Aspectos de Gerenciamento de Dados

Memória Virtual e Arquivos

Engenharia de Computação
Pontifícia Universidade Católica de Campinas

Prof. Dr. Denis M. L. Martins

Objetivos de Aprendizado

Ao final desta aula, você será capaz de:

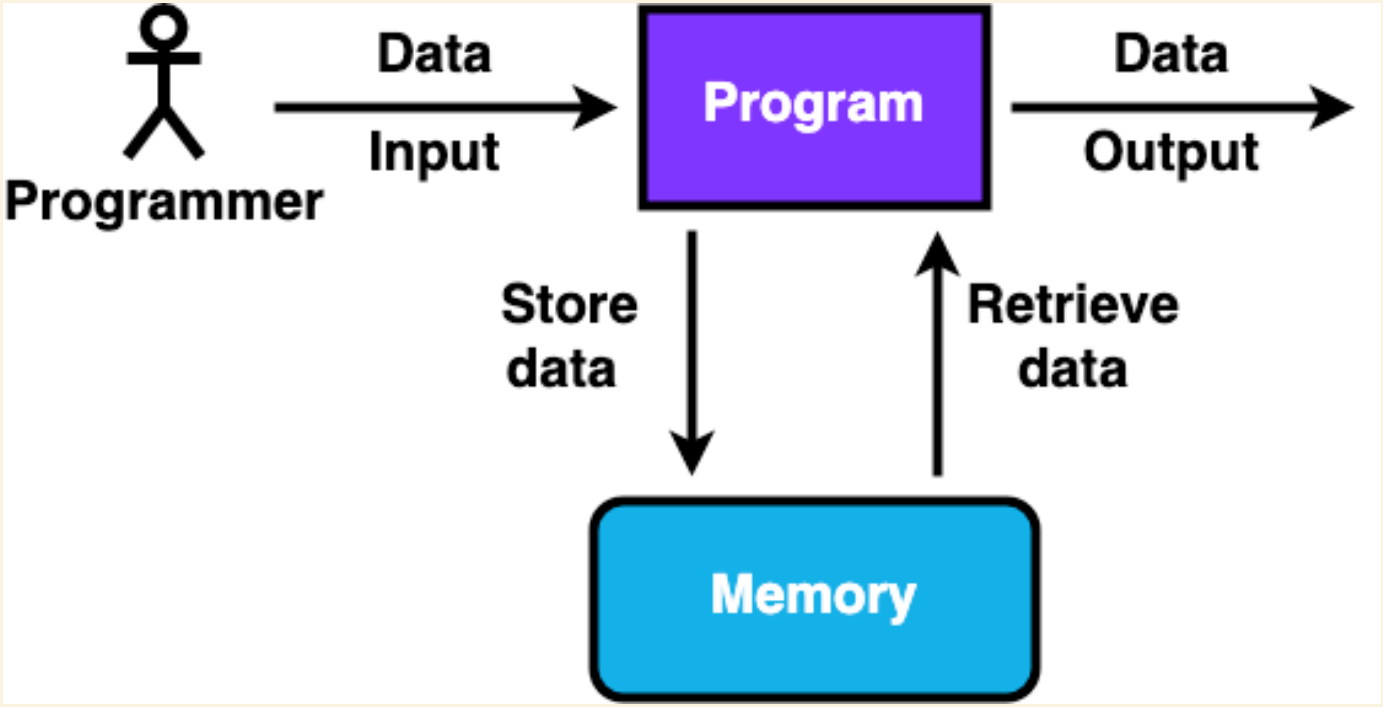
- Compreender abstrações de memória e arquivos em sistemas operacionais
- Entender o conceito e o funcionamento de memória virtual
- Descrever componentes-chave de sistemas de arquivos
- Aplicar permissões de arquivo

Disclaimer

Parte do material apresentado a seguir foi adaptado de:

- IT Systems – Open Educational Resource, produzido por Jens~Lechtenböger; e
- Open Education Hub - Operating Systems

Visão Geral



Fonte da Imagem: OER OS

Critérios para o SO

- Recuperar e armazenar o mais rápido possível
- Uso ótimo do espaço de memória
  - Quando os dados não estão sendo usados, a memória é liberada imediatamente
  - Minimizar o tempo em que a memória está reservada, mas não utilizada
  - Os dados devem ocupar o menor espaço necessário
- Segurança
  - Correção dos dados
  - Isolamento dos dados

Uma Perspectiva do Programador sobre os Dados

- Dados = variáveis
- Operações: declarar/ler/escrever
- Variáveis são armazenadas na memória, portanto, dependendo da linguagem, você também pode:
  - Alocar memória
  - Desalocar memória

Exemplo

```
class Pokemon:
    def __init__(self, id, name):
        self.id = id
        self.name = name

    def __repr__(self):
        return "[id: " + str(self.id) + ", name: " + self.name + "]"

    def __del__(self):
        print("destroying " + self.name)

pokedex = []
pokedex.append(Pokemon(1, "Pikachu"))
pokedex.append(Pokemon(2, "Charizard"))
# ...
print(pokedex)
```

Desempenho: Depende de...

- Número de cópias de memória

  `for i in range(1000000): print(pokemon[i])`

- Grau de reutilização da memória
- Número de alocações/desalocações de memória

```
arr->ptr = realloc(arr->ptr, sizeof(int)*(arr->len + 1));
arr->ptr[arr->len] = elem;
arr->len++;
```

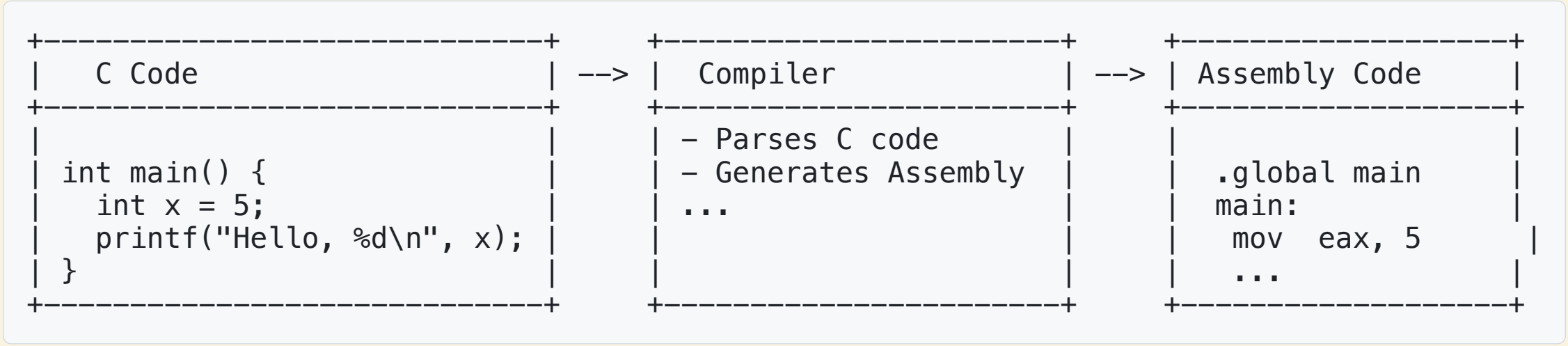
Uso de Espaço: Depende de...

- Como armazenamos os dados
- Uso tipos apropriados para as variáveis
- Quão cedo a memória é liberada

Quem gerencia a memória?

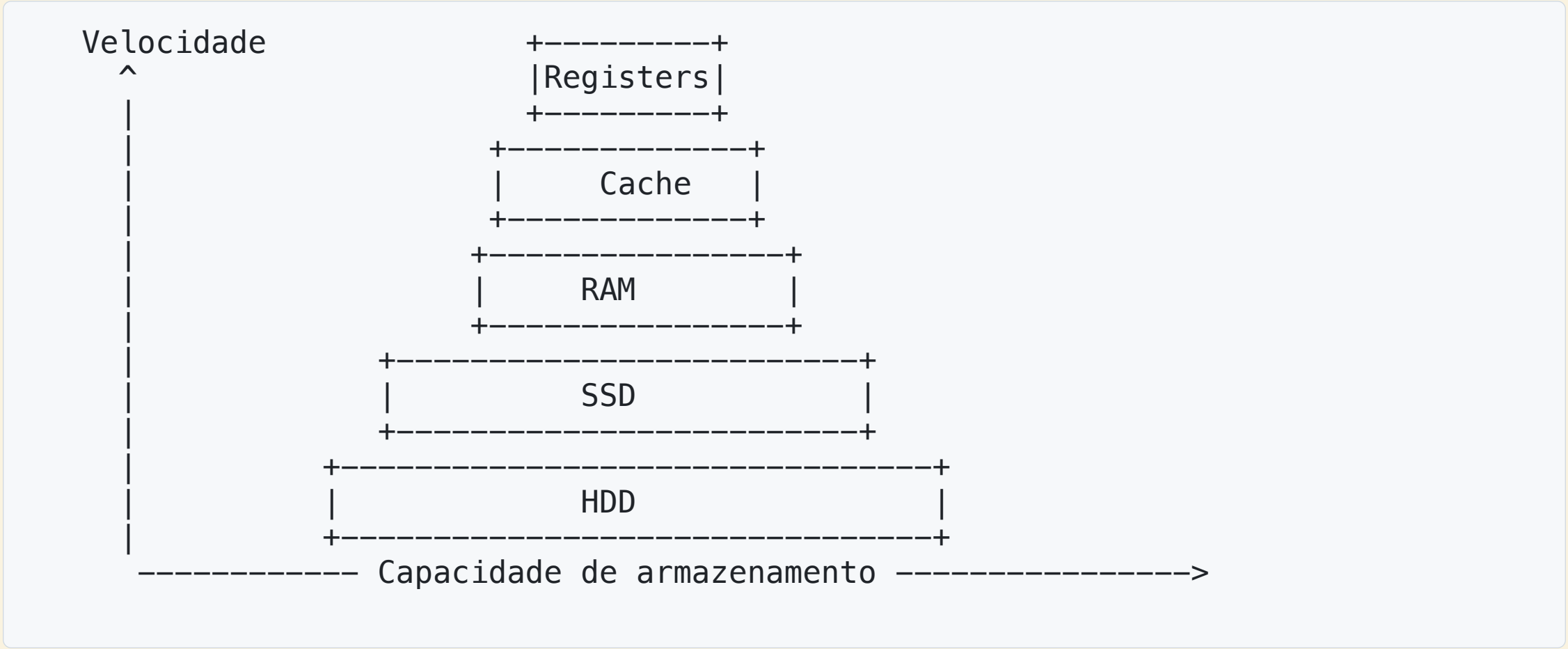
- Você (o programador) - C/C++
- A linguagem de programação - Python, Java
- Uma implementação de biblioteca - C/C++
- O sistema operacional - para todas as linguagens

Perspectiva da Linguagem de Programação



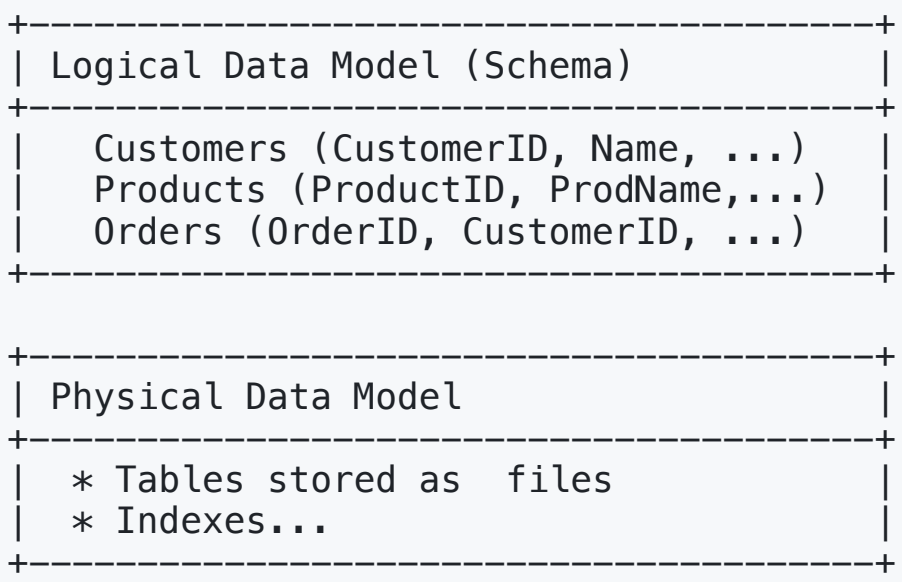
Perspectiva do Hardware

Hierarquia de memória: como dados são mapeados nessa hierarquia?



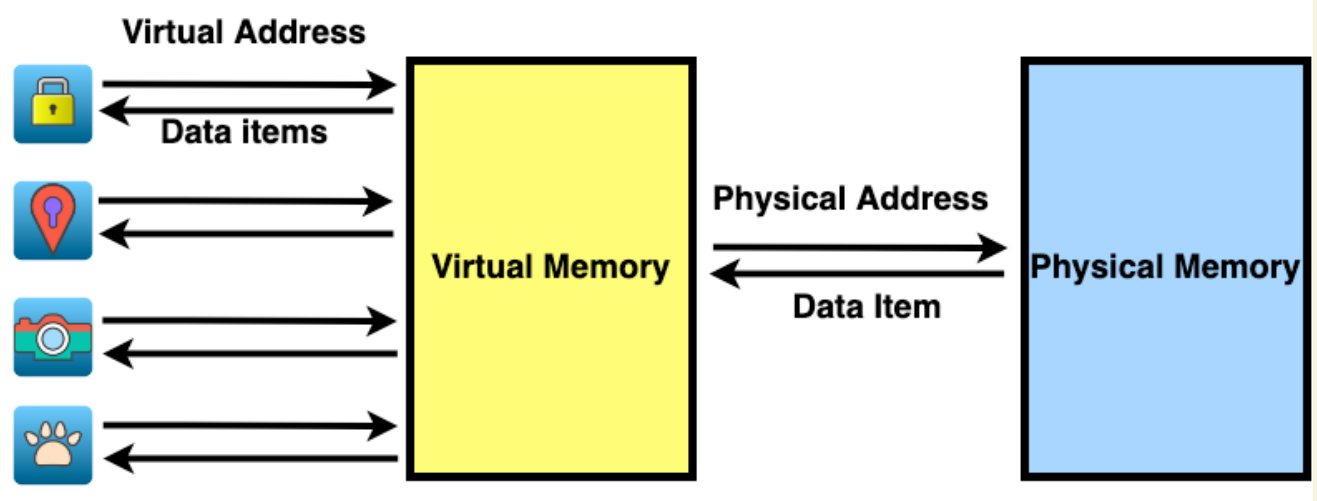
Exemplo: Bancos de Dados

- Memória principal armazena *database buffer* + código de aplicações e do SGBD
- Memória segundária armazena arquivos "físicos" do BD.
- Movimentação de dados entre memória secundária e memória princial é chamada de I/O (input/output) $\longrightarrow$ Supervisionada pelo SO



Memória Virtual

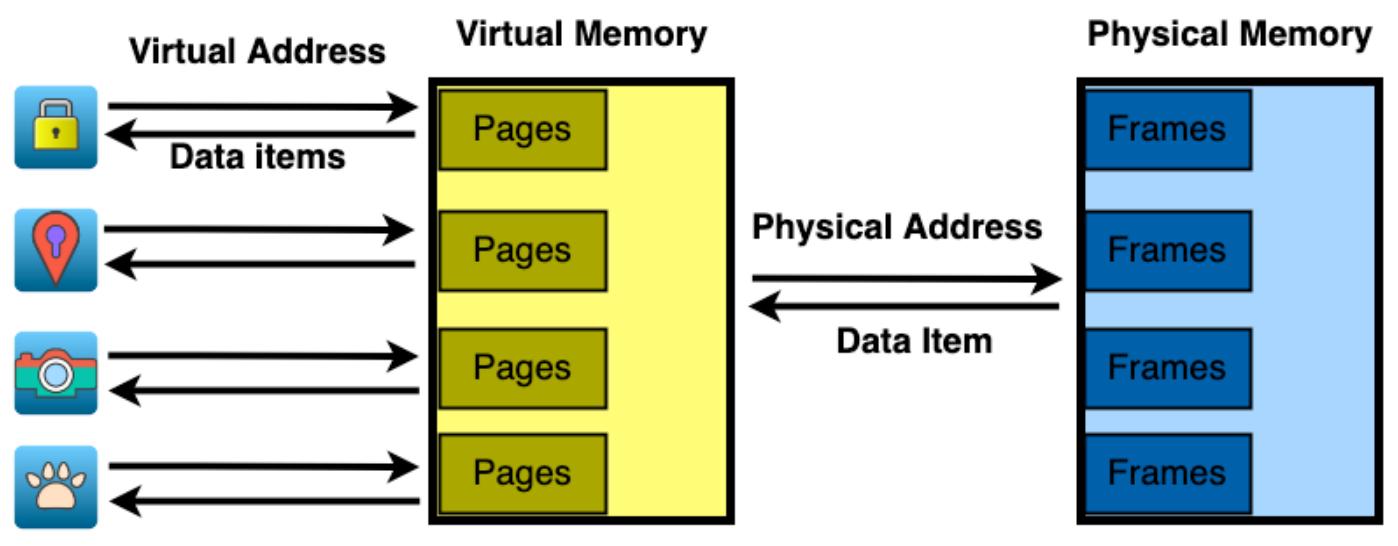
- Os processos trabalham com memória virtual como se fosse memória física
- Cada processo pode endereçar todo o espaço de endereço (potencialmente, mais do que o tamanho da RAM)
- Cada processo é encapsulado (não tem acesso à memória de outro processo)
- Penalidades: hardware e software adicionais necessários, tempo extra gasto ao acessar a memória



Fonte da Imagem: OER OS

Páginas/Frames

- Uma página é uma unidade de memória virtual
- Um frame é uma unidade de memória física
- Um tamanho de página/frame é tipicamente 4 KB
- Páginas virtuais são mapeadas para frames físicos
- O mapeamento é armazenado em uma tabela de páginas



Fonte da Imagem: OER OS

Interface do Sistema Operacional com Memória Virtual

- `/proc/<PID>/mem` - acesso à memória virtual
  - `demo/proc_mem/`
- `/proc/<PID>/page_map` - acesso aos mapeamentos de página
  - `demo/proc_pagemap/`
- `/dev/mem` - acesso à memória física
  - `demo/proc_pagemap/`

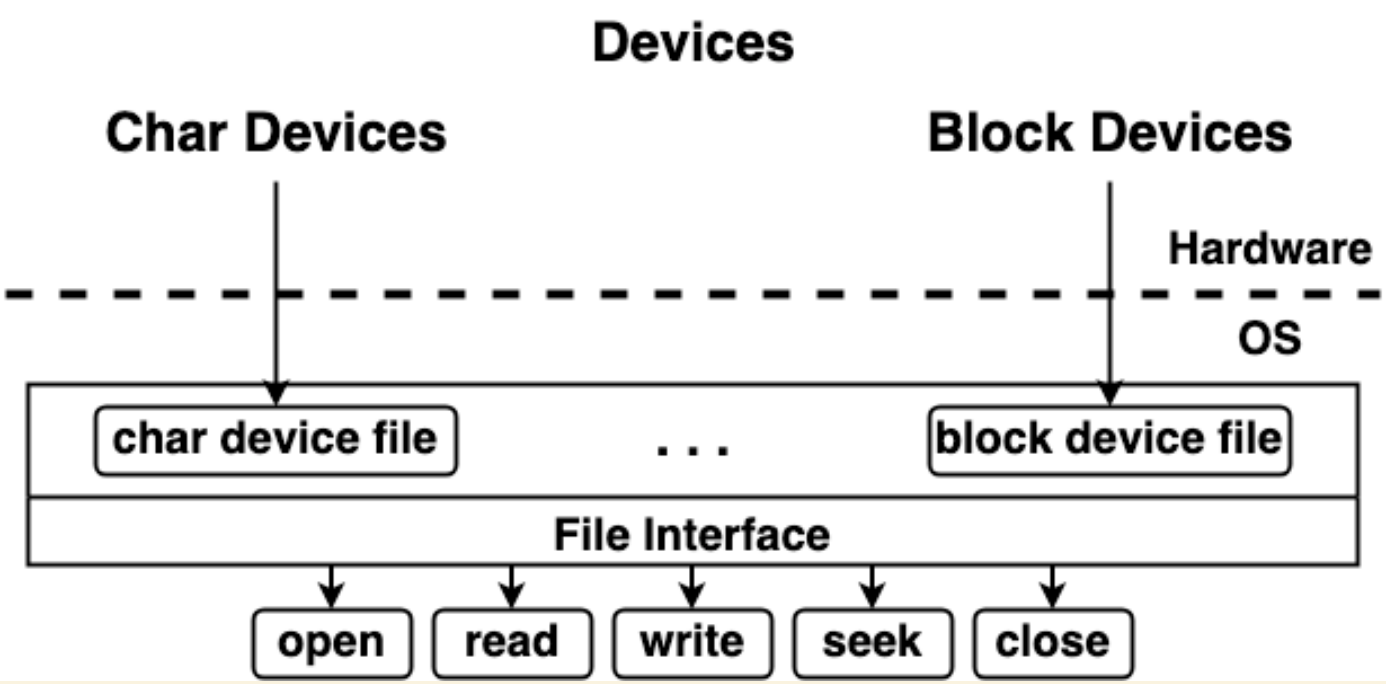
Sistemas de Arquivos

Arquivos

- Arquivos são abstrações comuns do SO para organizar dados e armazená-los de maneira persistente.
  - Persistência: SO deve manter dados mesmo que haja cortes de energia ou falhas no sistema.
- *File Descriptors*: SO representa arquivos via números inteiros chamados *descriptors*
  - Arquivos: *named streams of bytes*
  - Abstração: arquivos, diretórios, dispositivos de I/O, acesso de rede, etc.
- Operações: `open`, `close`, `read`, `write`

Representação de Dispositivos de I/O no Linux

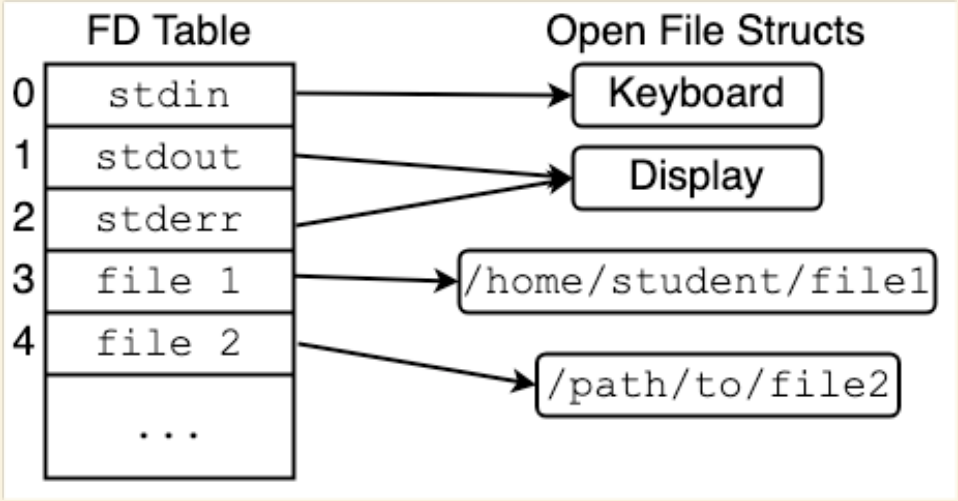
Linux abstrai um dispositivo de I/O como um arquivo especial (diretório `/dev`)



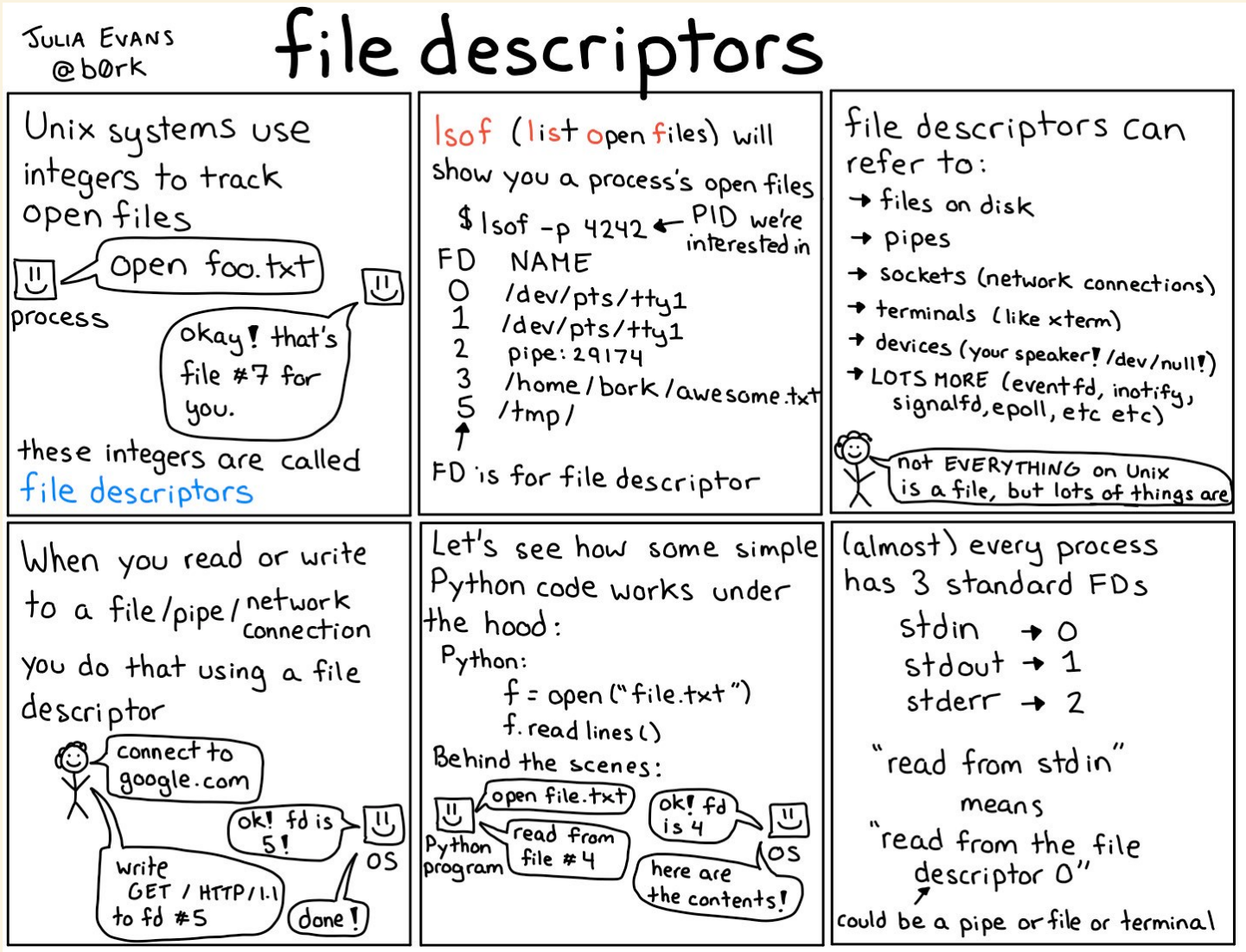
Fonte da Imagem: OS Team - OS OER

Arquivos

- O padrão POSIX descreve 3 descritores (numerados 0, 1, 2) para cada processo:
  - `0` : *Standard input*, `stdin` (e.g., entrada do teclado)
  - `1` : *Standard output*, `stdout` (e.g., imprimir na tela ou no terminal)
  - `2` : *Standard error*, `stderr` (e.g., imprimir mensagem de erro no terminal)



Fonte da Imagem: OS Team - OS OER



Fonte da Imagem: Julia Evans

Redirecionamento de Fluxos (Streams)

- Fluxos de bytes podem ser redirecionados:
- Por exemplo, enviar a saída para um arquivo em vez do terminal: `head names.txt > first10names.txt`
  - Relembre Command Line Murders
- O programa `head` invoca chamadas de sistema:
  - Abre o arquivo `names.txt`, resultando em um descritor de arquivo recém-alocado.
  - Lê do descritor de arquivo para `names.txt`.
  - Escreve no fluxo padrão de saída (stdout), que é aberto automaticamente por padrão.
- Operador `>` redireciona o stdout do processo para um arquivo: `first10names.txt`.
- Além disso, muitos comandos podem acessar dados em stdin: `head < names.txt`
- Operador `<` redireciona um arquivo para o stdin do processo: acesso ao `names.txt` via stdin.

Descritores de Arquivo em `/proc/<pid>/fd`

Para um processo com ID `<pid>`, o subdiretório `/proc/<pid>/fd` indica seus descritores de arquivo.

- As entradas são links simbólicos que apontam para os destinos reais.
- Use `ls -l` para ver os números e seus destinos, por exemplo:

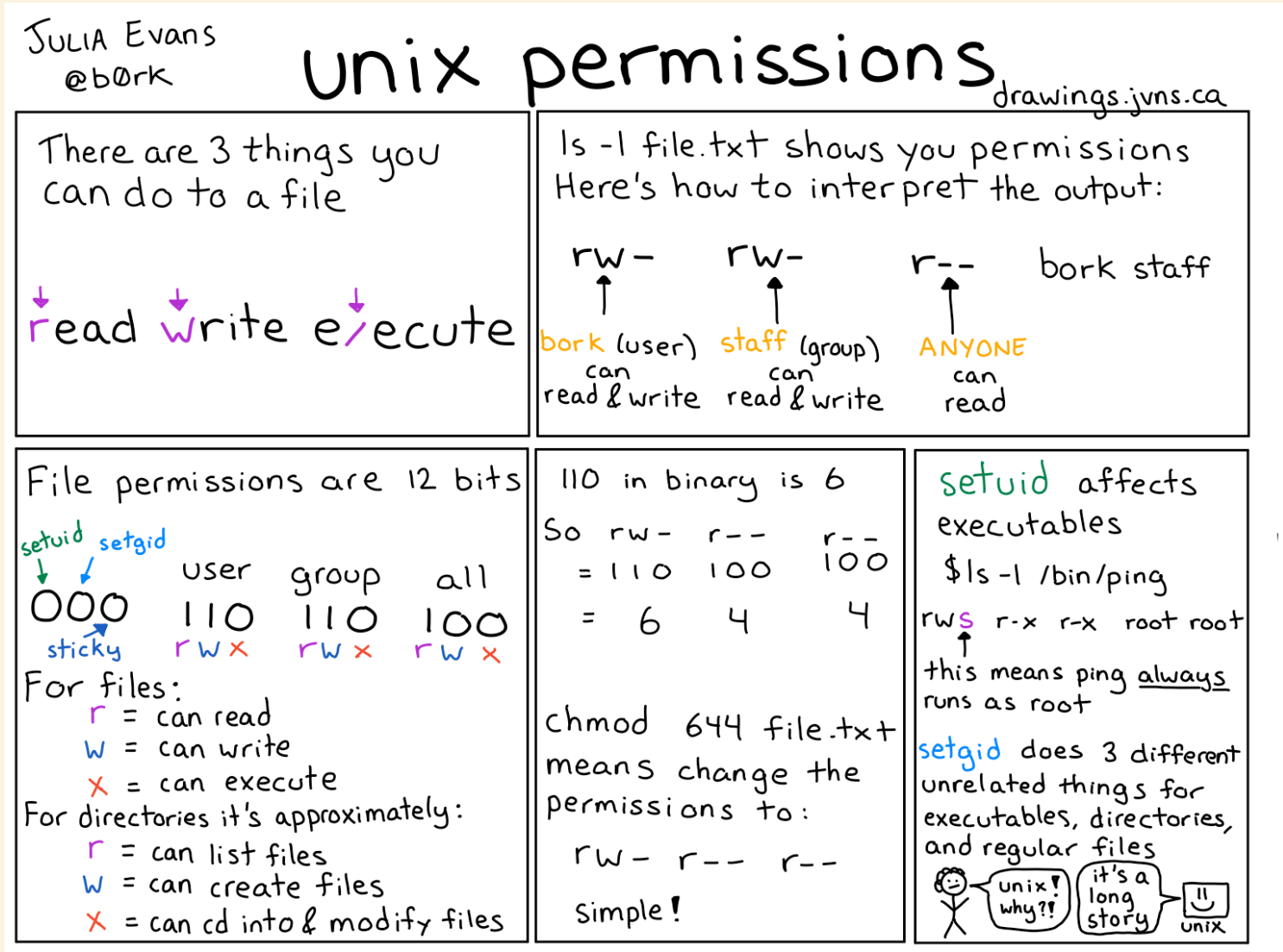
```
lrwx------ 1 root root 64 Jun 26 15:34 0 -> /dev/pts/3
lrwx------ 1 root root 64 Jun 26 15:34 1 -> /dev/pts/3
lrwx------ 1 root root 64 Jun 26 15:34 2 -> /dev/pts/3
lr-x------ 1 root root 64 Jun 26 15:34 3 -> /dev/tty
lr-x------ 1 root root 64 Jun 26 15:34 4 -> /etc/passwd
```

Permissões de Acesso

- Quem tem permissão para fazer o quê?
- O sistema controla o acesso a objetos por sujeitos.
- **Objeto:** qualquer coisa que precise ser protegida: por exemplo, uma região de memória, um arquivo, um serviço.
  - Com operações diferentes dependendo do tipo de objeto.
- **Sujeito:** entidade ativa que utiliza os objetos, ou seja, um processo.
  - Threads dentro de um processo compartilham as mesmas permissões de acesso.
  - O sujeito pode também ser o próprio objeto, por exemplo, terminar uma thread ou um processo.

Permissões de Acesso (cont.)

- O sujeito age em nome de um principal.
- ***Principal***: Usuário ou unidade organizacional.
- Diferentes principais e sujeitos têm permissões de acesso diferentes sobre objetos diferentes.



Fonte da Imagem: Julia Evans

Controles de Acesso a Arquivos

- ***Access Control List*** (ACL): lista de permissão de acesso associados a objetos.
- O comando `ls` lista arquivos e diretórios.
  - Com a opção `-l` em formato "longo" (detalhado).
- Os ACLs não são para usuários individuais; em vez disso, são definidos separadamente para o proprietário, grupo e outros.
  - **Owner:** Inicialmente, o criador; a propriedade pode ser transferida.
  - **Group:** Usuários podem ser agrupados, por exemplo, para compartilhar arquivos em um projeto conjunto.
  - **Other:** Todos os demais.

Controles de Acesso a Arquivos (cont.)

- Um tipo de arquivo, seguido por três triplas com permissões:
  - Arquivo (`-`), diretório (`d`), link simbólico (`l`), etc.
  - Leitura (`r`), escrita (`w`), execução (`x`) (para diretórios, "execução" significa "travar").
  - Definir ID de usuário/grupo (`s`), *sticky bit* (`t`).

```
> ls -l /etc/shadow /usr/bin/passwd

- rw- r-- ---  1  root  shadow   2206 Jan 11  2024 /etc/shadow
- rws r-x r-x  1  root  root    68208 Feb  6 13:49 /usr/bin/passwd*
```

Gerenciamento de ACLs

- **Gerenciamento de ACLs com chmod**
  - Consulte a página do manual: `man chmod`
- **Permissões através de padrão binário ou simbólico**
  - O desenho anterior ilustra os padrões binários para r, w, x
  - As especificações simbólicas contêm
    - entre outros, `u`, `g`, `o` para usuário, grupo, outros, respectivamente,
    - seguido de `+` ou `-` para adicionar ou remover uma permissão,
    - seguido de um de r, w, x, s, t (e mais)
  - Por exemplo: `chmod g+w file.txt` adiciona permissões de escrita para membros do grupo no arquivo `file.txt`.

Dúvidas e Discussão