Computação Distribuída

Odorico Machado Mendizabal

Universidade Federal de Santa Catarina – UFSC
Departamento de Informática e Estatística – INE

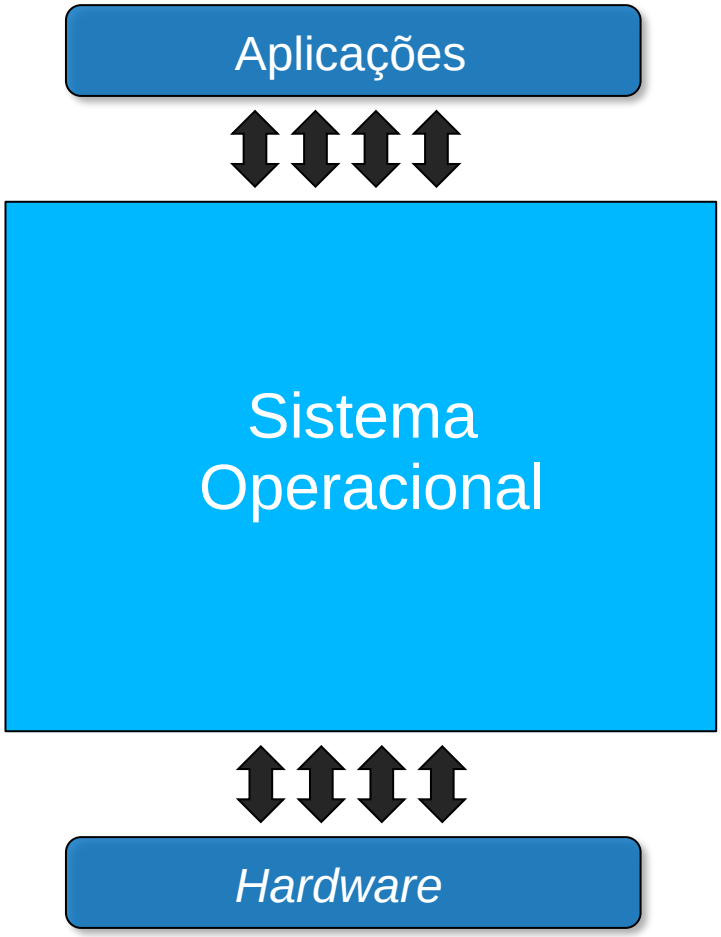
Revisão sobre Sistemas Operacionais

Revisão de Conceitos do Sistema Operacional

- Interface entre Aplicações e *Hardware* em um sistema de computação
  - Sistemas Operacionais para *desktop*, servidores, dispositivos móveis, sistemas embarcados em geral

- Gerência de recursos locais
  - Compartilhamento de recursos de forma eficiente, organizada e segura

- Evitar retrabalho e redundância de código
  - Bibliotecas, ligação dinâmica, etc.

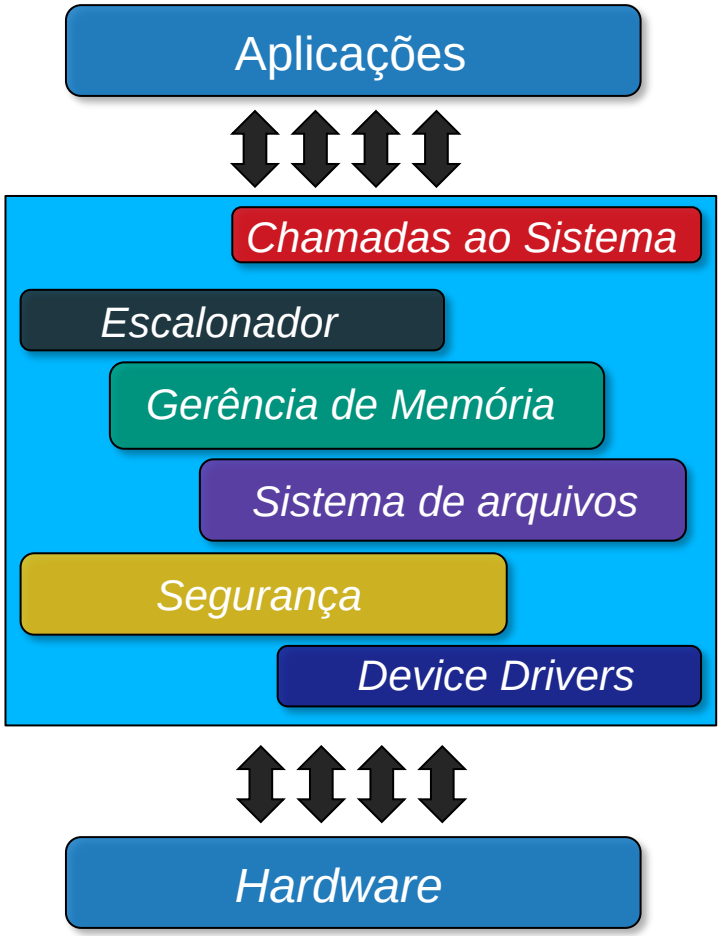


Revisão de Conceitos do Sistema Operacional

Módulos de serviços especializados para o gerenciamento de recursos:

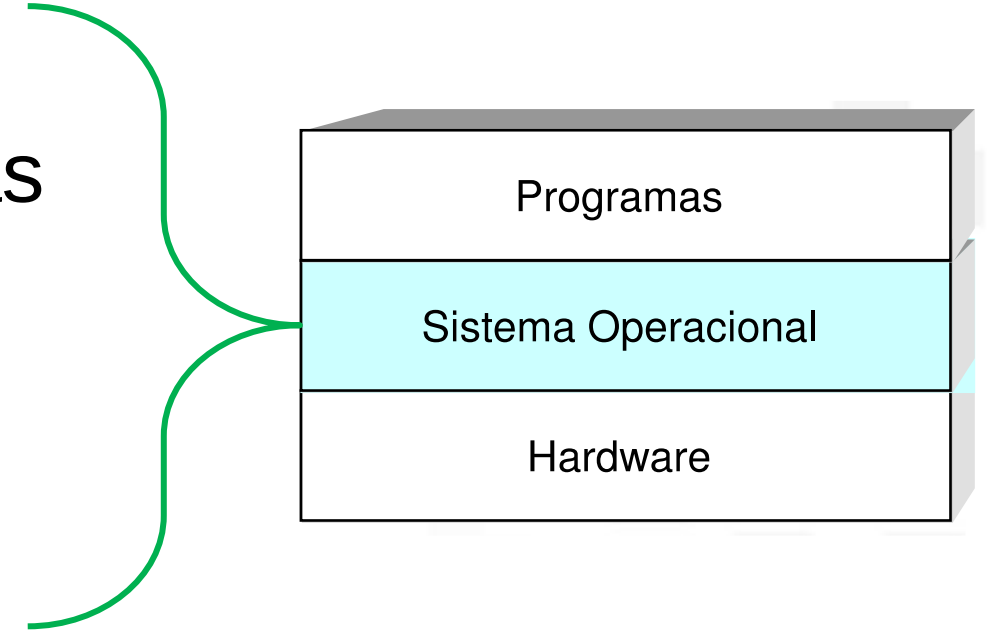
- Gerenciamento eficiente dos recursos (memória, CPU, ...):
  - políticas de escalonamento
  - paginação/segmentação
  - memória virtual

- Organização do espaço de armazenamento físico através da implementação de um sistema de arquivos



Estrutura dos Sistemas Operacionais

- Sistema Monolítico
- Sistema de camadas
- Micronúcleo
- Cliente-servidor
- Máquinas Virtuais

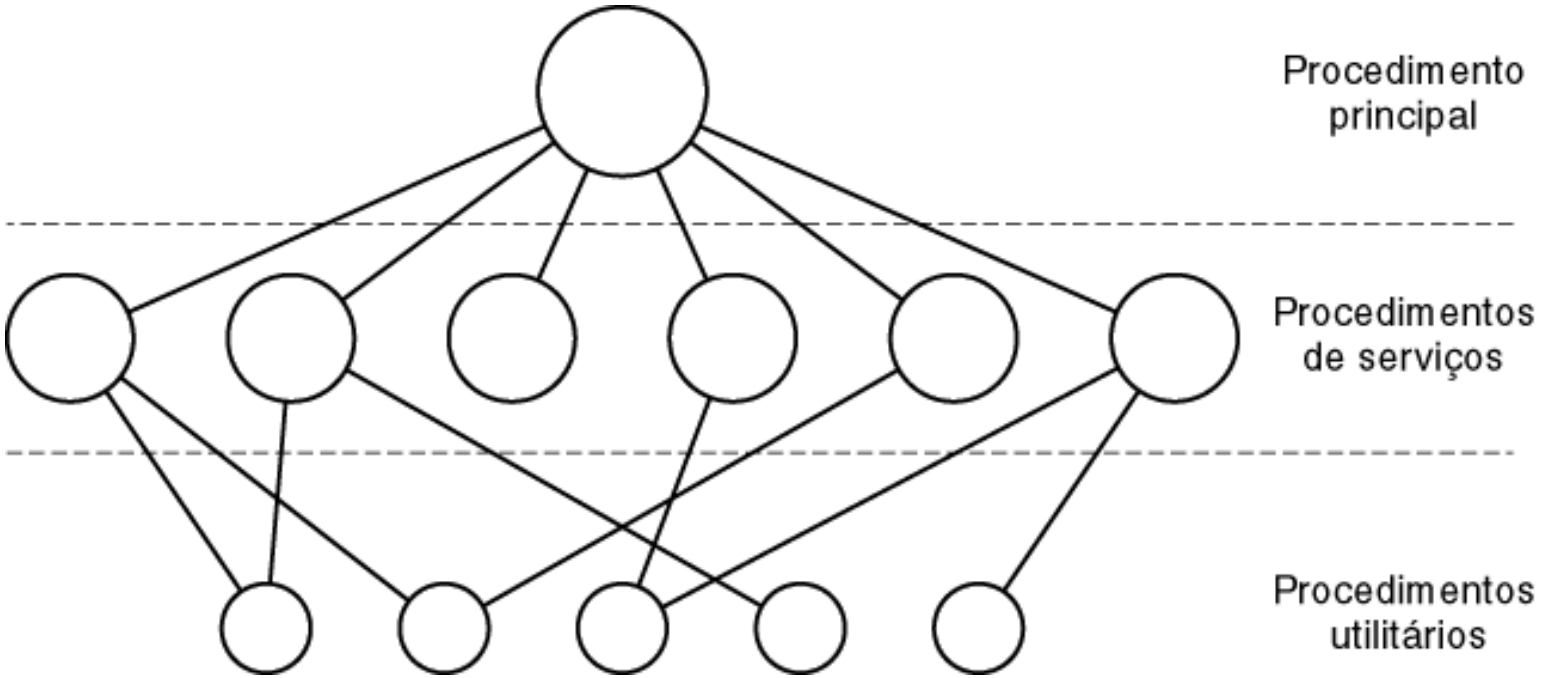


Sistema Monolítico

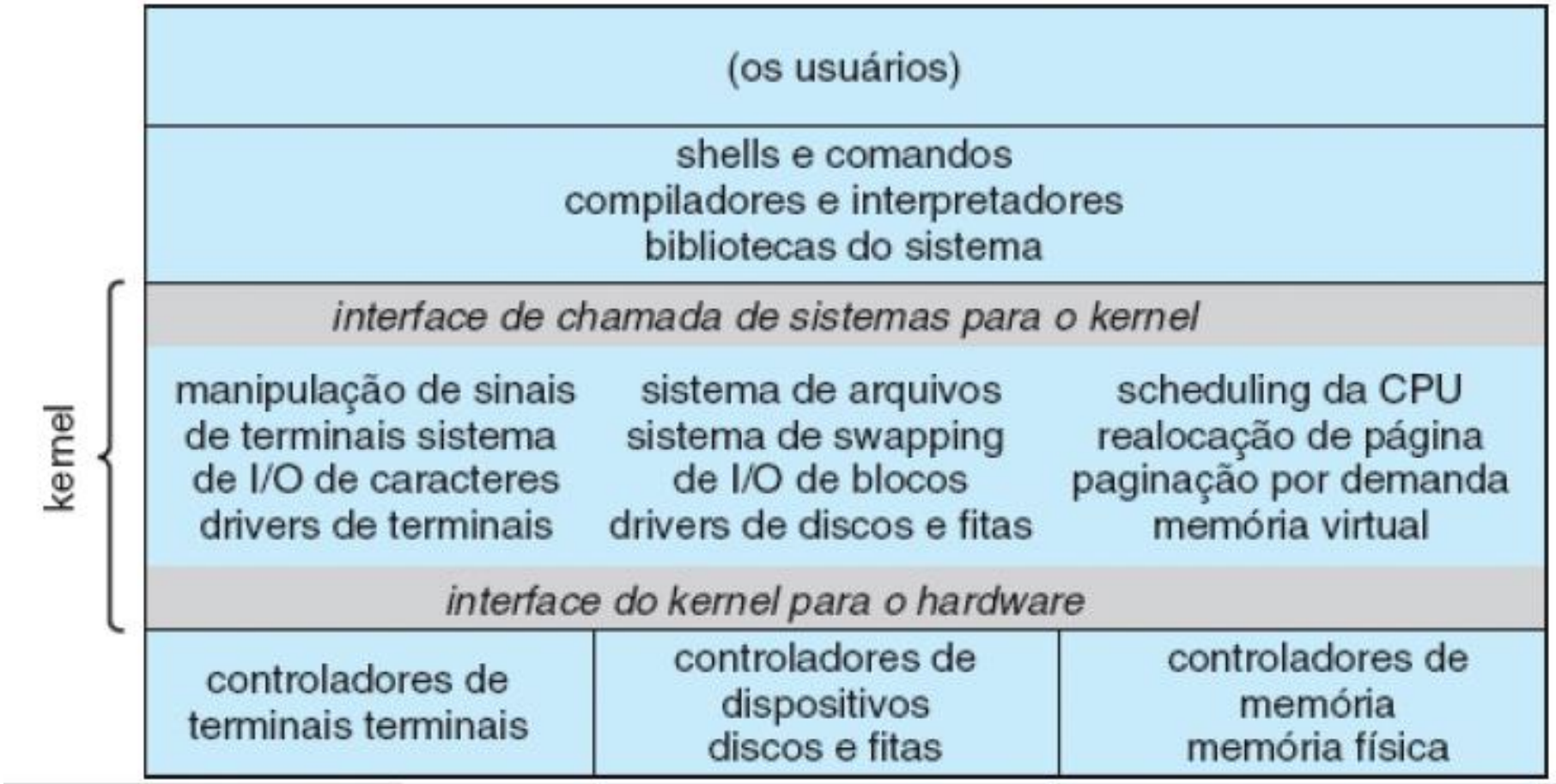
- Ainda é o mais comum na construção de SOs
  - ex. Unix, Linux
- O SO inteiro é executado como um único programa no modo núcleo
- SO é escrito como uma coleção de rotinas ligadas a um grande programa executável único
- Cada rotina tem uma interface bem definida e é livre para chamar qualquer outra
- Difícil de gerenciar a execução das rotinas

Sistema Monolítico

Modelo simples de estruturação de um sistema monolítico



Sistema Monolítico – Estrutura do Unix



Sistema de Camadas

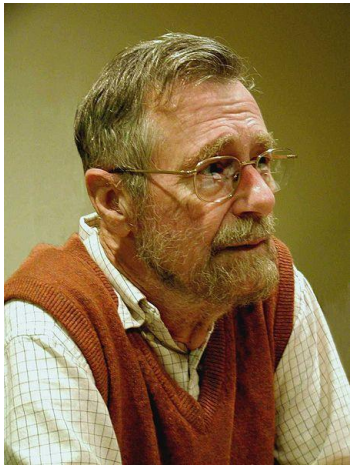
- Hierarquia de camadas
  - cada camada construída sobre a camada inferior
  - A comunicação é feita somente entre camadas adjacentes
- Primeiro foi o THE (*Technische Hogeschool Eindhoven*) – Implementado por Dijkstra

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

Estrutura do sistema operacional THE

Sistema de Camadas

- Primeiro foi o THE (*Technische Hogeschool Eindhoven*) – Implementado por Dijkstra



- **1968** *E. W. Dijkstra*: Cooperando processos sequenciais

Noções como: seções críticas e processos com execução cíclica
Critérios de correção:
exclusão mútua, justiça e independência de velocidade

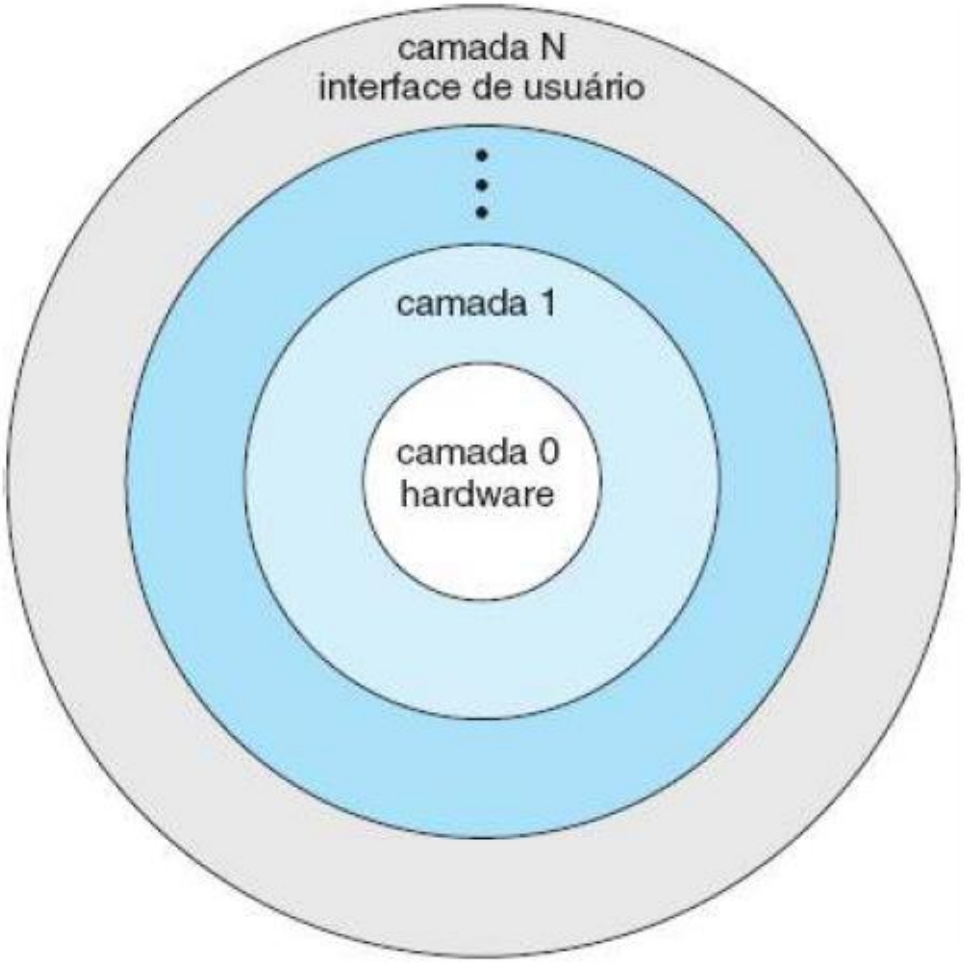
1972

- **1971** *E. W. Dijkstra*: Ordem hierárquica de processos sequenciais

Prova de correção de aspectos de concorrência relacionados ao Sistema Operacional THE
Proposta de encapsulamento de dados para controle à região crítica
Implementação de *buffer* limitado com semáforos
Resolução do problema do Jantar dos filósofos

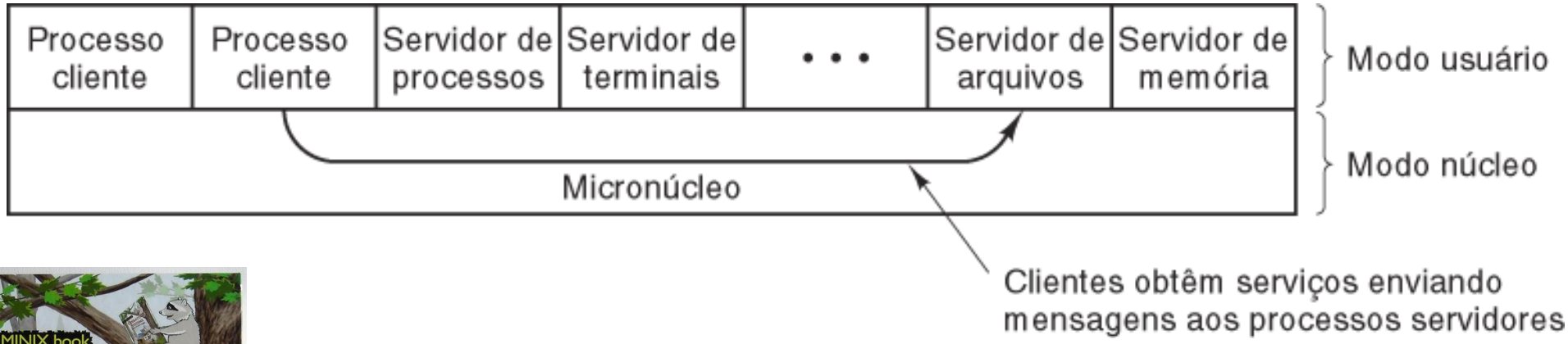
Sistema de Camadas

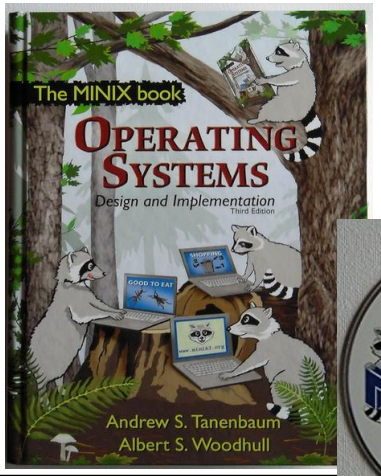
- Outro exemplo – MULTICS
  - Anéis concêntricos – sendo os anéis interiores com maior prioridade



Sistema de Micronúcleo

- Comuns em aplicações de tempo-real
- Ex. L4, Symbiam, Minix3 (proposto por Tanenbaum): www.minix3.org





DICA DE LEITURA:
- Operating Systems, Design and Implementation "The MINIX book"

Sistema de Micronúcleo

Você Sabia?

CURIOSIDADE:
- Monolítico vs. Micronúcleo

- Debate entre Linus Torvalds e Andrew Tanenbaum, em 1992.
    - https://www.oreilly.com/openbook/opensources/book/appa.html
- Tanenbaum projetou o Minix3 (micronúcleo)
- Linus Torvalds propôs o Linux (monolítico)

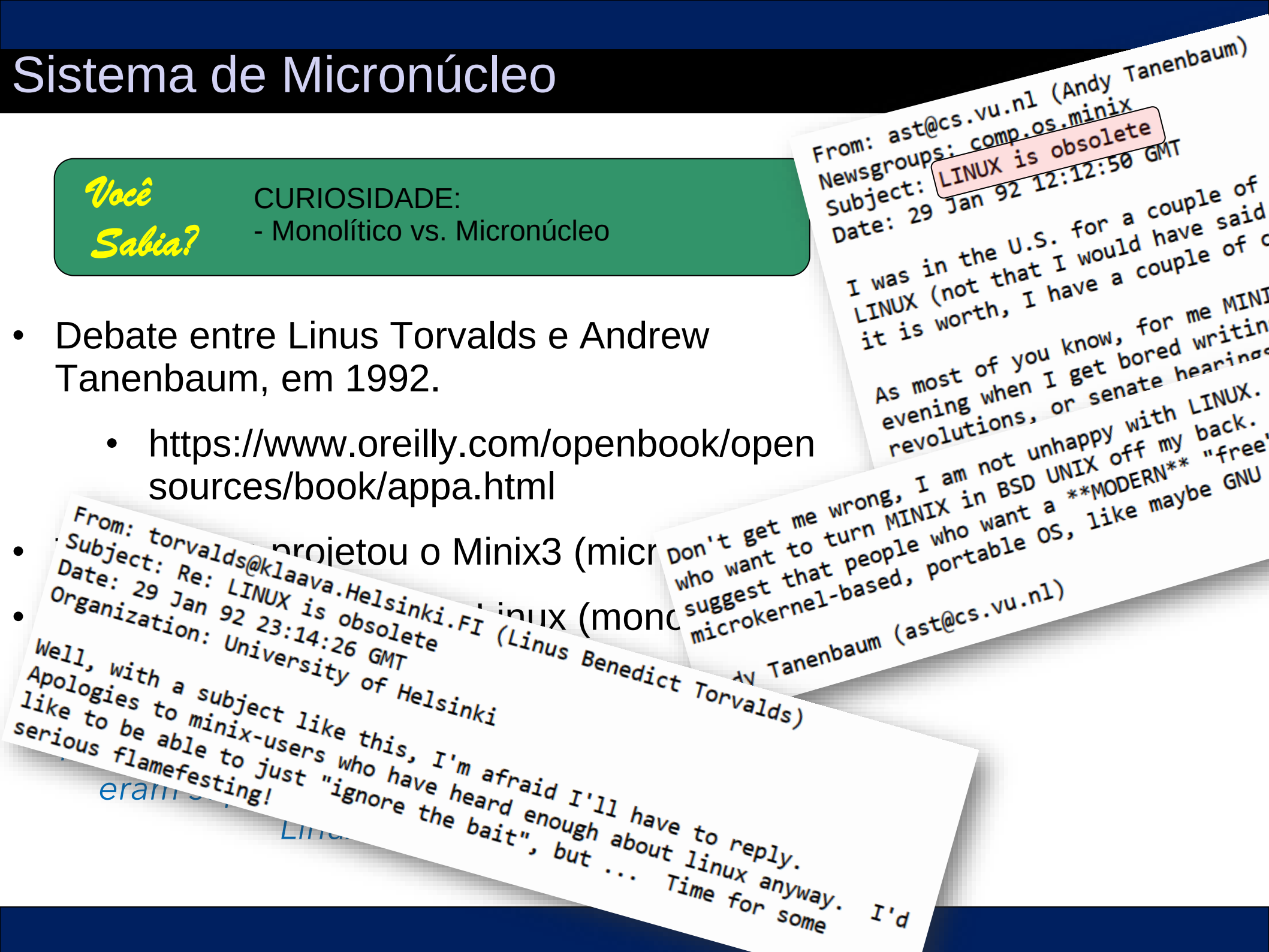
Tanenbaum argumentava que SOs micronúcleos eram superiores aos monolíticos, por isso o Linux seria obsoleto.

Sistema de Micronúcleo

Você Sabia?

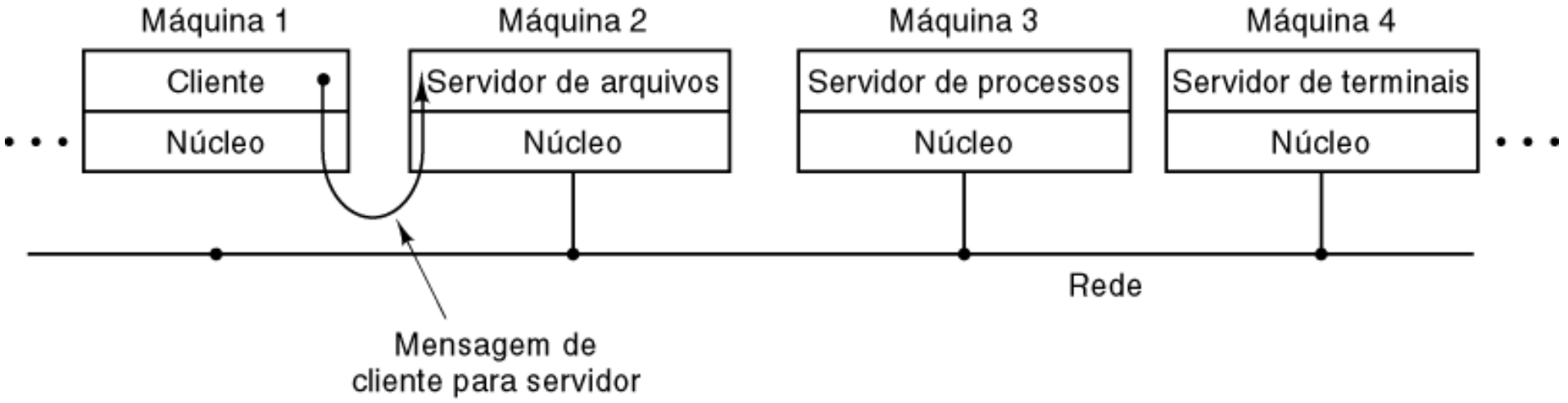
CURIOSIDADE:
- Monolítico vs. Micronúcleo

- Debate entre Linus Torvalds e Andrew Tanenbaum, em 1992.
    - https://www.oreilly.com/openbook/opensources/book/appa.html
- projetou o Minix3 (micr
- inux (monc



Sistema Cliente – Servidor

- Semelhante à estrutura micronúcleo
- Executa em um ambiente distribuído
  - O sistema operacional executa em várias máquinas
  - Comunicação por troca de mensagens através de uma rede de interconexão



Sistema Cliente – Servidor – SO Distribuídos

- Valor histórico e acadêmico
  - Amoeba (1980, Tanenbaum) – Oferece transparência sobre a distribuição

Sprite (1984, Berkeley) – Executa uma mesma imagem do sistema em todas as máquinas na rede

Plan 9 (1990, Bell Labs) – Estende princípios do UNIX para sistemas distribuídos, usa um Sistema de arquivos unificado

- SO distribuídos modernos

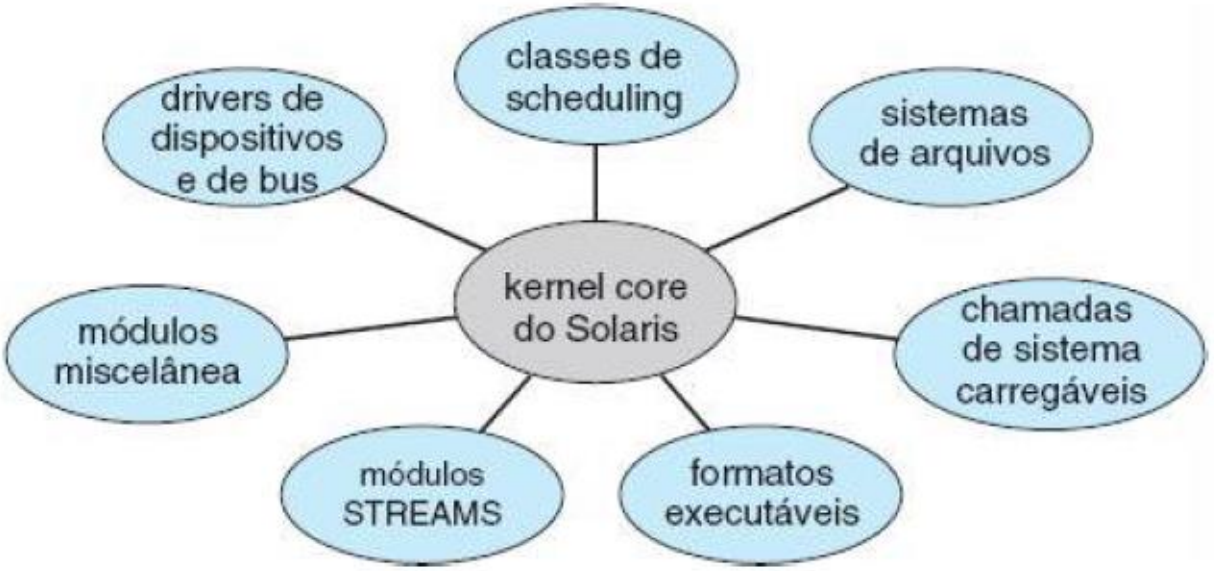
Barrelfish (2020, ETH Zurique) – enfoque em arquiteturas multi- e many-core (projeto foi descontinuado)

SONiC (2016, Microsoft) – Cria uma API abstrata desacoplando SW de rede do HW subjacente. Suporta protocolos BGP, RDMA, QoS. Usado por provedores de nuvem

  - SO para SoCs: Arteris, Navix (Kalray), etc..

Módulos Carregáveis

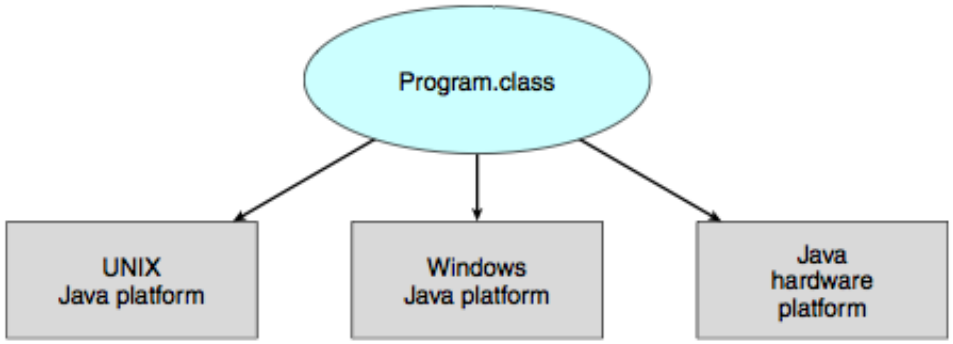
- Atualização de serviços é frequentemente necessária
  - Desafio: como integrar ao SO um novo serviço (ex. acesso à um novo dispositivo de E/S sem atualizar o SO)?
- Uso de módulos carregáveis
  - Abordagem é semelhante à de micronúcleos

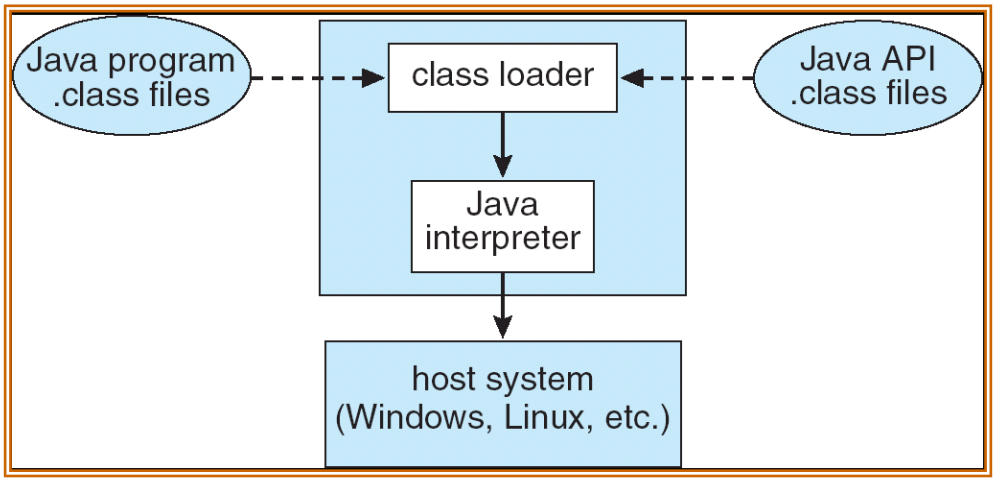


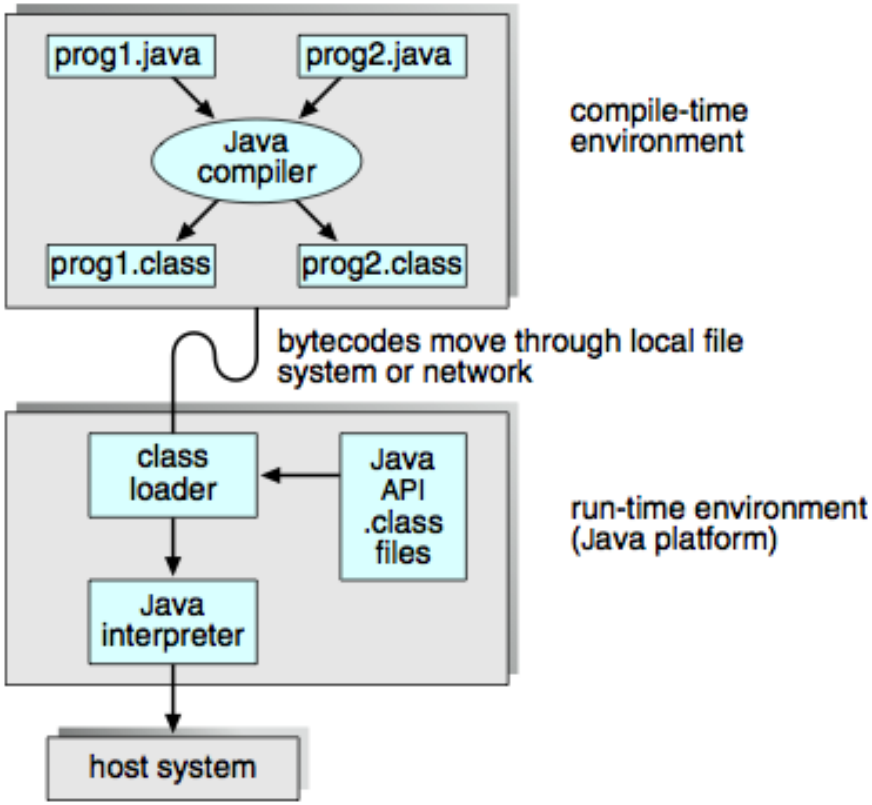
Máquinas Virtuais

- Software de emulação de uma máquina abstrata
  - Oferece aos programas a ilusão de que eles controlam uma máquina
    - Uma máquina física pode hospedar internamente diferentes ambientes virtuais, cada um simulando uma máquina com configurações distintas de CPU, memória, E/S, etc.
  - Fornece uma visão do hardware personalizado (exatamente como o usuário quer)
- Dois tipos de Máquinas virtuais
  - VM para processos: permitem a execução de um programa sobre a máquina virtual (e.x.: JVM, .Net Framework – CLR);
  - VM para sistemas: permitem a execução de um SO completo e suas aplicações (e.x. VMWare, Virtualbox, Xen, KVM, etc.)

Máquinas Virtuais – Máquina Virtual Java (JVM)

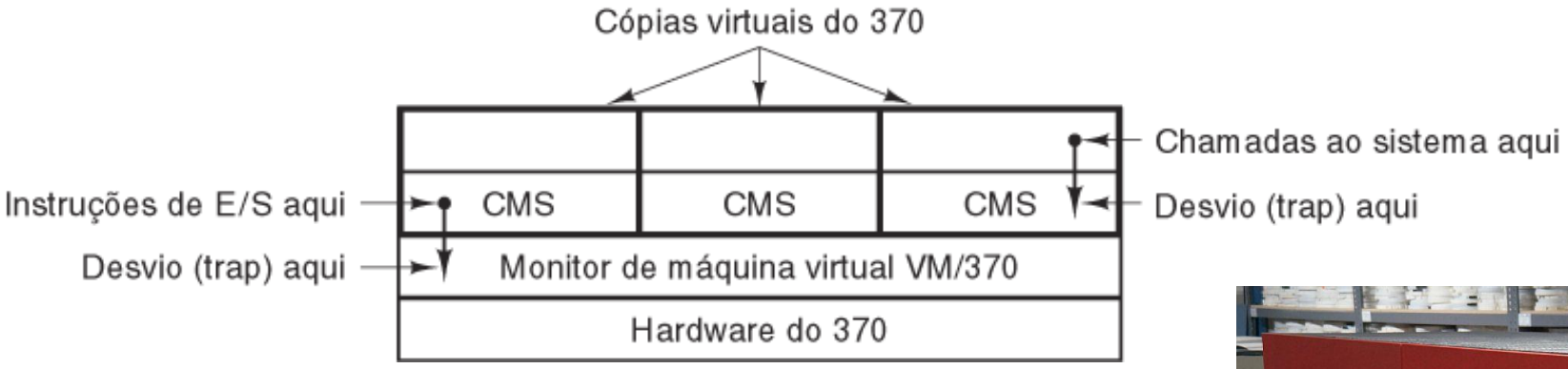






Máquinas Virtuais

- VM/370 (IBM System 370, 1972)
- Divisão entre Multiprogramação e Abstração de Hardware

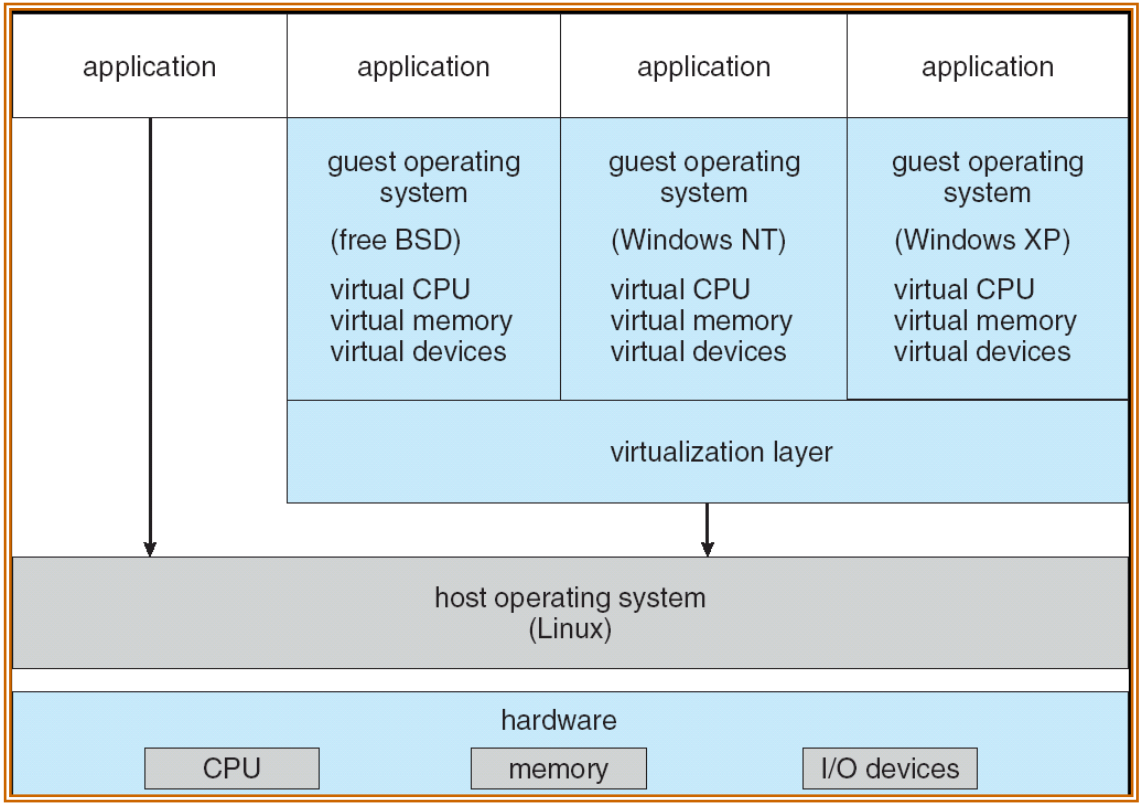


Estrutura do VM/370 com o CMS

(*Conversational Monitor System*)

Máquinas Virtuais

- Máquinas virtuais (VMware, Xen, KVM, etc.)
- Sistemas operacionais são virtualizados e executam como hóspedes em uma máquina hospedeira



Máquinas Virtuais – Vantagens

- Simplicidade de programação
  - Cada processo “pensa” que ele tem acesso à toda memória / Tempo de CPU
  - Cada processo pensa que ele detém todos os dispositivos
  - Diferentes dispositivos aparentam ter o mesmo alto nível de interface
- Isolamento de falhas
  - Processos não podem impactar outros processos diretamente
  - *Bugs* não podem afetar a máquina como um todo
- Proteção e portabilidade
  - Interface (ex. Java) é segura e estável em diversas plataformas

Considerações

- Sistemas Operacionais / Sistemas Distribuídos
  - Visam gerenciamento de recursos eficiente
  - Visam facilitar o uso e programação de aplicações
  - Devem prover segurança, disponibilidade, desempenho, etc.
  - Grande variedade de arquiteturas de SW e decisões de projeto possíveis
- Sistemas distribuídos também fazem uso de:
  - Mecanismos de sincronização entre tarefas (processos e *threads*)
  - Virtualização
  - Políticas de segurança

Referências

Parte destes slides são baseadas em material de aula dos livros:

- *OLIVEIRA, Rômulo Silva de; CARISSIMI, Alexandre da Silva.; TOSCANI, Simão Sirineo. Sistemas operacionais. 4. ed. Porto Alegre: Bookman, 2010. xii, 374p. (Livros didáticos, n.11) ISBN 9788577805211*

- *SILBERSCHATZ, Abraham.; GAGME, Greg; GALVIN, Peter B. Sistemas operacionais com Java. Rio de Janeiro: Elsevier, 2008. 673 p. ISBN 9788535224061*

- *TANENBAUM, Andrew S. Sistemas operacionais modernos. 3. ed. Rio de Janeiro (RJ): Prentice-Hall do Brasil, 2010. xiii, 653p. ISBN 9788576052371*