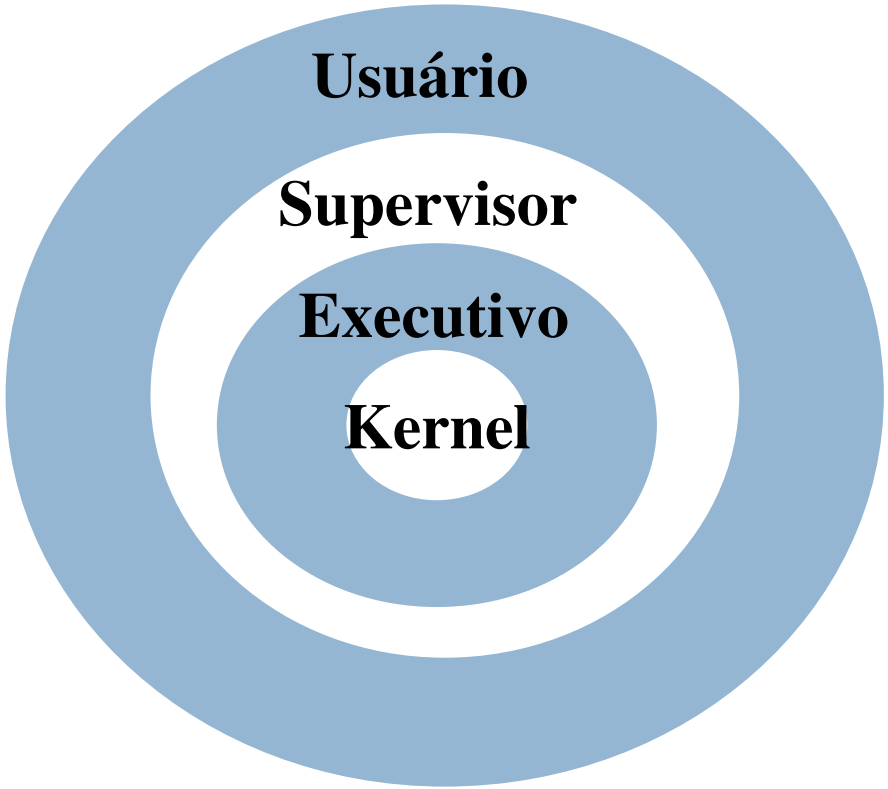
# Sistema em Camadas

- O sistema THE era um sistema de lote simples para um computador holândes, o Electrologica X8.
  - **Camada 0** – lidava com alocação do processador , alternando entre processos quando ocorriam interrupções ou quando os temporizadores expiravam. Em outras palavras, a camada 0 (zero) proporcionava a multiprogramação básica da CPU.
  - **Camada 1** – fazia o gerenciamento da memória. Ela alocava espaço para os processos da memória principal e em um tambor (Antigo meio magnético de armazenamento de dados) utilizado para armazenar partes do processo (páginas) para os quais não havia lugar na memória principal.
  - **Camada 2** – fazia a comunicação entre o console do operador e cada processo.
  - **Camada 3** – gerenciava dispositivos de entrada e saída.
  - Na **camada 4** – localizavam-se os programas de usuários. Eles não tinham de se preocupar com o gerenciamento de processo, memoria, console ou E/S.
  - Na **camada 5** – estava localizado o processo operador do sistema.

# Sistema em Camadas (anéis)

- Anéis mais internos são mais privilegiados que os externos;
- Procedimentos de anéis externos executavam chamadas de sistema para utilizar os serviços dos anéis internos;
- Proteção dos segmentos de memória.

# Sistema em Camadas




**Sistema Multics VMS**

+No sistema MULTICS VMS as camadas inferiores são as mais privilegiadas.

# Sistemas micro-núcleo (microkernel)

- Uma tendência dos sistemas operacionais é tornar o núcleo menor e mais simples possível e para implementar esta idéia o sistema é dividido em processos.
- Desta forma, sempre que uma aplicação deseja algum serviço ela solicita ao processo responsável, assim, a aplicação que solicita um serviço é chamada de cliente e o processo que responde a solicitação é chamado de servidor.

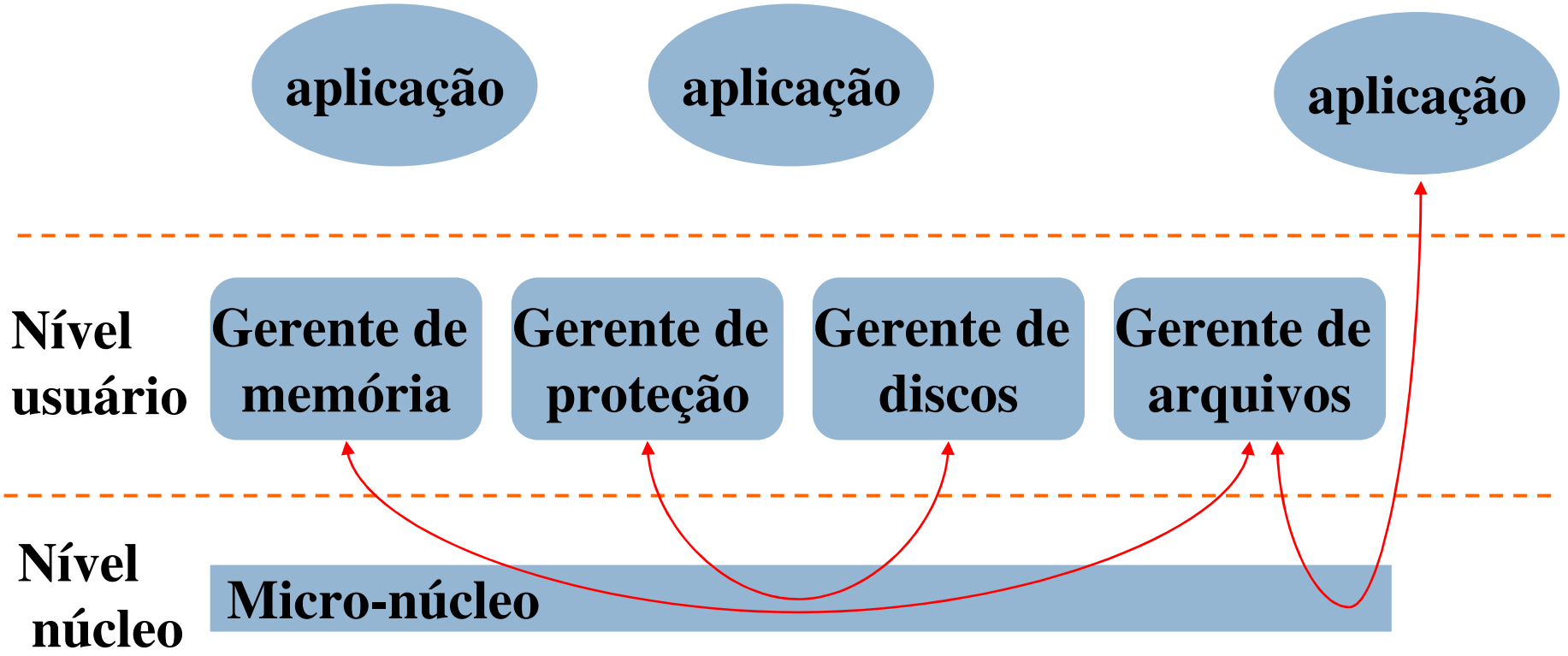
# Sistema micro-núcleo

- A utilização deste modelo permite que os servidores executem em modo usuário.
- Apenas o núcleo do sistema, responsável pela comunicação entre clientes e servidores, executa no modo kernel.
- O sistema operacional passa a ser de mais fácil manutenção.
- Não importa se o serviço esta sendo processado em um único processador, com múltiplos processadores (fortemente acoplado) ou em sistema distribuído (fracamente acoplado).

# Sistema micro-núcleo

- Um ambiente distribuído permite que um cliente solicite um serviço e a resposta seja processada remotamente.
- Sua implementação é difícil e mais usualmente é implantado uma combinação do modelo de camadas com o cliente-servidor.
- O núcleo do sistema passa a incorporar o escalonamento e gerência de memória além das funções de *device drivers*.

# Micro-núcleo – Visão Geral



# Micro-núcleo

- A idéia básica por trás do projeto do micronúcleo é alcançar alta confiabilidade por meio da divisão do sistema operacional em módulos pequenos, bem definidos, e apenas um desses módulos – o micronúcleo – é executado no modo núcleo e o restante é executado como processos de usuário.

- Quando há execução de cada driver de dispositivo e cada sistema de arquivo como processo separado, um erro em um deles pode quebrar aquele componente, mas não pode quebrar o sistema inteiro.

# Micro-núcleo

- Micronúcleos são comuns em aplicações de tempo real, industriais, avionica e militares, que são cruciais e tem requisitos de confiabilidade muito altos.
- Ex: Integrity, K42, PikeOS, QNX, Symbiam e MINIX3.

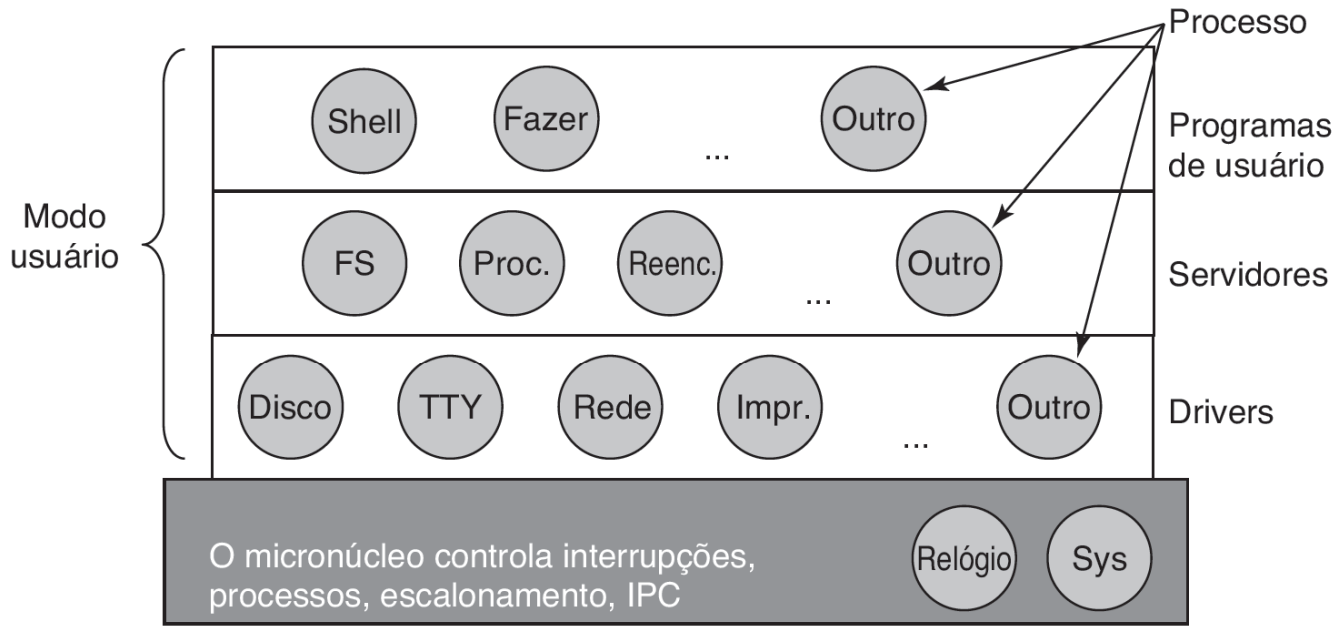
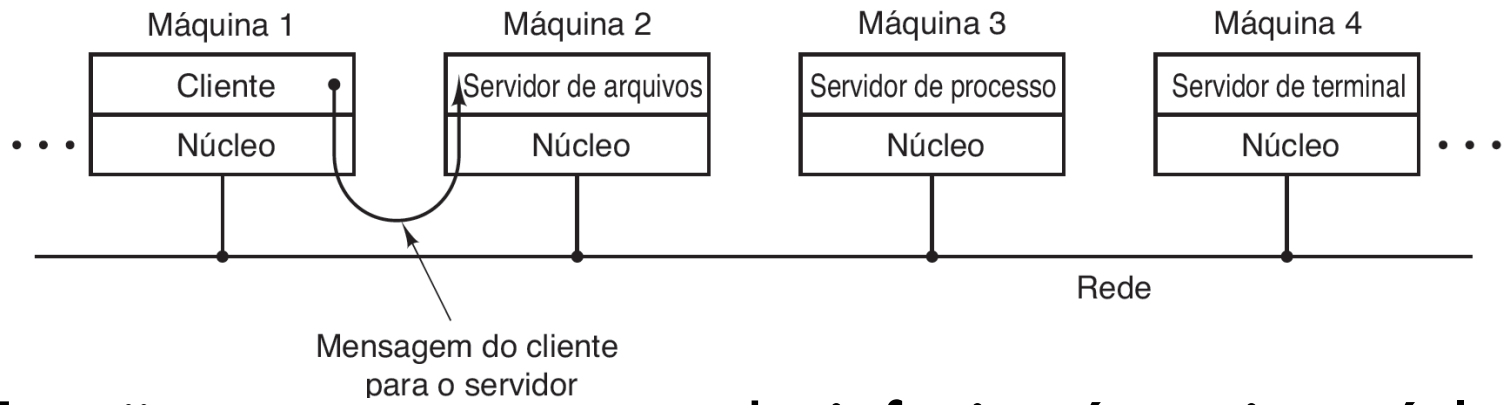



**Figura 1.23** Estrutura do sistema MINIX 3.

# Modelo Cliente Servidor

- Uma variação da idéia do micronúcleo é distinguir entre duas classes de processos, os **servidores**, que prestam serviço, e os **clientes**, que usam serviços.

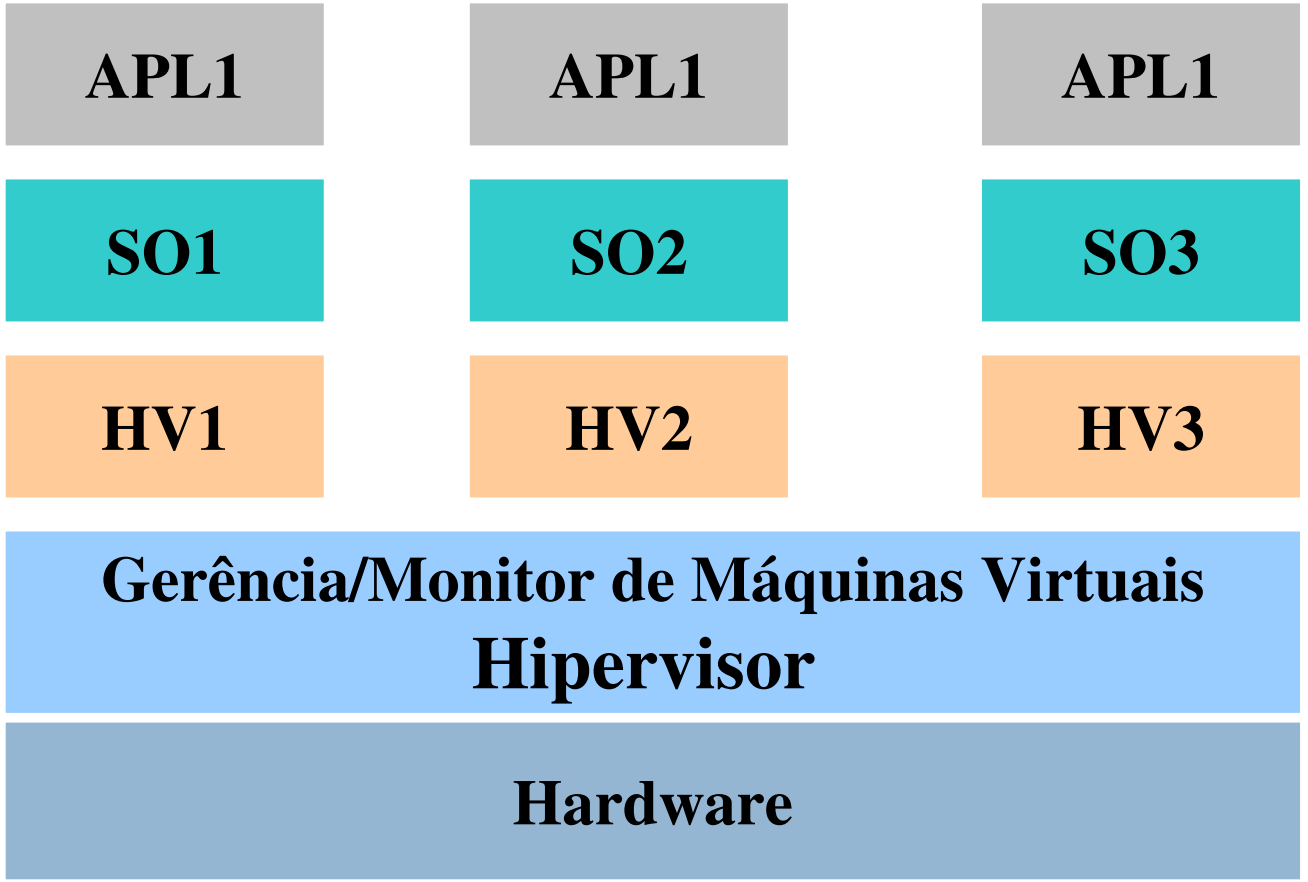



- Freqüentemente a camada inferior é o micronúcleo mas não obrigatoriamente. A essência é a presença de processos clientes e servidores.

# Máquina Virtual

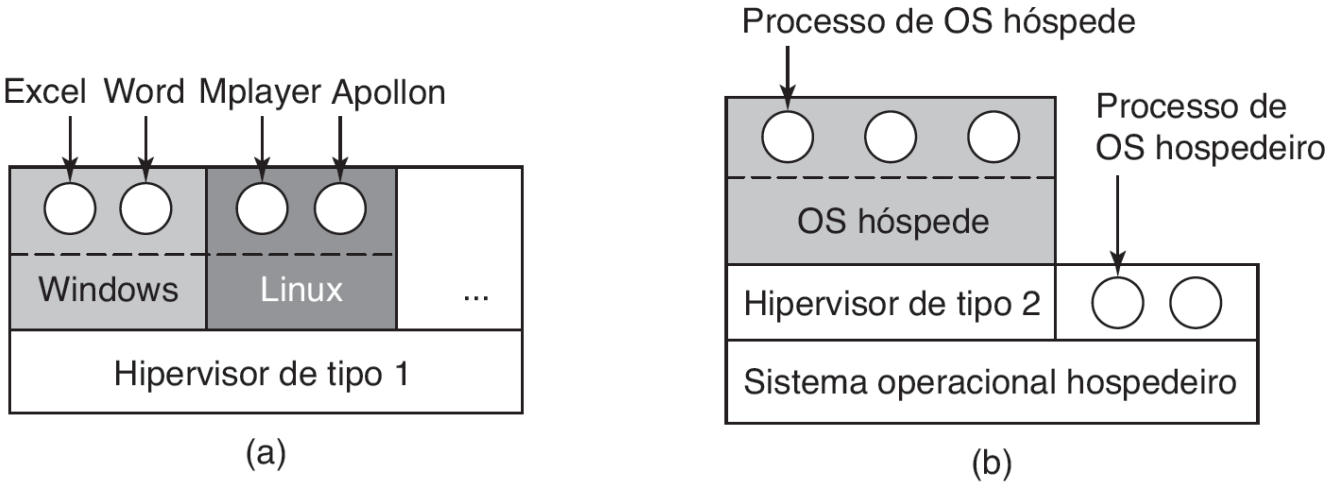
- Máquinas virtuais não são máquinas estendidas com arquivos e outras características convenientes.
- São cópias exatas do hardware, inclusive com modos núcleo/usuário, E/S, interrupções e tudo o que uma máquina real tem.
- Cada VM pode executar qualquer SO capaz de ser executado diretamente sobre o hardware.
- Diferentes VMs podem executar direrentes Sos.

# Máquina Virtual



# Máquina Virtual

- Hipervisor do tipo 1(a) são executados diretamente no hardware.
  - Eucalyptus
- Hipervisor do tipo 2(b) são executados como aplicativos na camada superior do SO
  - Ex: VMware, VirtualBox



# Exonúcleo

- Em vez de clonar a máquina real, como nas VMs, outra estratégia é dividi-la ou dar a cada usuário um subconjunto de recursos.
  - Assim uma VM pode ter os blocos de 0 a 1023 do disco e outra de 1024 a 2047 e assim por diante.
- Na camada inferior há o programa **exonúcleo** responsável por alocar recursos às VMs e verificar as tentativas de uso para assegurar que uma máquina não esteja utilizando os recursos de outra.
- Cada VM pode utilizar seu próprio SO mas somente com os recursos que pediu e que foram alocados.
- Este tem vantagem de separar, com menor custo, a multiprogramação (no exonúcleo) do código do SO do usuário, visto que o exonúcleo mantém as VMs umas fora do alcance das outras.