Introdução a Sistemas Operacionais

Professor: Pedro Horchulhack

Agenda

1. Apresentação do módulo
2. Introdução a sistemas operacionais
3. Atividade

Prof. Pedro Horchulhack

Formação acadêmica

- Bacharel em Ciência da Computação (PUCPR)
- Mestre em Informática (PUCPR)
- Doutorando em Informática (PUCPR)

Áreas de atuação

- Virtualização (Containers)
- Aprendizagem de Máquina Aplicada a Cibersegurança
- Aprendizagem de Máquina Federada
- Sistemas Distribuidos

Organização do curso

1. Visão geral de sistemas operacionais
    1. Estrutura de um SO
    2. Virtualização e seus tipos
2. Básico de SOs baseados em Linux
3. Containerização e suas tecnologias
    1. Docker
    2. Kubernetes
4. Segurança de containers

– Avaliações mensais entregues no AVA

O que é um Sistema Operacional?

- Hardware e software são coisas distintas
  - “Hardware é o que chutamos. Software é o que xingamos”
- Software intermediário
- Um SO **dois** principais objetivos: **abstrair** e **gerenciar**.

O que é um Sistema Operacional?

Abstração

- Vamos pensar no seguinte exemplo de leitura de arquivo:
  1. Verificamos se os parâmetros do arquivo estão corretos (nome, descritor de arquivo, buffer de memória está disponível)
  2. Verificar se o disco está disponível
  3. Ligar o motor do disco e esperar ele atingir a velocidade correta
  4. Posicionar a cabeça de leitura sobre a trilha da tabela de diretórios
  5. Ler a tabela de diretórios e encontrar onde está o subdiretório do arquivo solicitado
  6. Mover a cabeça até a posição inicial do arquivo
  7. Ler o bloco inicial e transferi-lo ao buffer na memória

O que é um Sistema Operacional?

Abstração

– Através do exemplo conseguimos identificar três coisas:

1. Precisamos de uma interface mais fácil para **interagir** com os dados
2. Precisamos de uma forma **homogênea** para acessar os dispositivos
3. Tornar as aplicações **independentes** do hardware

O que é um Sistema Operacional?

Gerenciamento

- É comum que múltiplas aplicações interajam com muitos hardwares simultaneamente

- Se não houver uma gestão dos recursos físicos, pode induzir erros de:
  - Disputa
  - Conflito

O que é um Sistema Operacional?

Gerenciamento

Exemplos:

1. Multiplas aplicações sobre um núcleo de CPU
2. Impressora

O que é um Sistema Operacional?

Funcionalidades

- Gestão de CPU
- Gestão de Memória
- Gestão de Entrada/Saída (Dispositivos)
- Gestão de Arquivos
- Gestão de Segurança

O que é um Sistema Operacional?

- Mesmo que tenhamos diversos recursos para gerenciar, ainda temos outros que são complementares:
  - Interface gráfica
  - Fontes de energia
  - Suporte de multimídia
  - Suporte de rede

- Também podemos notar que todos os recursos operam de maneira **interdependente**

O que é um Sistema Operacional?

- Até aqui entendemos que existem uma série de recursos para gerenciar.
  - Como?

- Através de **políticas** e **mecanismos** conseguimos estabelecer e implementar alguns critérios para o SO.
  - Políticas: regras abstratas de **como** algo deve funcionar
  - Mecanismos: **implementação** das regras

O que é um Sistema Operacional?

Categorias de SOs

- Em lote (batch)
- De rede
- Distribuídos
- Multi-usuário
- Servidores
- Desktops
- Móveis
- Tempo-real
- Embarcados

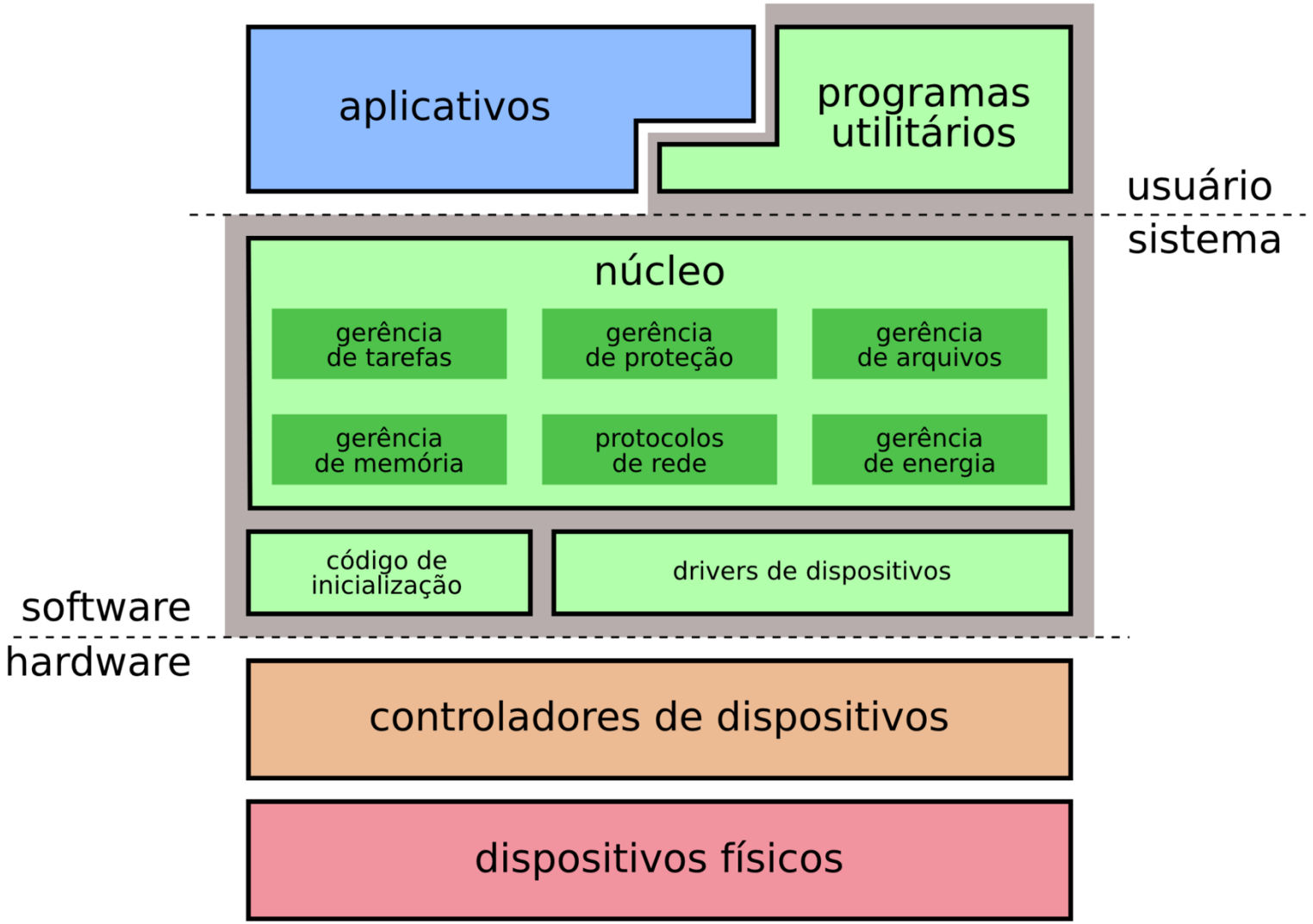
Estrutura de um SO

Estrutura de um SO

- Um SO é uma composição de diversos softwares que interagem com o hardware

- Os componentes principais são:
  - Núcleo
  - Código de Inicialização
  - *Drivers*
  - Programas utilitários

Estrutura de um SO

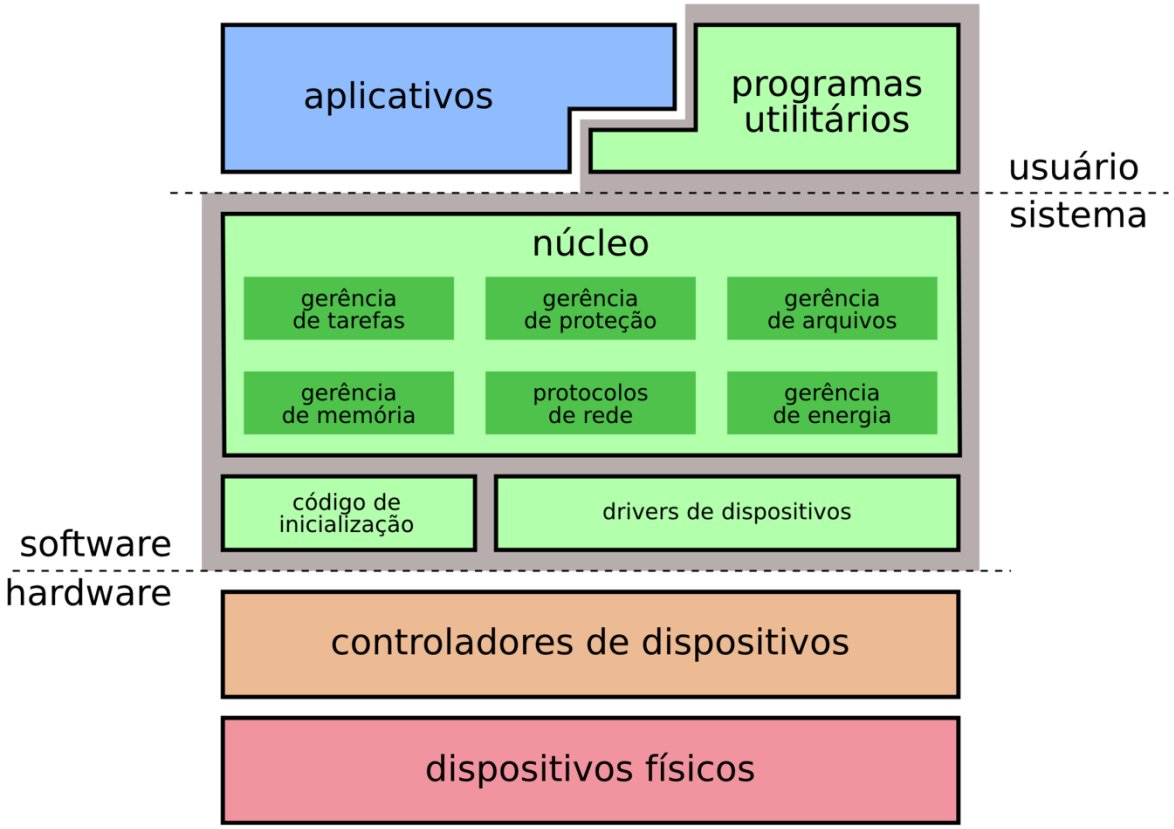


Estrutura de um SO típico (MAZIERO, p. 14, 2019).

Estrutura de um SO

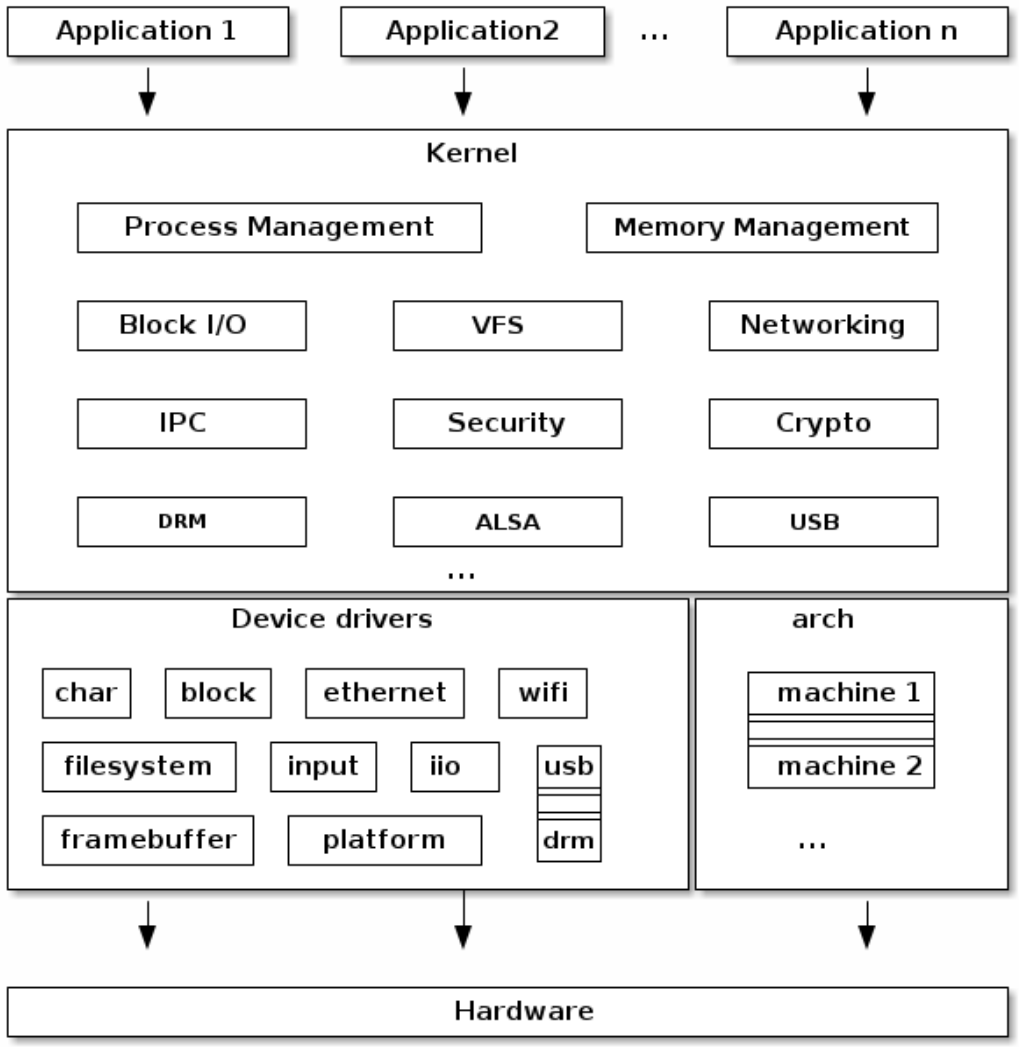
- Modos de execução
  - Privilegiado
  - Usuário

- É comum que SOs sigam essa estrutura, porém geralmente são bem mais complexas.



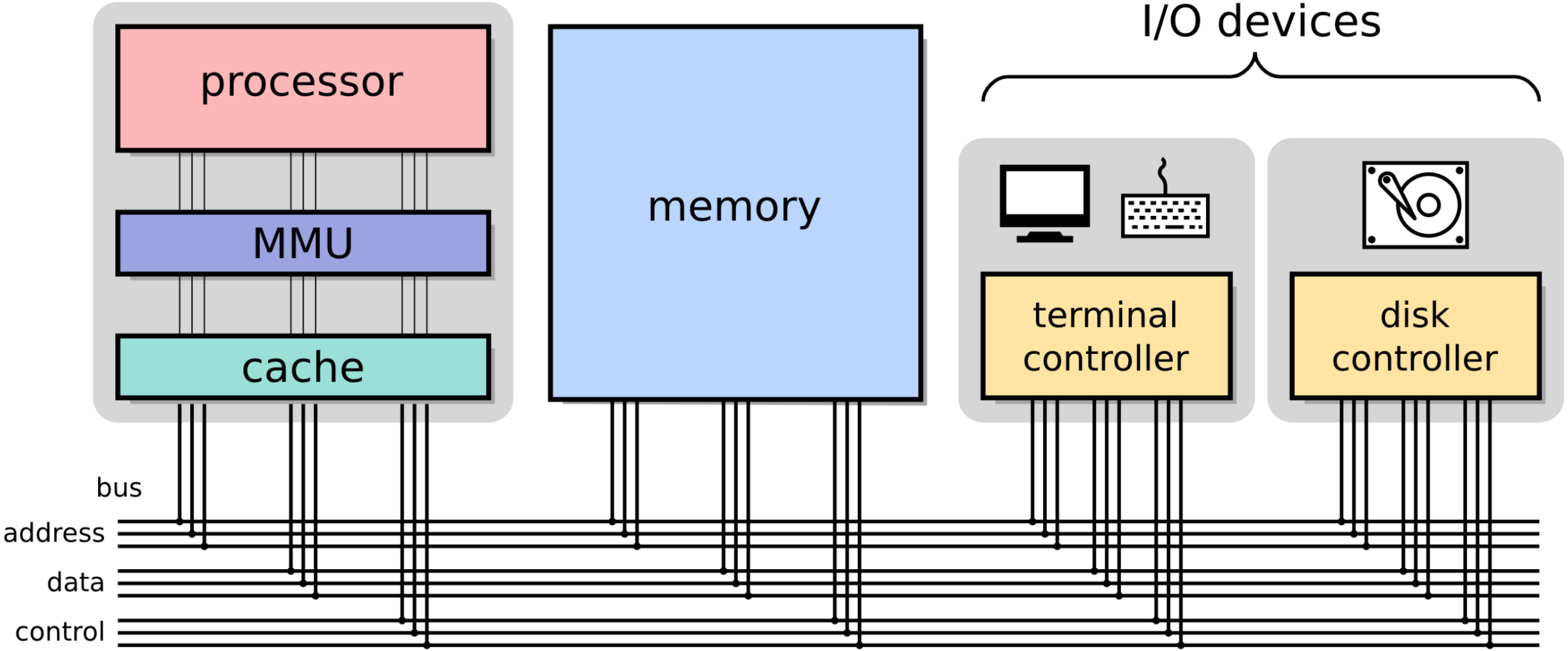
Estrutura de um SO típico (MAZIERO, p. 14, 2019).

Estrutura de um SO



Estrutura do kernel do Linux (LINUX, c2024)

Estrutura de um SO



Arquitetura de um computador típico(MAZIERO, p. 16, 2019).

Estrutura de um SO

- Quando queremos que o programa se comunique com dispositivos (entrada/saída), recorremos ao que se chama de **interrupção**.

- O **controlador** de dispositivos pode escolher:
  - Aguardar que o processador faça uma consulta a ele para ver se uma tarefa acabou
  - Notificar o processador (interrupção, IRQ – Interrupt ReQuest) através do barramento de controle

- Ao receber a notificação de interrupção o processador **suspende** seu fluxo de execução para atender ao dispositivo.
  - O processador desvia sua execução para um endereço pré-definido onde se encontra um *interrupt handler*

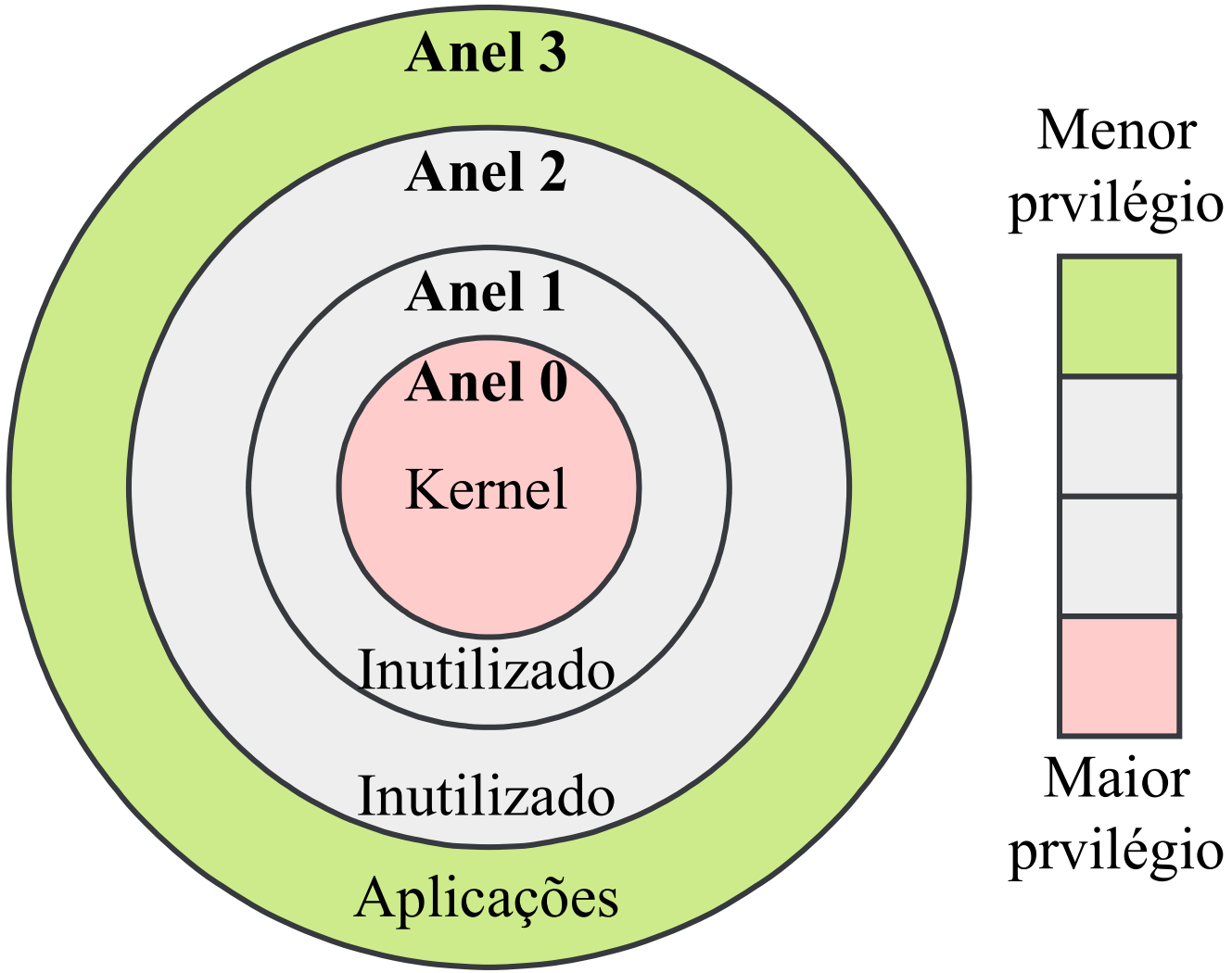
Estrutura de um SO

- Também temos a ideia de **exceções**, que são interrupções enviadas do controlador ao CPU, porém para indicar **erros** que ocorreram durante a execução

- Se não existissem as interrupções o processador deveria verificar a todo instante se os dispositivos terminaram ou necessitam de alguma demanda.
  - O sistema fica mais lento

Estrutura de um SO

- Níveis de privilégio buscam separar **quem** pode executar operações críticas ou não
- Aplicações com acesso **completo** ao hardware poderia implicar em problemas de segurança
  - Facilmente contornar mecanismos de controle de acesso
- Dessa forma, processadores modernos implementam **níveis de proteção de execução**.

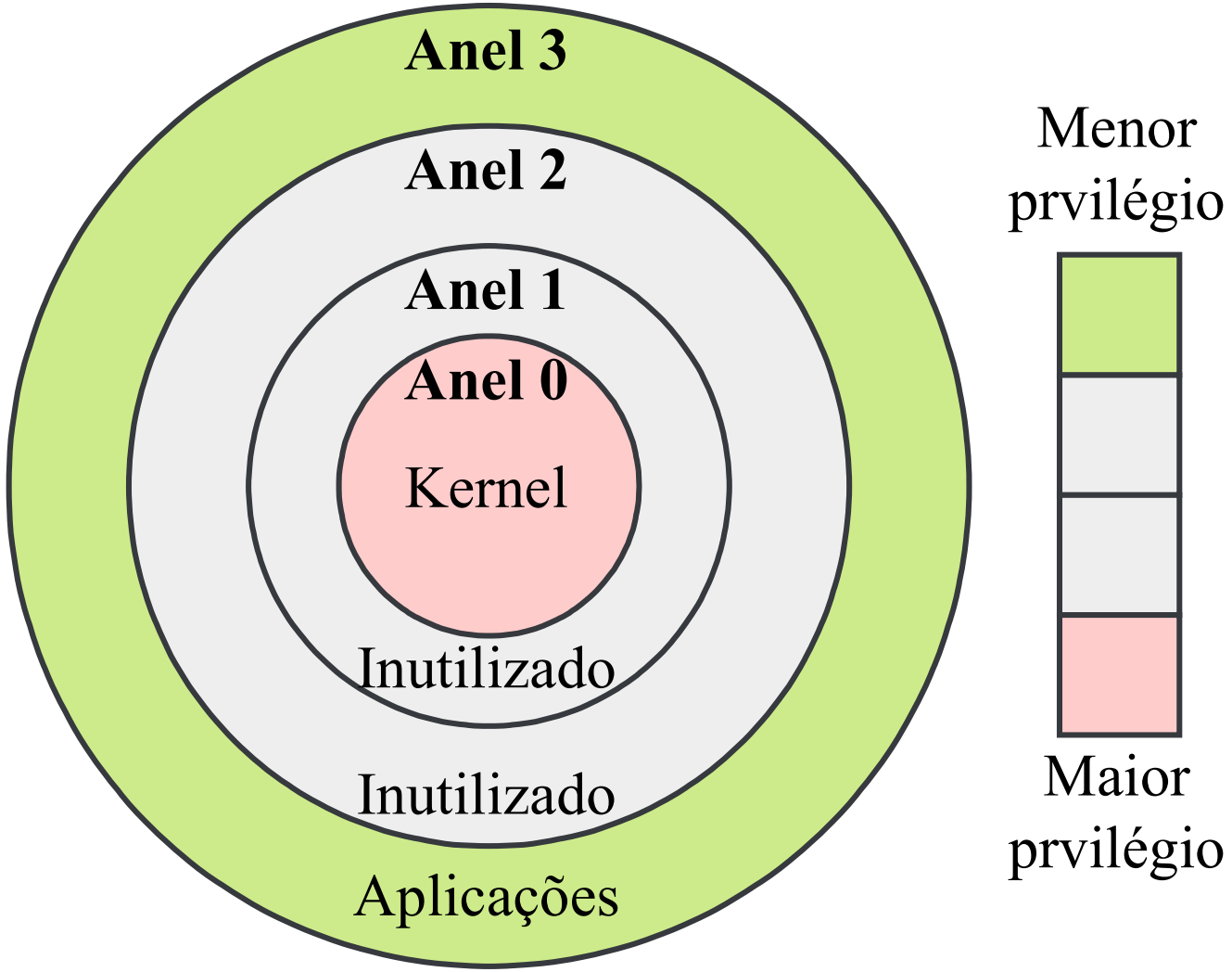


Estrutura de um SO

- Assim, as aplicações executam tanto em modo **usuário** (*userspace*) ou em modo **kernel** (*kernel space*).

- Kernel:
  - Controle total do hardware

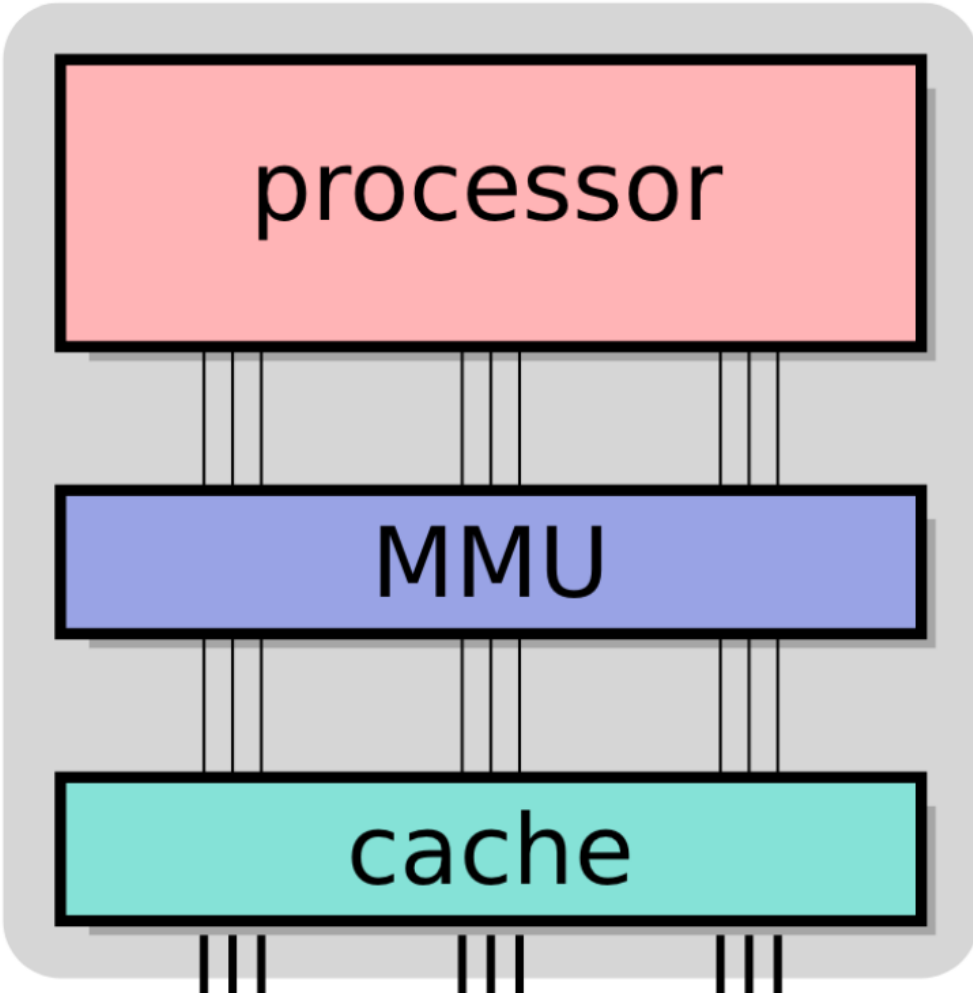
- Usuário:
  - Subconjunto de instruções de kernel.
  - Execução de instruções “perigosas” geram interrupções



Estrutura de um SO

- A unidade de gerência de memória (MMU) cria áreas exclusivas de memória para as aplicações em execução (MMU opera em modo **kernel**)

- Ela permite com que aplicações possuam diferentes espaços de memória, uma não interferindo na execução da outra. Permanecem **isoladas** umas das outras.



Estrutura de um SO

- Devido aos níveis de privilégio e a MMU, não temos como acessar as rotinas do SO diretamente

- Recorremos à **chamadas de sistema** para nos comunicarmos diretamente com o núcleo do SO

- Essas chamadas são **funções** implementadas pelo próprio SO para que aplicações no **nível do usuário** interaja com ele
  - Acessar arquivos
  - Alocar memória
  - Acessar periféricos

Estrutura de um SO

- A lista de chamadas de sistema varia de um sistema para outro, fornecendo ao usuário uma Application Programming Interface (API)

- Elas podem ser divididas em grandes áreas:
  - Gestão de processos
  - Gestão da memória
  - Gestão de arquivos
  - Comunicação
  - Gestão de dispositivos
  - Gestão do sistema

Atividade

- Configurar máquina virtual
- Utilizar o strace para visualizar as chamadas de sistema
- Utilizar o ltrace para visualizar as chamadas de biblioteca

Referências

1. MAZIERO, Carlos A. **Sistemas operacionais: conceitos e mecanismos**. [s.l.]: Ufpr, 2019.
2. TANENBAUM, Andrew S.; BOS, Herbert. **Modern operating systems**. 4. ed. Boston: Prentice Hall, 2015.
3. LINUX**. Introduction – The Linux Kernel Documentation.** c2024**.** Disponível em: https://linux-kernel-labs.github.io/refs/heads/master/lectures/intro.html. Acesso em: 02 de abril de 2024.

Contato: p.horchulhack@pucpr.br