Sistemas Operacionais

Projeto e Estruturas de Sistemas Operacionais

Lesandro Ponciano

2024

Objetivos da Aula

- Discutir aspectos de projeto e implementação
  - Objetivos
  - Estruturas
  - Políticas e mecanismos
- Analisar as estruturas de Sistemas Operacionais
  - Simples, Em Camadas, Microkernels, Em Módulos
- Analisar o papel do SO no contexto de máquinas virtuais

Projeto e Implementação do SO

- Problemas complexos e ainda não solucionados em diversos aspectos
  - Existem apenas abordagens que têm sido bem-sucedidas

- O projeto e implementação do SO
  - Começam pela definição de metas e especificações
  - São afetados pela escolha do hardware e tipo de sistema
  - São afetados pelos objetivos do usuário e do sistema
  - Levam em conta mecanismos e políticas
  - Definem a estrutura (simples, em camadas, microkernels, em módulos)

Objetivos do Projeto

- O projeto pode ser orientado a objetivos dos usuários e/ou objetivos do sistema
  - Podem ser conflitantes

- Objetivos do usuário
  - O SO deve ser conveniente de usar, fácil de aprender, confiável, seguro e rápido

- Objetivos do sistema
  - O SO deve ser fácil de projetar, implementar e manter, bem como flexível, confiável, livre de erros e eficiente

Mecanismos e Políticas

- Mecanismos (como fazer algo)
  - Sempre que a questão é “como” em vez de “o que”, é um mecanismo que deve ser pensado
- Políticas (o que fazer)
  - São importantes para toda alocação de recursos
  - Sempre que é necessário decidir se quer ou não alocar um recurso, uma decisão política deve ser feita
- Políticas tendem a mudar dependendo do ambiente, enquanto os mecanismos não

Estrutura do Código do SO

- SOs são construído para um propósito específico, mas crescem para incluir novas funcionalidades
  - Ex.: *Disk Operating System* (DOS) e UNIX
- O código deve funcionar de maneira apropriada e ser de fácil modificação
  - Apesar de grande e complexo
- Estruturas típicas do código do SO
  - Simples, Em camadas, Microkernels, Em módulos

Estrutura Simples (Monolítico)

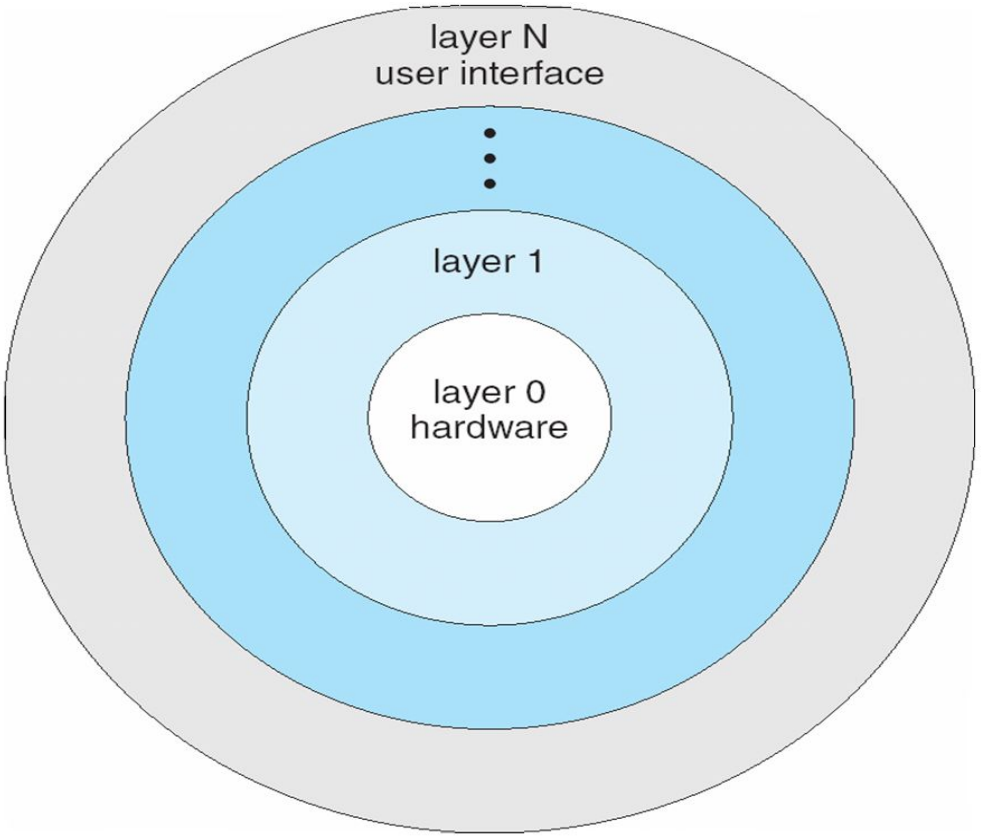
- Não há estruturação visível
  - Ex.: DOS e os primeiros sistemas baseados no UNIX
- Escrito como uma coleção de rotinas
  - Cada rotina pode fazer chamadas a qualquer outra
- Chamadas são requisitadas por meio da
  1) Colocação dos parâmetros em lugares definidos (pilhas e registradores)
  2) Execução de uma chamada de sistema especial ao kernel

Estrutura em Camadas

- O SO é dividido em um certo número de camadas
  - Cada camada é construída no topo das mais baixas
  - A camada interior (camada 0) é o hardware
  - A mais alta (camada N) é a interface do usuário

- Cada camada usa funções (operações) e serviços de apenas camadas de nível mais baixo

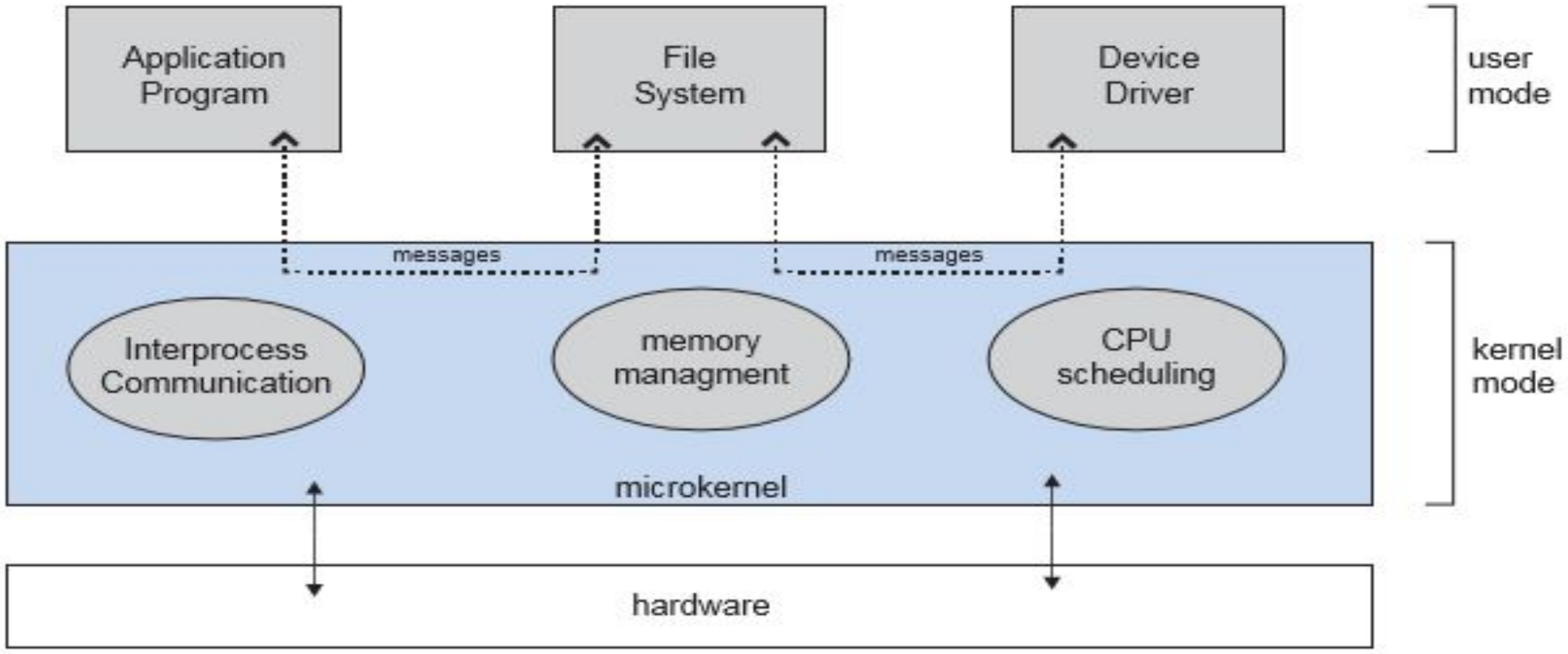
Estrutura em Camadas



Estrutura em Microkernels

- Remove do kernel componentes não essenciais
  - São implementados como programas de sistema ou de usuário
- A comunicação entre os componentes se dá por meio de troca de mensagens

Diagrama da Estrutura Microkernel



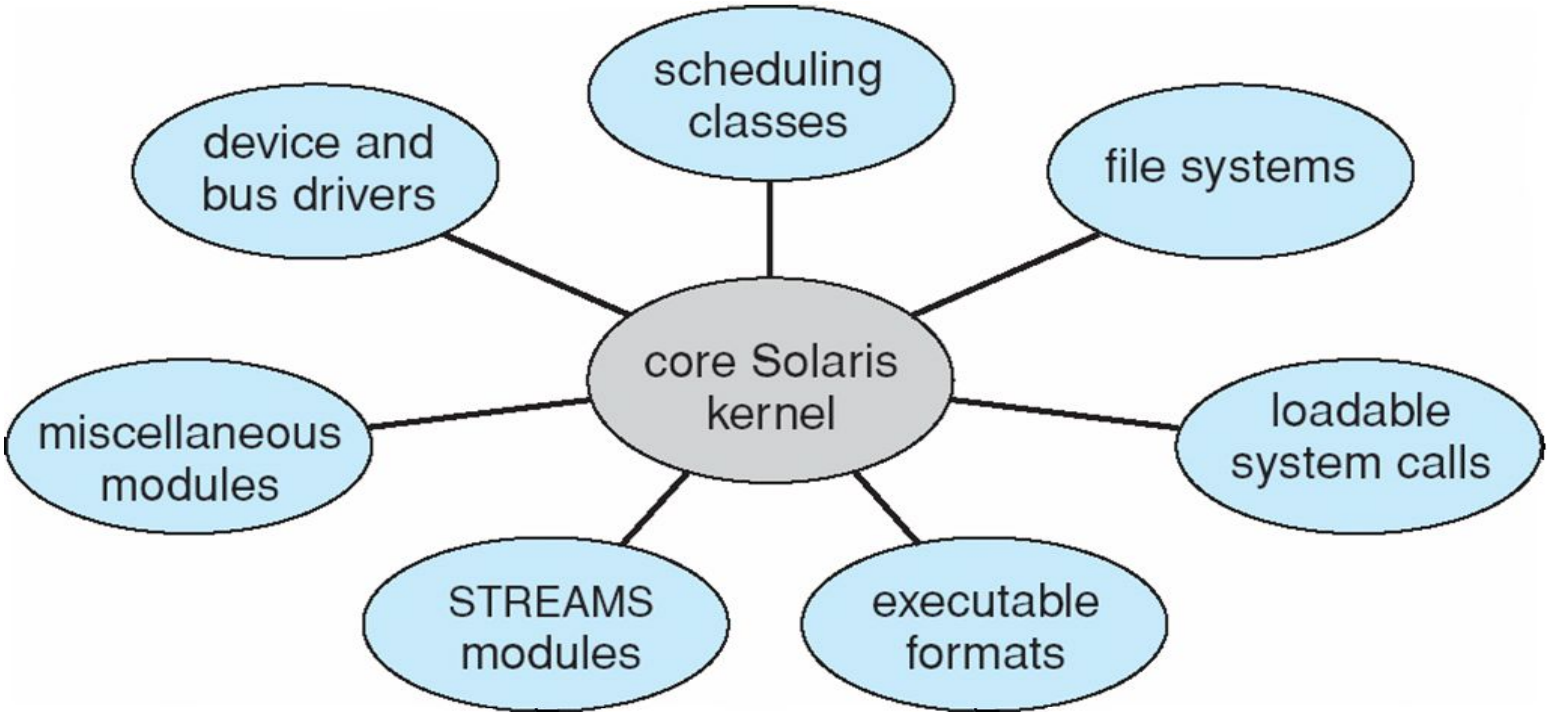
Estrutura em Microkernels

- Benefícios
  - Mais fácil de estender um microkernel
  - Mais fácil de portar o SO para novas arquiteturas
  - Mais confiável (menos código está sendo executado no modo kernel)
  - Mais seguro
- Detrimentos
  - *Overhead* do espaço do usuário para o espaço do kernel

Estrutura em Módulos

- Usa abordagem orientada a objeto
  - Cada componente principal é separado
  - Cada componente conversa com os outros através de interfaces conhecidas
  - Cada componente é carregado dentro do kernel conforme necessário
- Semelhante à estrutura em camadas, entretanto é mais flexível
- Muitos SOs modernos implementam módulos

Módulos no SO Solaris



Pensando sobre as Estruturas

- As estruturas estudadas estão muito relacionadas ao paradigma de programação e às práticas de projeto em vigor na época em que foram propostas

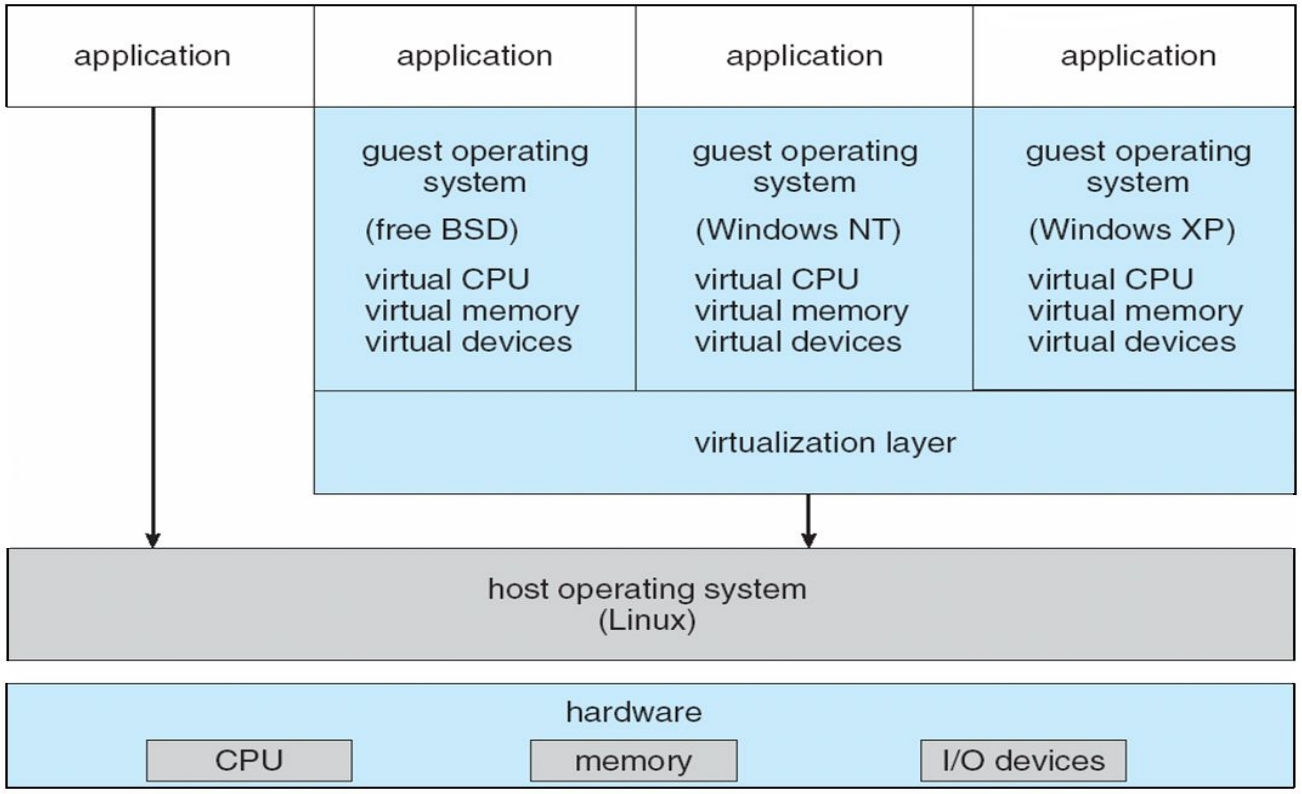
- Mas então, qual é a melhor estrutura?
  - Há argumentos favoráveis e contras cada tipo
  - O principal aspecto de conflito é quanto ao requisito de segurança
  - Pesquise sobre o famoso debate entre Andrew S. Tanenbaum e Linus Torvalds em 1992 e depois em 2006

Máquinas Virtuais

- Emular uma máquina ou servir de interface

- Emular uma ou várias máquinas (virtuais) dentro de uma mesma máquina física (hospedeira)
  - Uma máquina virtual fornece uma interface idêntica ao hardware de uma máquina física
  - A máquina virtual tem a ilusão de que tem seus próprios recursos, como processador, memória, etc

Arquitetura do VMware



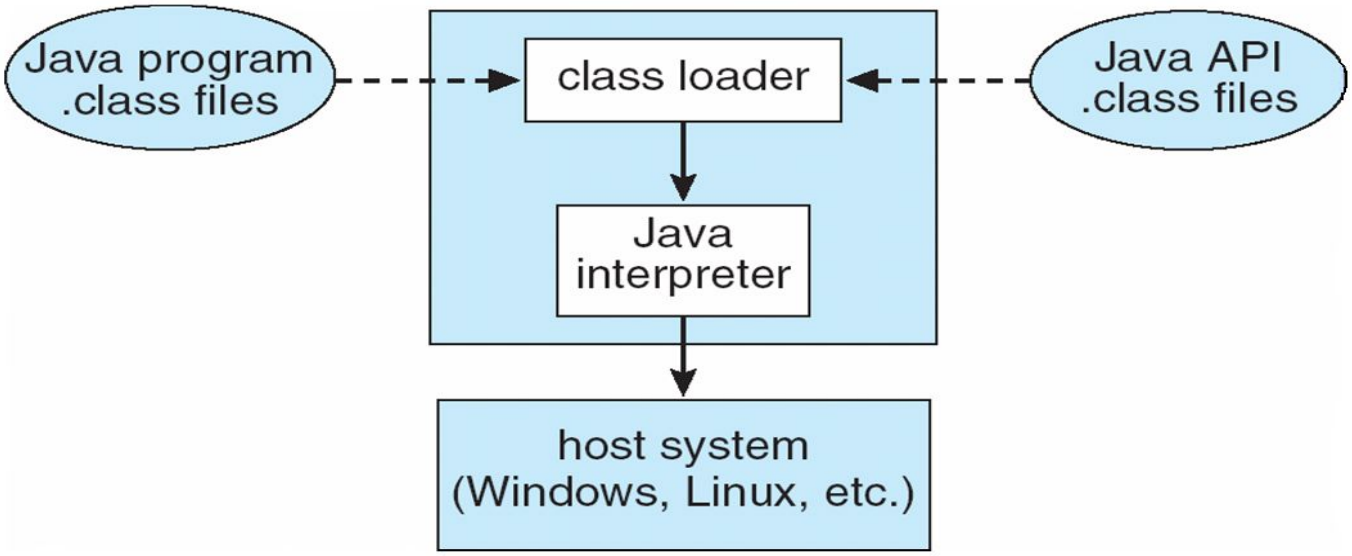
Três máquinas virtuais executando em uma mesma máquina física

Máquinas Virtuais

- Construir uma interface

- Exemplo: Máquina Virtual Java
  - O código é gerado para ser entendido pela máquina virtual
  - A máquina virtual faz a conversão do código para que ele seja entendido pela máquina física e SO hospedeiro
  - Permite a portabilidade da aplicação para diferentes sistemas operacionais e diferentes hardwares

Máquina Virtual Java



O uso da máquina virtual java permite que um mesmo programa java (*bytecode*) execute em diferentes ambientes de computação

Características

- Em uma mesma máquina física, pode-se criar máquinas virtuais com diferentes configurações
  - Diferentes SOs
  - Diferentes configurações de hardware
- Vários ambientes de execução podem compartilhar o mesmo hardware ao mesmo tempo
  - Ambientes protegidos um do outro, mas algum compartilhamento de arquivo pode ser permitido
  - Proteção entre a máquina física e as máquinas virtuais
  - O SO que executa na máquina virtual gerencia recursos virtuais

Prós e Contras

- Vantagens/Aplicações
  - Testar software em diferentes configurações de SO e hardware
  - Consolidação de muitos sistemas que usam poucos recursos em poucos sistemas mais utilizados
  - Prover recursos a terceiros sem colocar as máquinas físicas em risco (computação na nuvem)
- Desvantagens
  - Sobrecarga na máquina física afeta todas as máquinas virtuais
  - A emulação da máquina virtual em si é um programa em execução e, portanto, consome recursos do hardware

Atividade de Fixação

A estrutura de sistemas operacionais que impõe uma rígida hierarquia de uso de serviços entre componentes e que restringe que apenas um subconjunto deles tenha acesso direto ao hardware é:

a) Camadas
b) Módulos
c) Microkernel
d) Monolítico

Material Complementar

- **Texto**: “The Tanenbaum-Torvalds Debate - 1992” Disponível em https://www.oreilly.com/openbook/opensources/book/appa.html Acesso em: 04 Fev. 2024
- **Texto**: “Tanenbaum-Torvalds Debate Part II" 2006”. Disponível em: https://www.cs.vu.nl/~ast/reliable-os/ Acesso em: 04 Fev. 2024

Referências

TANENBAUM, Andrew S. Sistemas operacionais modernos. 3. ed. São Paulo: Pearson Prentice Hall, 2009. xvi, 653 p. ISBN 9788576052371

SILBERSCHATZ, Abraham; GALVIN, Peter B.; GAGNE, Greg. Fundamentos de sistemas operacionais: princípios básicos. Rio de Janeiro, RJ: LTC, 2013. xvi, 432 p. ISBN 9788521622055 (Capítulos 1, 2 e 3)

Há várias estruturas de organização do código do sistema operacional propostas ao longo da história. A escolha da estrutura impacta requisitos não funcionais, como a manutenibilidade. Nem sempre é trivial escolher entre uma arquitetura ou outra em termos de requisitos como segurança.

Sistemas Operacionais

Prof. Dr. Lesandro Ponciano

https://orcid.org/0000-0002-5724-0094