Prof. Ricardo Inácio Álvares e Silva

Principais Conceitos

Sistemas Operacionais

Principais Conceitos

Estruturas de Sistemas Operacionais

Organização do espaço do supervisor

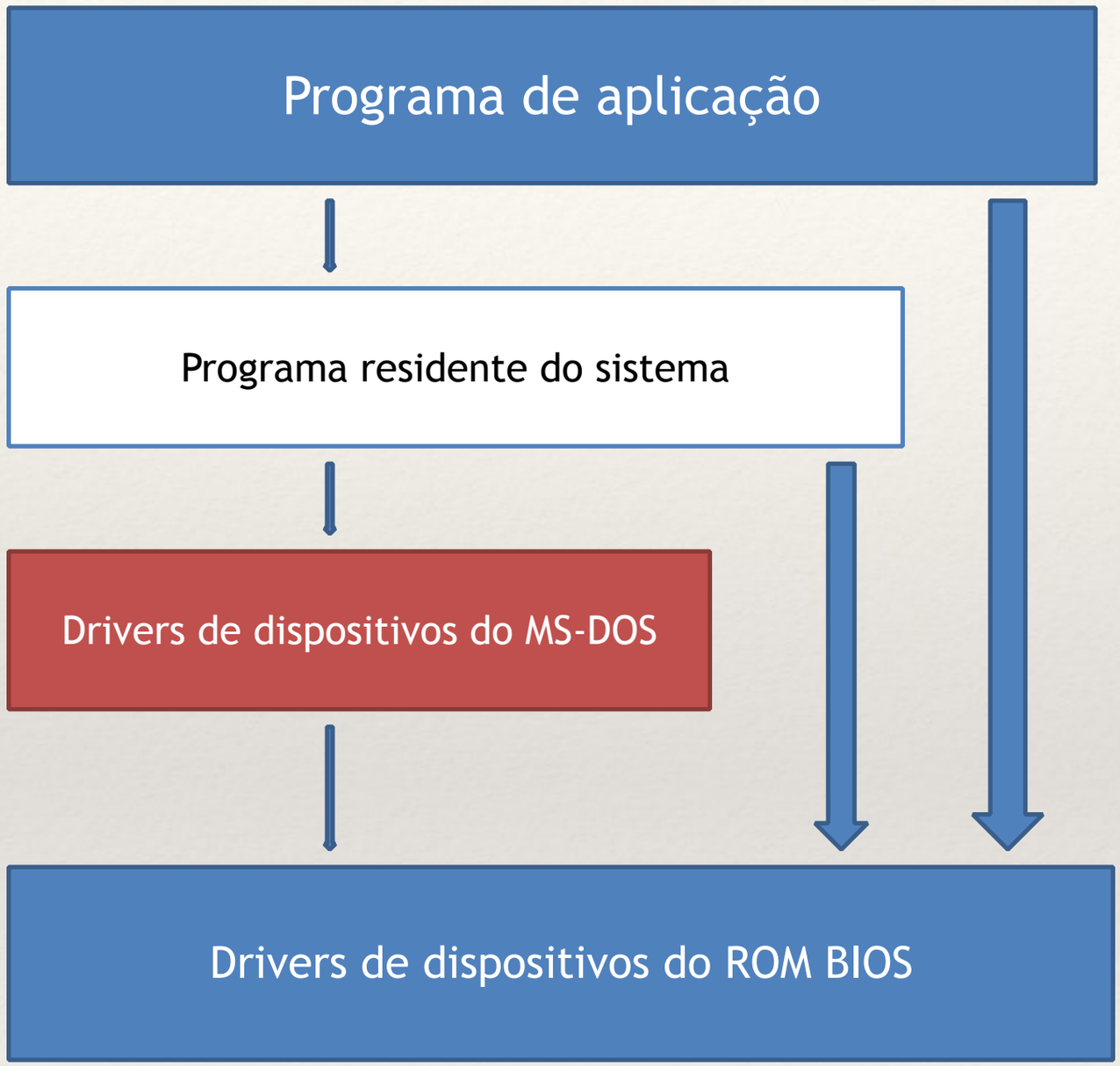
Escopo da Aula

- Estruturas de Sistemas Operacionais
  - Estrutura simples
  - Monolítica
    - com módulos
  - Em camadas
  - Microkernel
- Máquinas Virtuais

- Sistemas operacionais são projetos
  - grandes e complexos, milhões de linhas de código
  - estão sujeitos a diversos erros e falhas
- Por isso precisam ser feitos com cuidado para terem
  - **estabilidade**
  - **segurança**
  - **manutenibilidade**

Estrutura Simples

- Não há separação entre **espaço de usuário** e **espaço do *kernel*** (ou supervisor)
- Programa do usuário pode acessar qualquer funcionalidade do sistema
- Programa do usuário pode inclusive modificar as rotinas do sistema
- Não necessita de suporte específico de hardware
  - é uma solução barata
  - sistemas de hardware simples
- Exemplos:
  - MS-DOS, primeira versão do UNIX



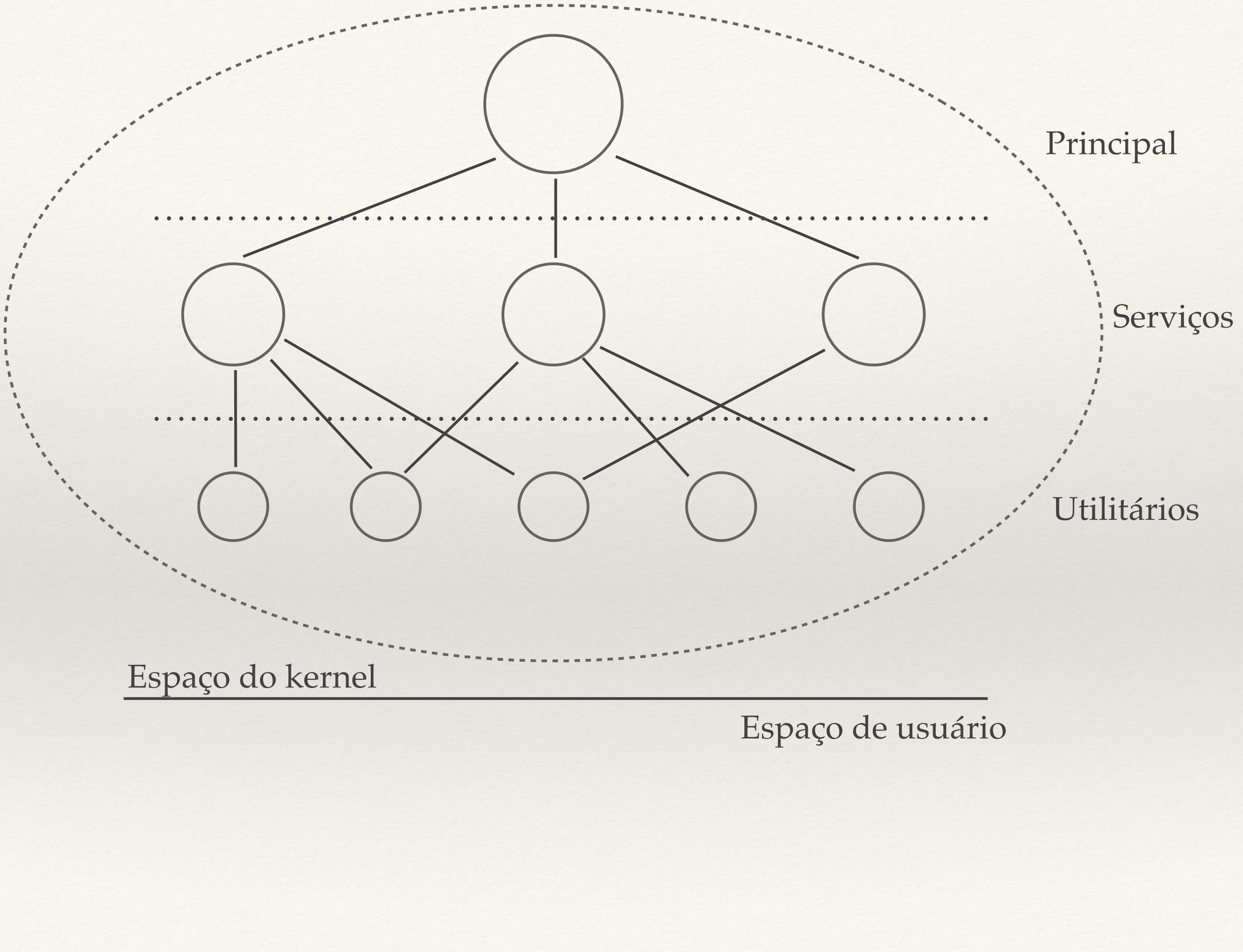
Estrutura do MS-DOS

Kernels Estruturados

- Necessita suporte do hardware
  - sistemas mais simples são impossibilitados de tomar essa abordagem
- Funcionalidades e recursos gerais são separados em componentes.
- Possibilidade de ocultação de informação
  - desenvolvedores do *kernel* devem respeitar as interfaces existentes
  - liberdade para mudar implementação e criar novos recursos
- Distinção do espaço de usuário (*userland*) e espaço de kernel (supervisor)
- Tipos:
  - Monolítica, com módulos, em camadas e microkernel

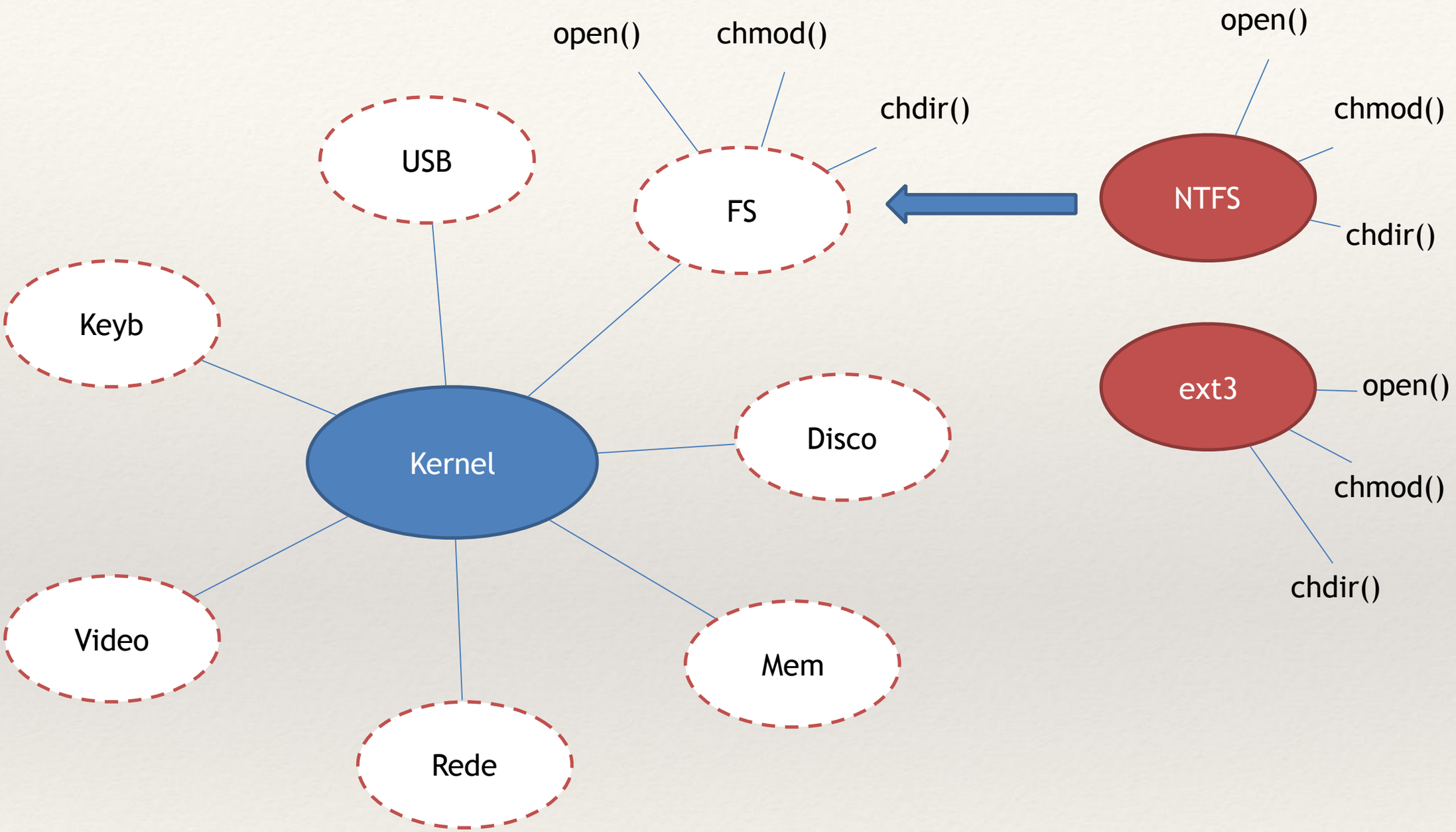
Estrutura Monolítica

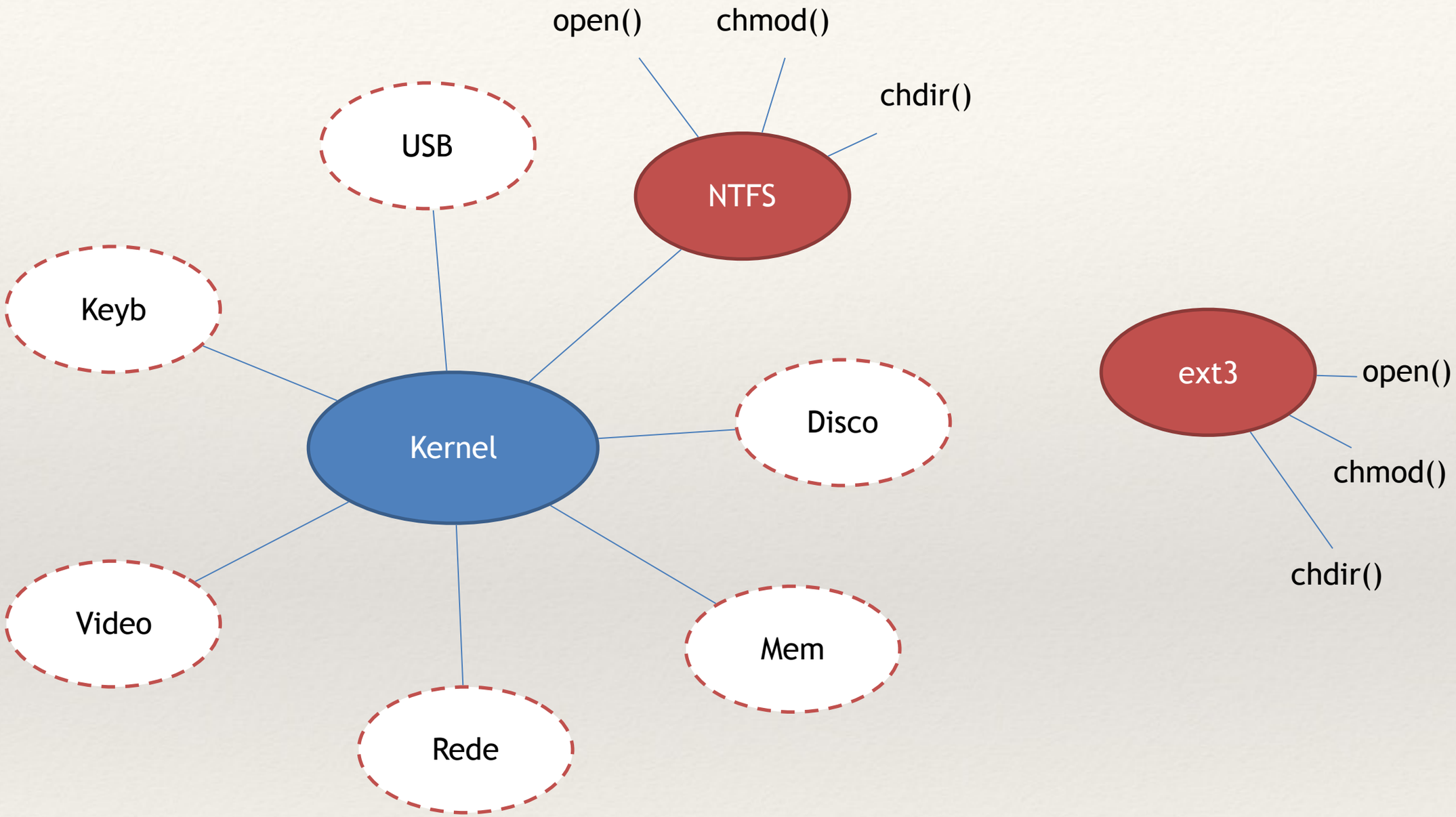
- Funcionalidades separadas em vários objetos
- O sistema inteiro é feito como um só
  - Os vários objetos são compilado e link-editados em um único programa binário e executável
- Qualquer chamada ao sistema está disponível para qualquer programa
  - programas de usuário
  - outras chamadas do próprio sistema
- Não há ocultação de informações e interfaces dentro do *kernel*, mas as chamadas ao sistema podem ser estruturadas em vários objetos, que são link-editados em um único
- Exemplos:
  - Linux, Windows XP até Windows 10


Principal
Serviços
Utilitários
Espaço do kernel
Espaço de usuário

Estrutura Monolítica com Módulos

- O sistema continua sendo um único objeto, porém aceita extensões, os módulos
- Módulos são acoplados ao *kernel* dinamicamente, em tempo de execução
- Módulos ficam no espaço de *kernel*
- Módulos respondem por chamadas ao sistema
  - Chamadas ao sistema são apenas interfaces, implementadas por terceiros
- Utilizado por:
  - OS X, Solaris, Linux, Windows 95/98, Windows XP em diante

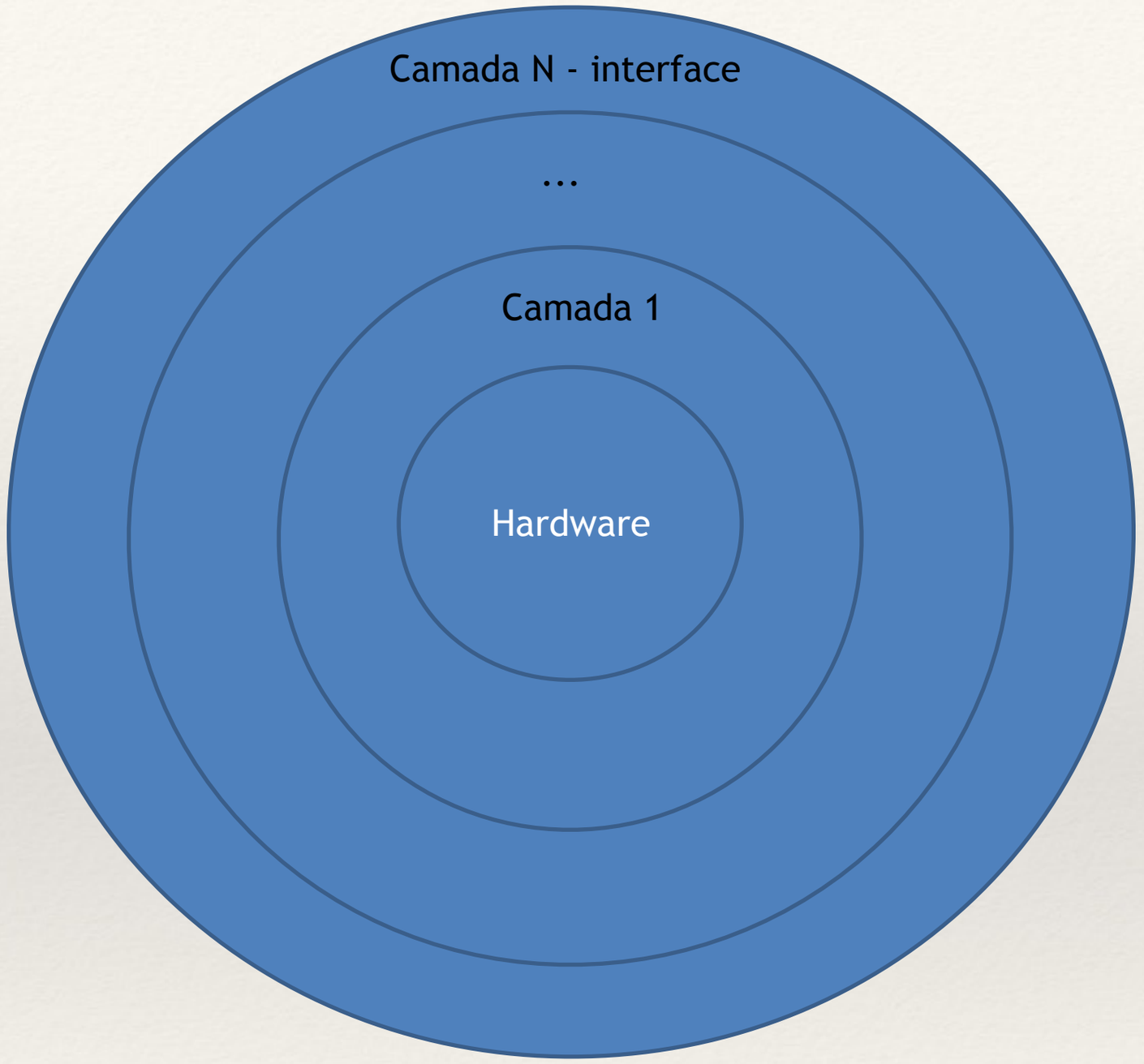

open()
chmod()
chdir()
USB
FS
NTFS
open()
chmod()
chdir()
Keyb
Kernel
Disco
ext3
open()
chmod()
chdir()
Video
Mem
Rede


open()
chmod()
chdir()
USB
NTFS
Keyb
ext3
open()
chmod()
chdir()
Kernel
Disco
Video
Mem
Rede

Estrutura em Camadas

- Sistema operacional é dividido em uma série de N camadas, ou N níveis
  - Camada 0, a inferior, é o hardware
    - Camada N, a mais alta, é a interface com o usuário (programas)
- Camada N só pode acessar chamadas da camada N - 1
- Camada N - 1 serve a camada N, fazendo chamadas à camada N – 2

- Dificuldade de implementação, definir o que fica em qual camada
  - Exemplo: drivers de dispositivos de armazenamento fica abaixo do gerenciador de memória, já que é acionado por este
- Desempenho inferior em relação a outras estruturas
  - necessidade de propagação de chamadas ao sistema pelas camadas
  - Transformações constantes nos dados sendo repassados entre as camadas
- Exemplos:
  - THE, MULTICS

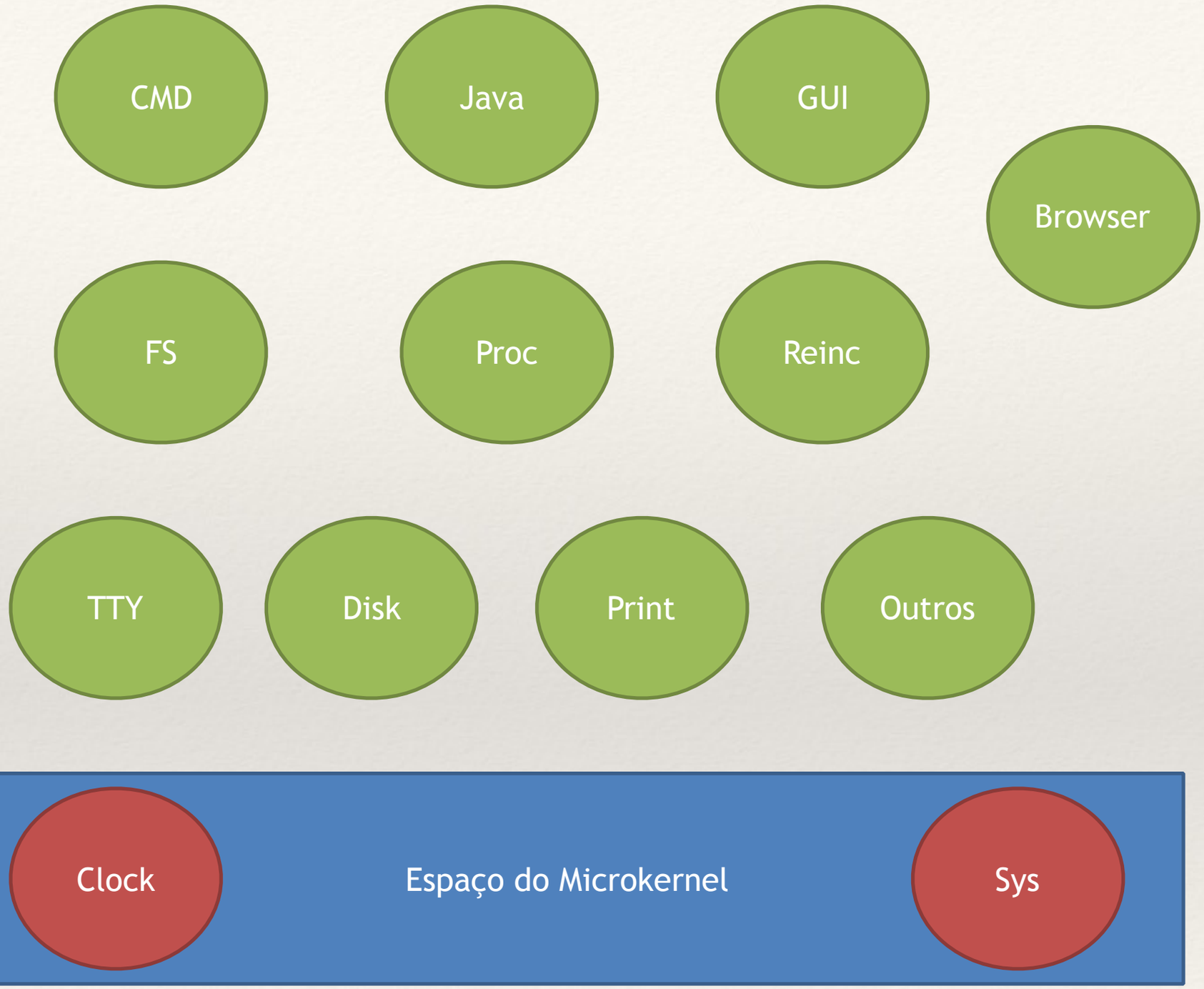


Estrutura do MULTICS

Estrutura de Microkernel

- Pesquisas sobre qualidade de código em sistemas industriais mostraram que, em casos mais comuns, um sistema apresenta cerca de 10 bugs para cada 1000 linhas de código
- Sistemas operacionais normalmente possuem milhões de linhas de código, o que acarreta em cerca de dezenas de milhares de bugs
- Um único bug que derrube um sistema monolítico ou uma camada, pode comprometer todo o funcionamento do computador
- Conclusão: módulos independentes que no caso de falhas não comprometam os outros
  - em muitos casos, podem apenas ser reiniciados

- O *microkernel* é um programa mínimo no espaço do kernel
  - apenas **escalona** (escolhe e troca) **processos** em execução na CPU
  - exemplo: o *microkernel* do Minix3 tem apenas ~3.200 linhas de código
- Todos os outros módulos funcionam no espaço de usuário
  - drivers de dispositivos, gerenciador de memória, gerenciadores de recursos, sistemas de arquivos, etc.
- Há pouquíssimas chamadas ao sistema de fato
  - a comunicação entre os módulos se dá por **troca de mensagens**
  - por esse motivo, o desempenho do sistema é pior
- Essa estrutura troca o desempenho por estabilidade
- Exemplos:
  - QNX (BlackBerry 10), Symbian, Windows NT 1.0 até 3.0, Minix3, Hurd

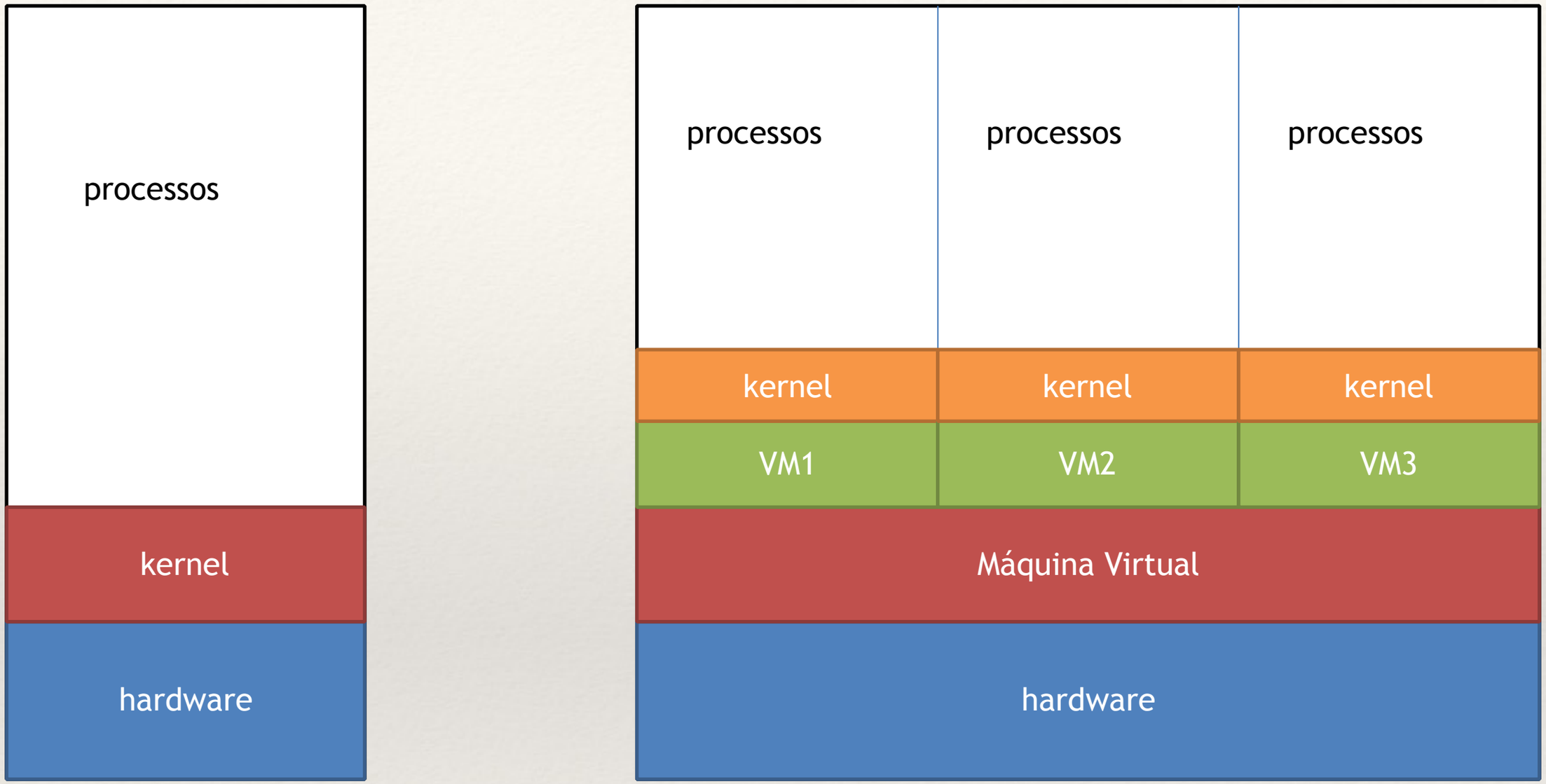


Estrutura *microkernel* do Minix3

Máquinas Virtuais

- Separa o hardware em vários ambientes de execução
- Cada ambiente executa seu próprio *kernel*
  - da mesma forma que cada *kernel* executa vários processos, em sistemas comuns
- Motivação:
  - facilidade de desenvolvimento e testes de sistemas operacionais
  - aluguel de tempo de processamento e recurso em computadores compartilhados
  - em caso de falha irreversível, basta restaurar uma imagem do ponto anterior
  - proteção ao sistema operacional principal, já que se houver alguma falha grave de hardware no sistema virtualizado, apenas este falhará
  - protege também os diversos sistemas operacionais funcionando paralelamente

- Dois modelos de VM
  - **Hipervisor tipo 1**: principal sistema é a VM
    - exemplos: Xen, VMware ESX, Hyper-V
  - **Hipervisor tipo 2**: VM é um programa de usuário sobre outro SO
    - exemplos: Virtualbox, VirtualPC, Parallels, VMware Player



Sistema comum

Sistema VM