Sistemas Operacionais

Introdução - Estrutura de um SO

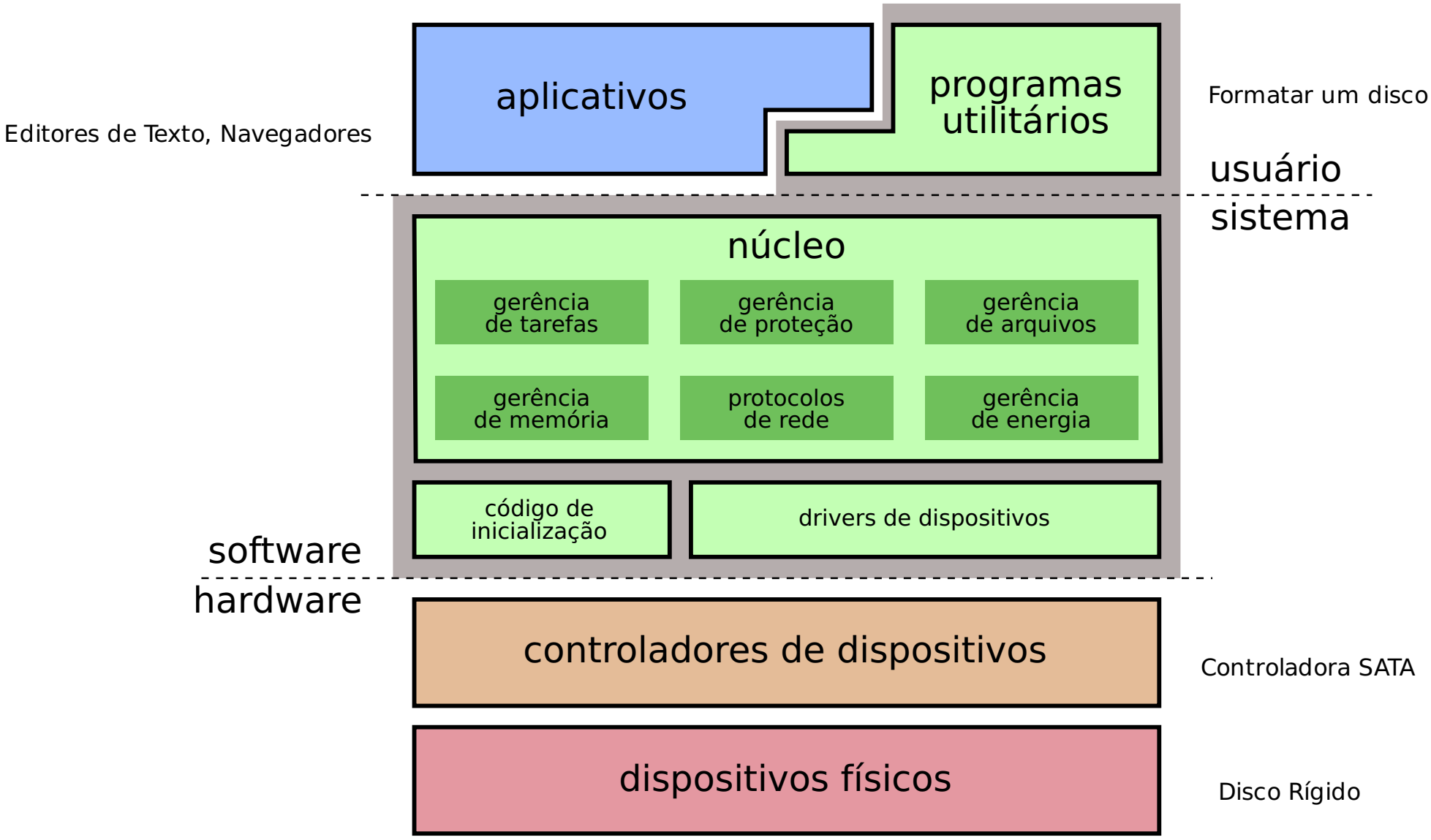
Prof. Carlos Maziero

DInf UFPR, Curitiba PR

Julho de 2020

Conteúdo

Estrutura de um SO



Estrutura de um SO

Componentes mais relevantes:

Núcleo : gerência dos recursos do hardware usados pelas aplicações. Também implementa as principais abstrações utilizadas pelos aplicativos.

Inicialização : reconhece os dispositivos instalados e carrega o núcleo do sistema na memória.

Drivers : módulos de código para acessar os dispositivos físicos.

Utilitários : funcionalidades complementares: formatação de discos, *shell* de comandos, interface gráfica, etc.

Estrutura de um SO

Política

Aspecto abstrato de alto nível: decidir a quantidade de memória para cada aplicação, o próximo pacote de rede a enviar, etc.

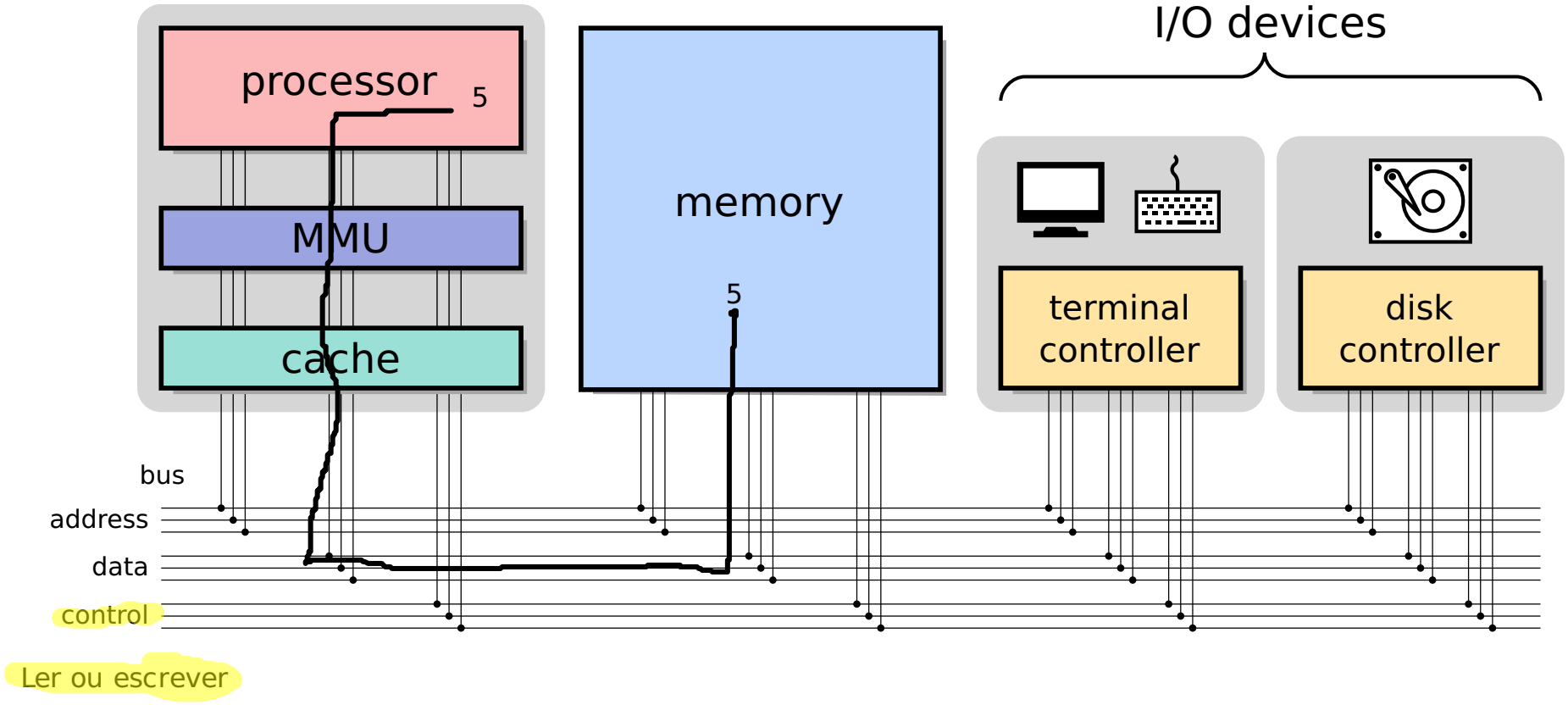
Mecanismo

Procedimento de baixo nível usado para implementar políticas: como iniciar um processo, enviar um pacote de rede, etc.

Filosofia da estrutura: separar políticas de mecanismos

- As políticas devem ser independentes dos mecanismos
- Os mecanismos devem ser genéricos para várias políticas

Arquitetura de um computador



Interrupções, exceções e traps

Mecanismos de hardware usados para desviar a execução do processador em caso de eventos:

Interrupção : desvia a execução por evento externo, gerado por um periférico

Evento gerado pelo hardware, por exemplo, teclado.

Exceção : desvia a execução por evento interno (erro numérico, etc)

Divisão por zero

Trap : desvia a execução a pedido do software

Níveis de privilégio

- Implementados pelos processadores modernos
- No Multics: anéis de proteção (0 ... 7)
- Nas CPUs Intel: 4 níveis

Acesso restrito aos recursos

Código que está rodando pode fazer o que quiser

Níveis de privilégio

Nível núcleo (*supervisor*, *sistema*, *monitor*):

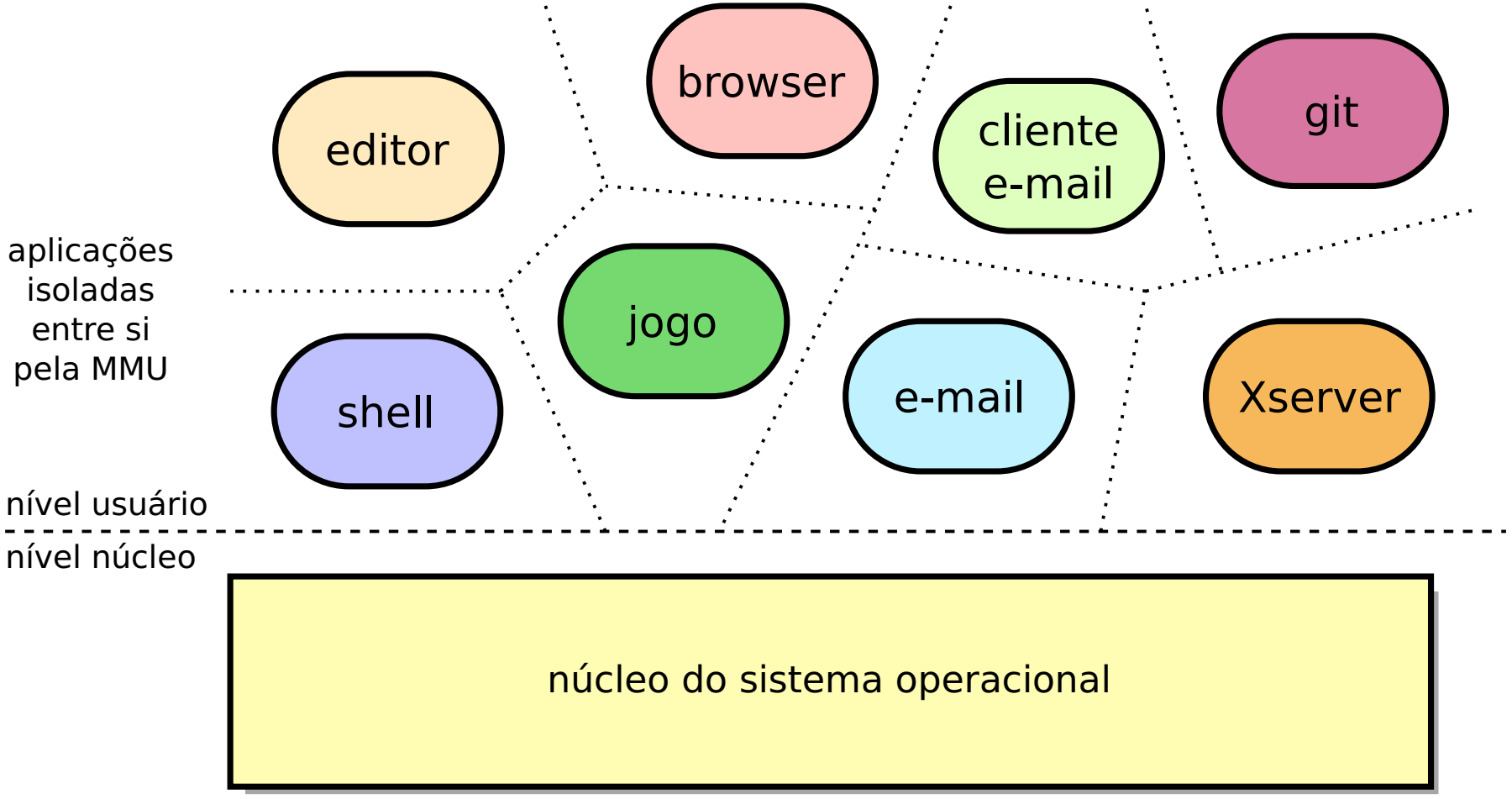
- Acesso amplo aos recursos:
  - todas as instruções do processador
  - todos os registradores
  - todas as portas de entrada/saída
  - todas as áreas da memória
- Nível usado pelo núcleo e os *drivers*.

Níveis de privilégio

Nível usuário (*userspace*):

- Acesso restrito aos recursos:
  - subconjunto das instruções do processador
  - subconjunto de registradores
  - subconjunto de portas de entrada/saída
  - subconjunto de áreas da memória
- Tentativas de acesso inválidas geram exceções
- Nível usado pelos utilitários e aplicações

Separação do núcleo



Estabilidade (Bug contido em uma área de memória) e
Segurança (busca por informações de cartões de crédito)

Chamadas de sistema

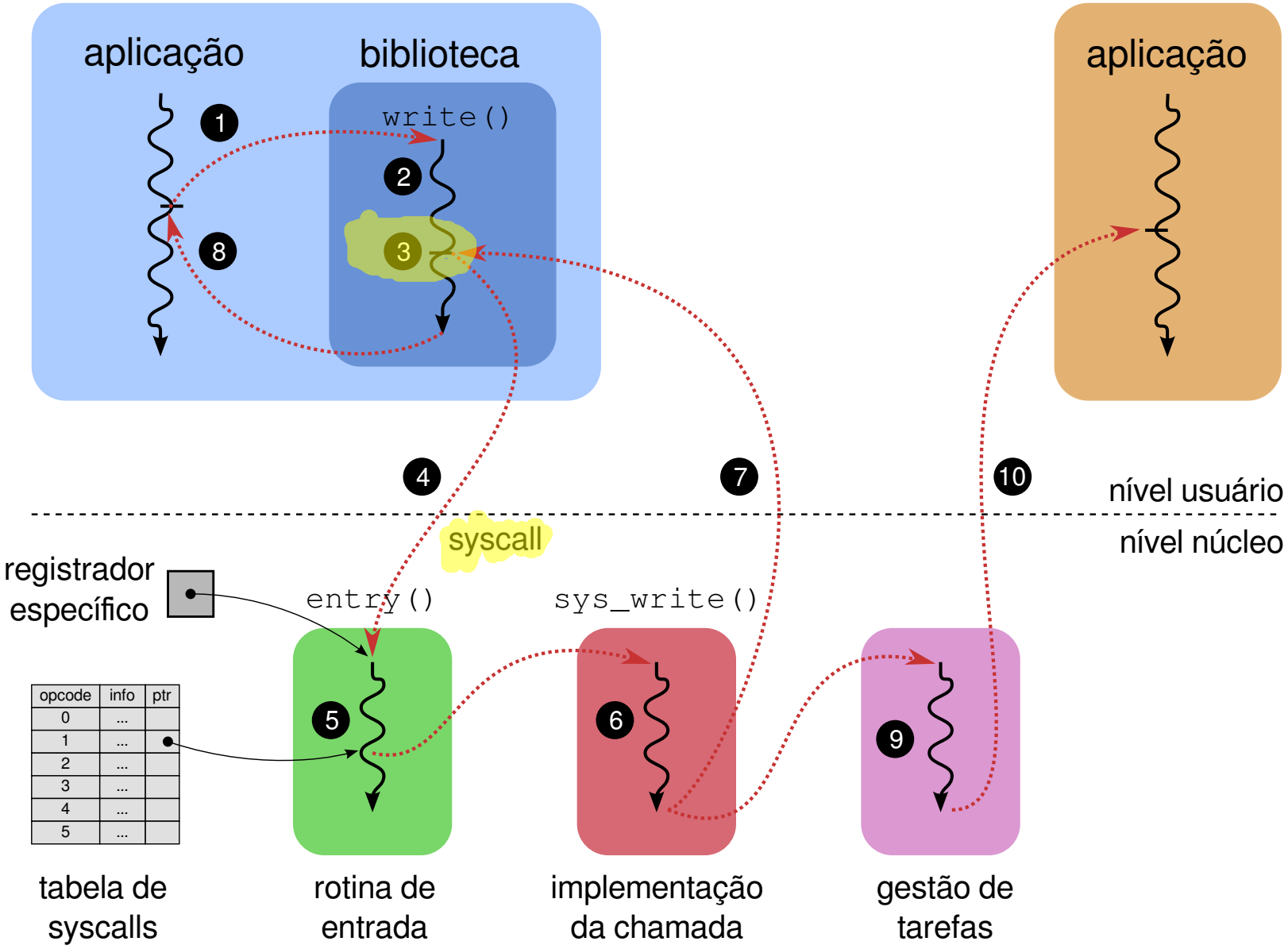
São funções que permitem o acesso aos serviços do núcleo:

- Abrir/ler/escrever/fechar arquivos
- Enviar/receber dados através da rede
- Ler teclado
- Escrever dados na tela

Problema:

Como uma aplicação pode invocar uma função do núcleo?

Etapas de uma chamada de sistema



Etapas de uma chamada de sistema

1. Aplicação chama `write(...)` da biblioteca.
2. A função `write` preenche os registradores da CPU.
3. A função `write` invoca uma chamada de sistema.
4. A CPU vai para o núcleo e ativa a rotina de entrada (`entry`).
5. A rotina de entrada consulta a tabela de *syscalls* e ativa a função `sys_write`.
6. A função `sys_write` efetua a operação solicitada.
7. A CPU retorna à função `write`, em modo usuário.
8. A função `write` retorna ao código principal da aplicação.

Conjunto de chamadas de sistema

Processos: criar, carregar código, terminar, esperar

Memória: alocar/liberar/modificar áreas de memória

Arquivos: criar, remover, abrir, fechar, ler, escrever

Comunicação: criar/destruir canais, receber/enviar dados

Dispositivos: ler/mudar configurações, ler/escrever dados

Sistema: ler/mudar data e hora, desligar o sistema

Cada SO define seu próprio conjunto de *syscalls*: OS API

Interface de Programação de Aplicações

- Sistemas Windows: *Win32*
- Sistemas UNIX: *POSIX*