

# Programação Estruturada

2024-Q2

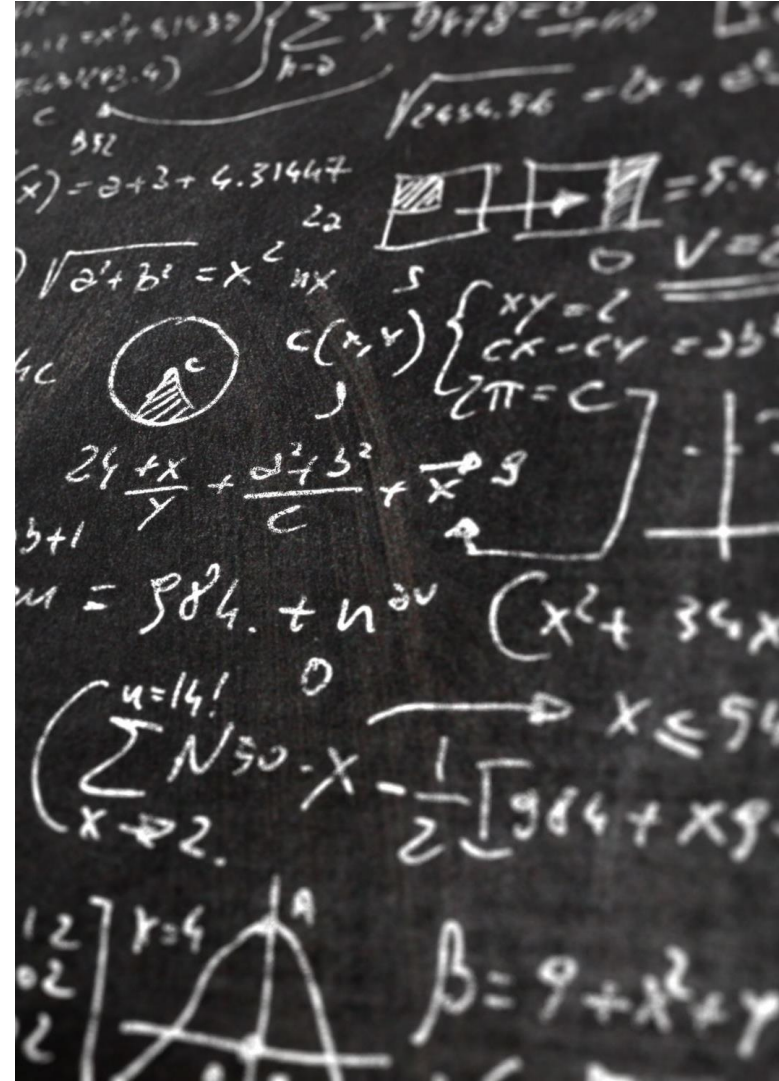
---

PROF. MAYCON SAMBINELLI

[HTTP://PROFESSOR.UFABC.EDU.BR/~M.SAMBINELLI](http://professor.ufabc.edu.br/~m.sambinelli)

# Objetivos

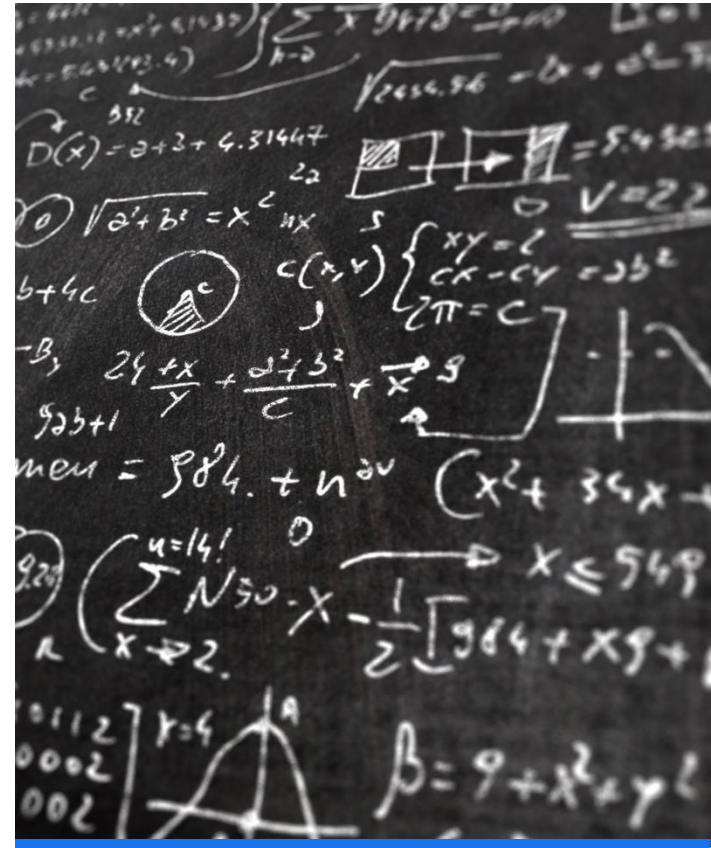
*“Apresentar noções básicas e intermediárias sobre algoritmos, programação em linguagens compiladas, compilação, programas em execução (processos), ponteiros, alocação estática e dinâmica de memória, vetores e matrizes, funções e passagem de parâmetros, registros, arquivos e recursividade. Aplicar todos os conceitos apresentados no contexto da resolução de problemas clássicos e novos da computação.”*



# Tradução dos Objetivos Oficiais

Ao final do curso, espera-se que você seja capaz:

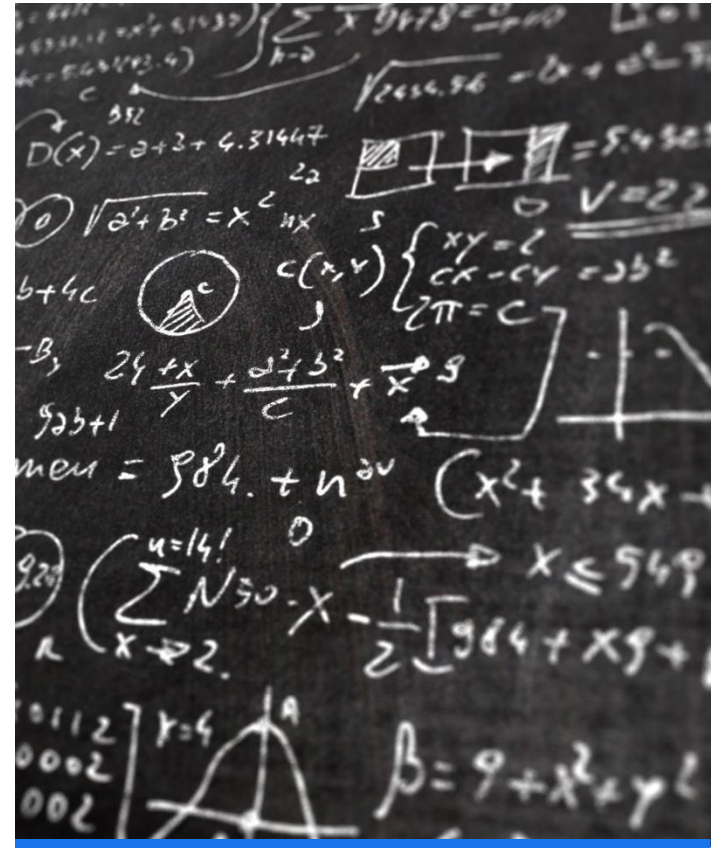
- Programar em C
- Gerenciar memória/Usar ponteiros
- Trabalhar com linguagem fortemente tipada
- Encontrar erros em programas defeituosos
  - Estratégias para isolar o problema
  - Usar um depurador (gdb, lldb, vscode)
  - Compreender mensagens de erro
- Compreender os artefatos das linguagens compiladas
- Saber se virar com o seu programa!



# Objetivos Secundários do Prof.

Ao final do curso, espera-se que você seja capaz:

- Usar o terminal de um sistema Unix-like
- Fazer versionamento de código com git
- Compilar o programa na linha de comando
- Depurar o programa usando uma ferramenta simples (gdb/lldb)



# Página do Curso

URL: <http://professor.ufabc.edu.br/~m.sambinelli/courses/2024Q3-PE/index.html>

Você encontrará:

- Avisos
- Horários e salas de aula
- Material didático
- Critério de avaliação
- Datas importantes



## MCTA028-15 – Programação Estruturada

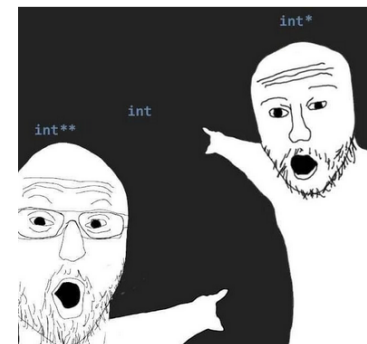
Terceiro Quadrimestre de 2024

**Turma(s):** Diurno

**Professor:** Maycon Sambinelli

**E-mail:** [m.sambinelli@ufabc.edu.br](mailto:m.sambinelli@ufabc.edu.br)

 [Caixa de Sugestões](#)



### Avisos importantes

- 29/09 - Página no ar.

### Objetivos

Apresentar noções básicas e intermediárias sobre algoritmos, programação em linguagens compiladas, compilação, programas em execução (processos), ponteiros, alocação estática e


---

# Disclaimer

Slides baseados nos Slides dos professores: Carla Negri Lintzmayer e Emílio Francesquini





A photograph of a computer classroom. The room is filled with rows of desks and chairs. The chairs are dark-colored with a distinctive perforated pattern on the backrest. The desks are light-colored. The overall lighting is soft and even. The text "O que é um computador?" is overlaid on the left side of the image in a white, sans-serif font. A blue horizontal line is positioned below the text.

O que é um  
computador?

---



---

# O que é computar?

Segundo o dicionário *Priberam*:

**Computar** (*verbo transitivo*)

1. Faz o cômputo de
2. Calcular; orçar
3. Contar
4. [informática] processar através de computadores

**Origem etimológica:** latim *computo*, -are, calcular, contar.



# O que é um computador?

Segundo o dicionário *Michaelis*:

**Computador** (*substantivo masculino*)

1. Aquele ou aquilo que calcula baseado em valores digitais; calculador, calculista.
2. [Informática] Máquina destinada ao recebimento, armazenamento e/ou processamento de dados, em pequena ou grande escala, de forma rápida, conforme um programa específico; computador eletrônico.



Fotos de um computador moderno

---

# Fotos de um computador antigo



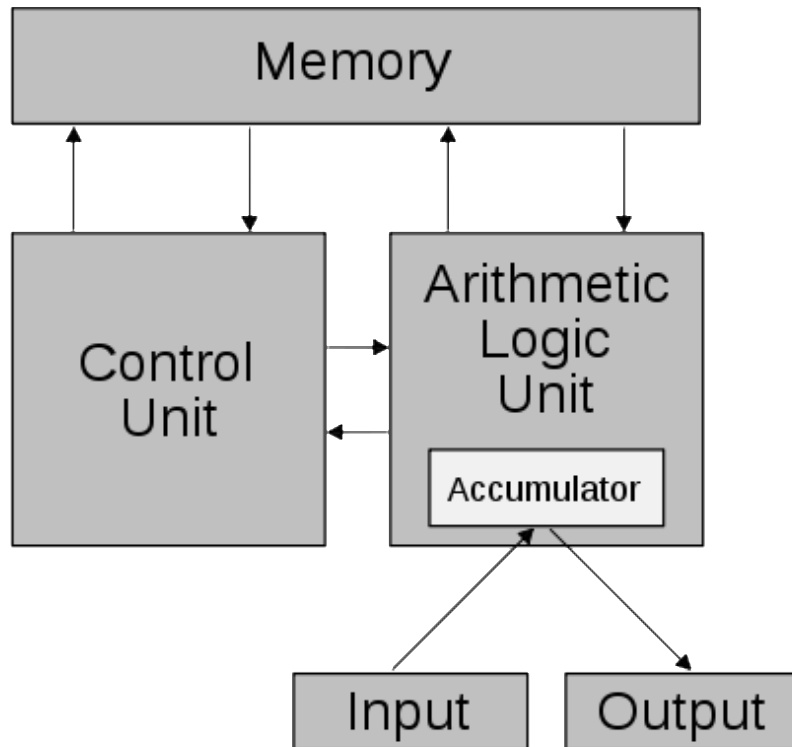
---

# Um computador antigo

- Durante a Segunda Guerra Mundial:
  - Aumento na necessidade de computação: principalmente cálculos balísticos
  - Computadores humanos
  - Formado principalmente por mulheres



# Hardware



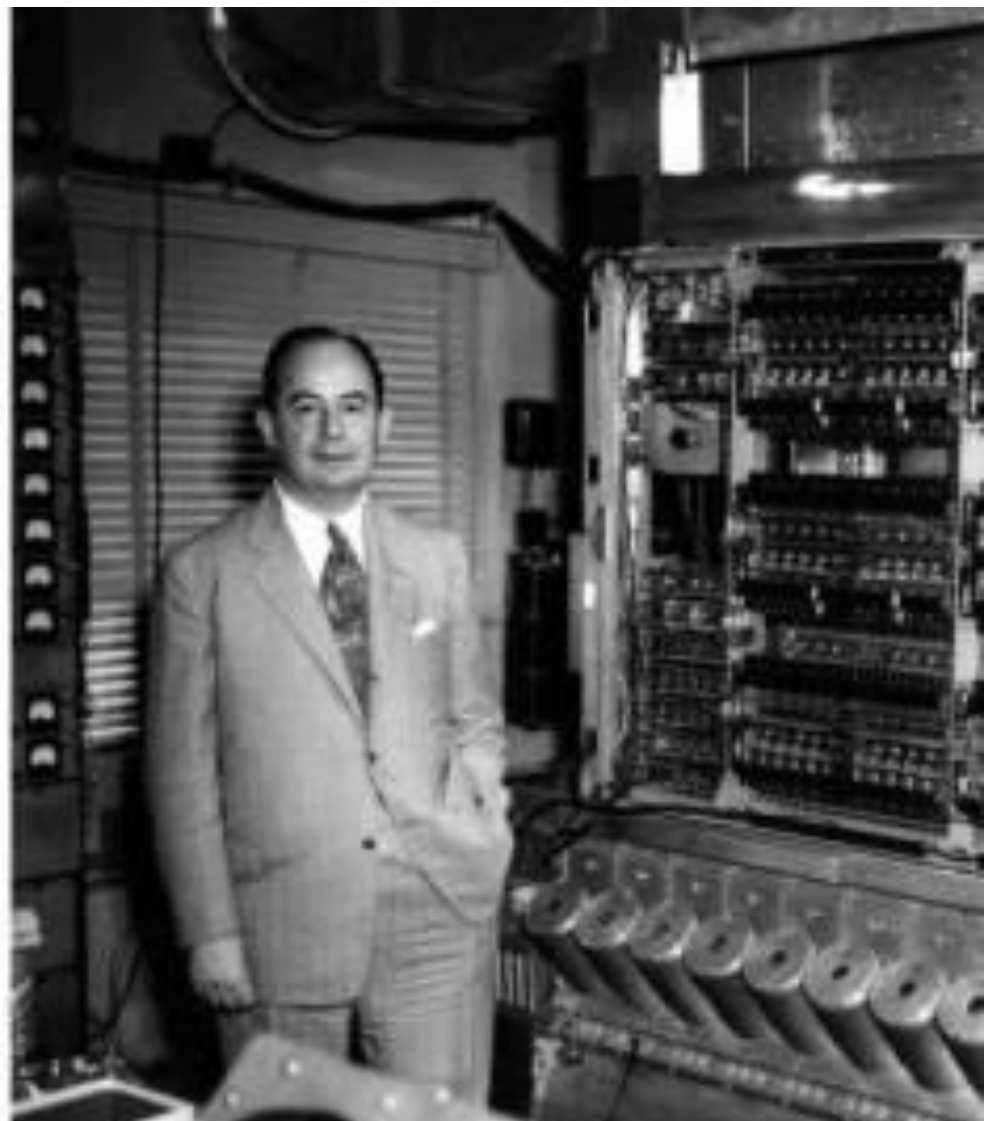
- **Hardware:** são todos os dispositivos físicos que compõem um computador, como CPU, disco rígido, memória, etc.
- Seguem uma organização básica como na figura (Arq. de Von Neumann)





# John von Neumann

- Matemático, Físico, Cientista da Computação e Engenheiro
- Muitas contribuições e em diversas áreas:
  - Arquitetura de Von Neumann
  - Merge-Sort
  - Contribuiu para a criação do ramo de Teoria dos Jogos
  - Muitas outras contribuições





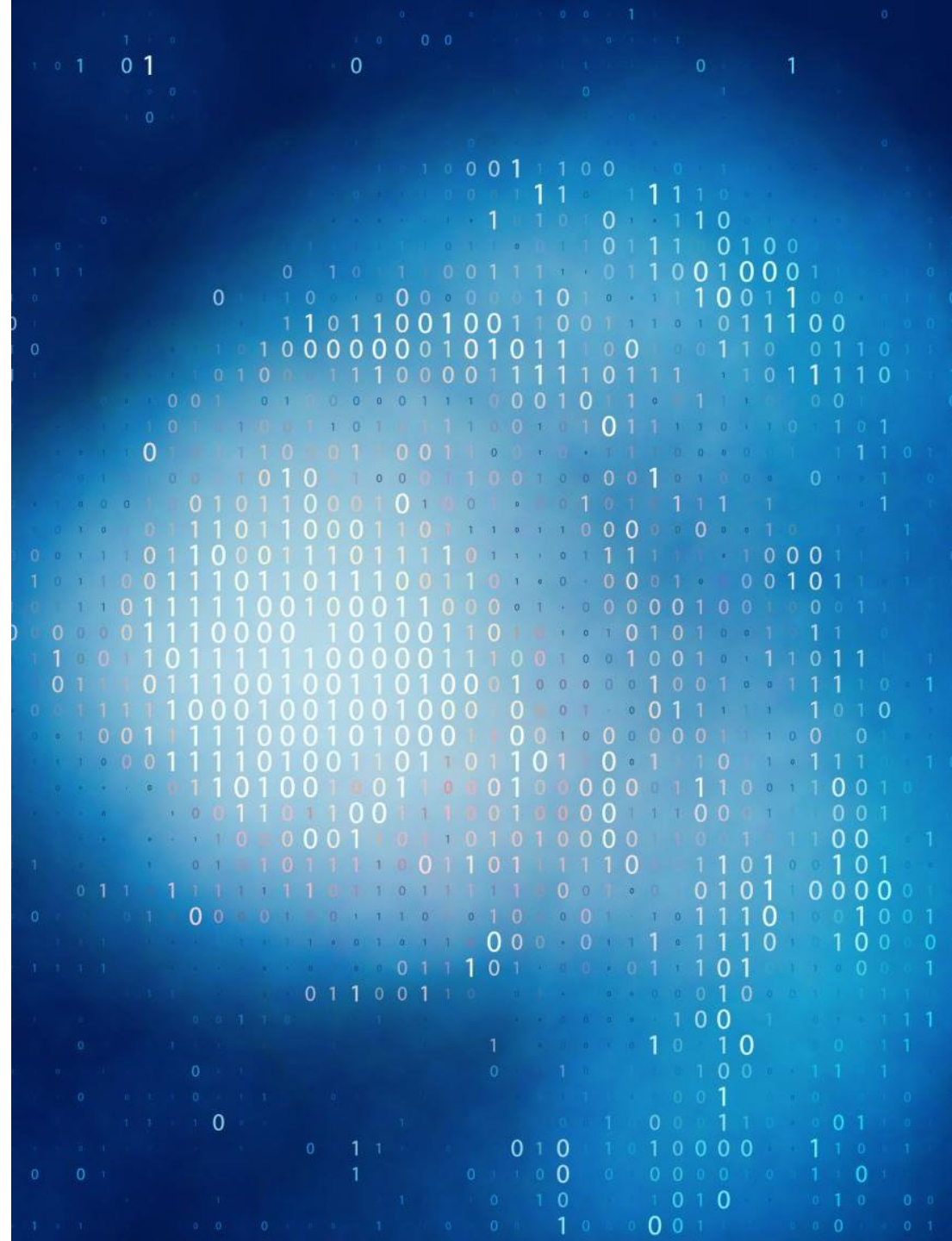
# Hardware e dispositivos

- Virtualmente todos os computadores atuais são digitais e operam com dois sinais: *sem energia* (0) e *com energia* (1)
- Chamamos estes sinais de **bit**  $\mapsto$  valores 0 ou 1
- Chamamos de **byte** um agrupamento de 8 bits
- Todas as informações armazenadas no computador são representadas por números 0s e 1s (letras, símbolos, imagens, programas, etc.)

  
\*Bytes

Nome	Simbolo	Tamanho
Byte	B	8 Bit
KiloByte	KB	1024 Byte
MegaByte	MB	1024 KByte
GigaByte	GB	1024 MByte
TeraByte	TB	1024 GByte
PetaByte	PB	1024 TByte
ExaByte	EB	1024 PByte
ZettaByte	ZB	1024 EByte
YottaByte	YB	1024 ZByte

# Software





# Software


**Softwares** são os programas que executam tarefas utilizando o hardware de um computador

- São compostos por um conjunto de instruções que operam o hardware
- Temos abaixo, por exemplo, três instruções para um computador de 32 bits

```
0100 0010 0011 0101 0101 0100 0011 0110
0100 1110 1100 1100 1001 0110 0110 1000
0000 0101 1111 1110 1101 0011 0000 1100
```

- Um software é composto por milhares de instruções deste tipo



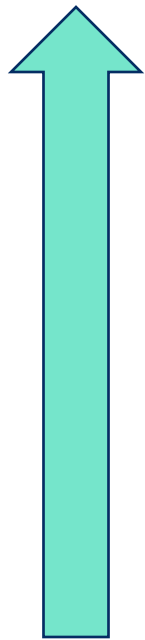


# Organização de um sistema computacional

---

---

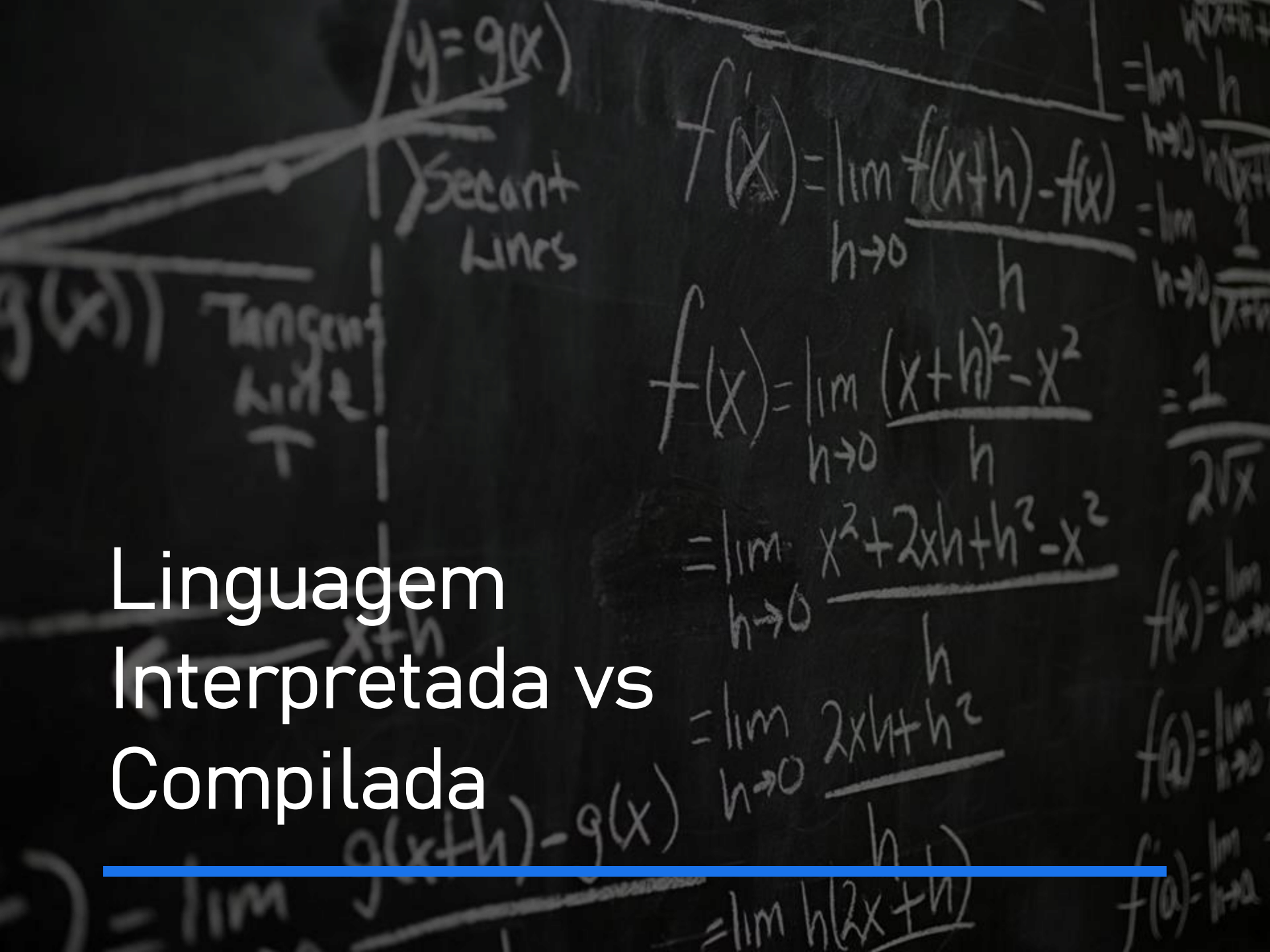
# Organização de um Sistema Computacional



- Aplicações: programas usados por usuários comuns (whatsapp, Firefox, Word)
- Compiladores/Interpretadores: programas usados para criar aplicações (gcc, python, clang)
- Sistema Operacional: programa que gerencia a máquina e coordena a execução das aplicações (Windows, Linux, FreeBSD, Android, IOS)

# Linguagem Interpretada vs Compilada

---



# Linguagem Interpretada

---



- Além do programa, o usuário precisa ter um interpretador instalado
- Geralmente são mais lentas
- Geralmente são mais amigáveis
- Exemplos: Python, Javascript, Julia, Lua

# Linguagem Interpretada

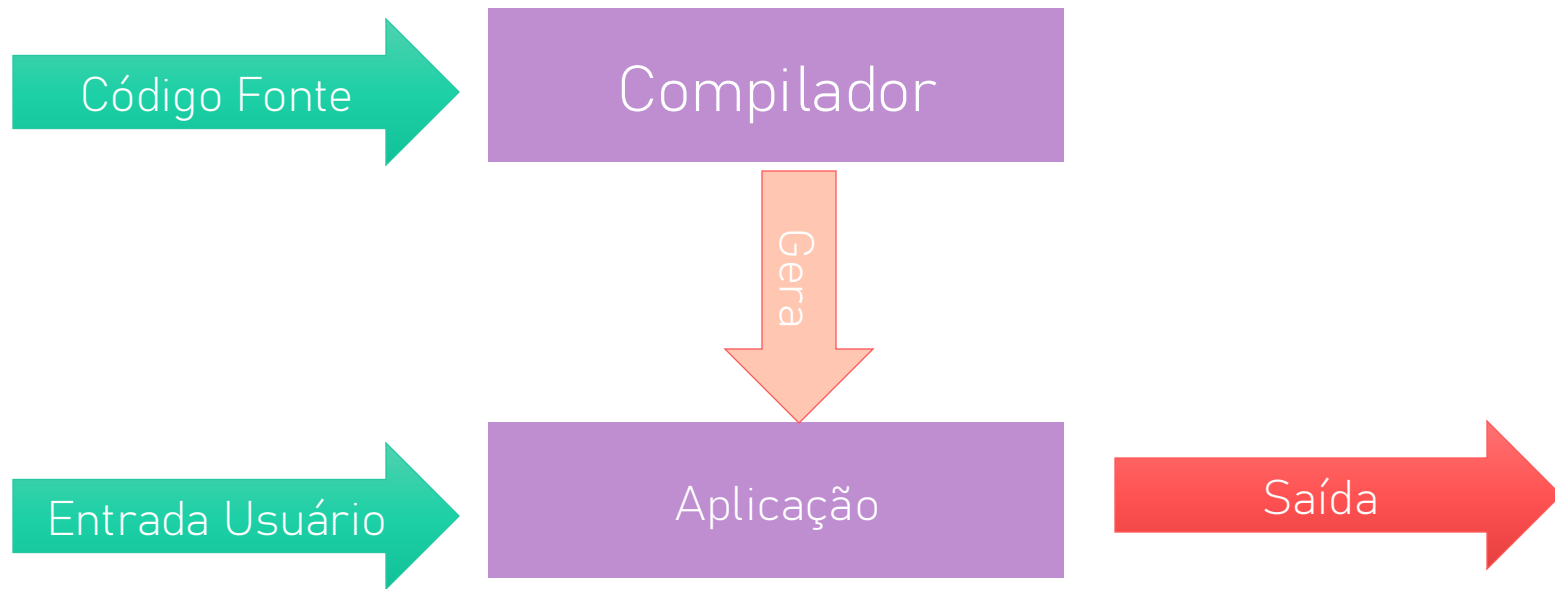
---



- Além do programa, o usuário precisa ter um interpretador instalado
- Geralmente são mais lentas
- Geralmente são mais amigáveis
- Exemplos: Python, Javascript, Julia, Lua



# Linguagem Compilada



- O usuário não precisa ter nada instalado em sua máquina, mas o compilador deve ser instruído sobre a arquitetura alvo (e SO)
- Geralmente mais rápidos
- Geralmente são mais burocráticas
- Exemplos: C, C++, Rust, Go, Fortran



# Hello World!

```
#include <stdio.h>

int main() {
    printf("Hello world!\n");
    return 0;
}
```

```
gcc hello.c -S -o hello.s
```



# Hello World!

```
#include <stdio.h>

int main() {
    printf("Hello world!\n");
    return 0;
}
```

```
gcc hello.c -S -o hello.s
```

```
global _main
extern _printf
section .text
_main:
    push message
    call _printf
    add esp, 4
    ret
message:
    db 'Hello, World', 10, 0
```

---

# Hello World!

```
global _main
extern _printf
section .text
_main:
    push message
    call _printf
    add esp, 4
    ret
message:
    db 'Hello, World', 10, 0
```

Hello.exe

```
21 0a 00 00
0c 10 00 06
6f 72 6c 64
08 10 00 06
6f 2c 20 57
04 10 00 06
48 65 6c 6c
00 10 00 06
00 10 00 06
10 00 00 00
01 00 00 00
04 00 00 00
80
01 00 00 00
80
```

# Hello World!

## Hexadecimal to Decimal to Binary Conversion Table



Hexadecimal Digit	Decimal Digit	Binary Digit
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Hello.exe

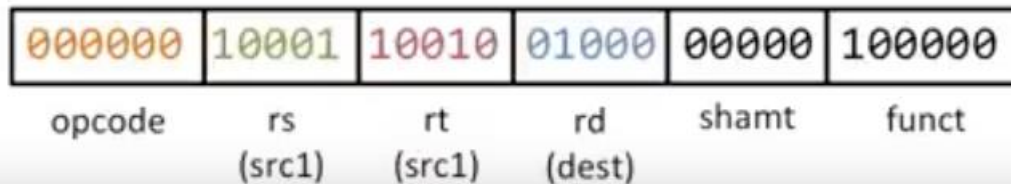
```
21 0a 00 00
0c 10 00 06
6f 72 6c 64
08 10 00 06
6f 2c 20 57
04 10 00 06
48 65 6c 6c
00 10 00 06
00 10 00 06
10 00 00 00
01 00 00 00
04 00 00 00
80
01 00 00 00
80
```

# Assembly e Opcodes

- **Example:**

add R8, R17, R18 →  
00000010 00110010 01000000 00100000

- MIPS instructions have logical fields:







# Linguagem C

- Criada em 1972 (52 Anos) por [Dennis Ritchie](#)
- Linguagem compilada de propósito geral
- Linguagem muito simples e que acredita no programador!
- Linguagem muito rápida
- Linguagem Bonita e muito influente
- Amplamente utilizada nos dias de hoje:
  - Ótima para sistemas embarcados (tem compilador para qualquer torradeira)
  - Muitos projetos antigos usam (Kernel Linux, GCC, Git, Apache, Openssh, PostgreSQL e etc)
- Uma linguagem que não esconde nada: treino bom para análise de algoritmos
- A linguagem que usaremos no curso!

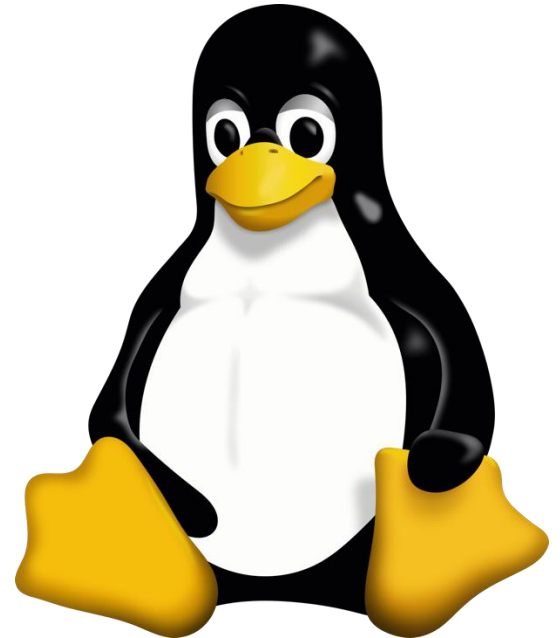


Linux

---

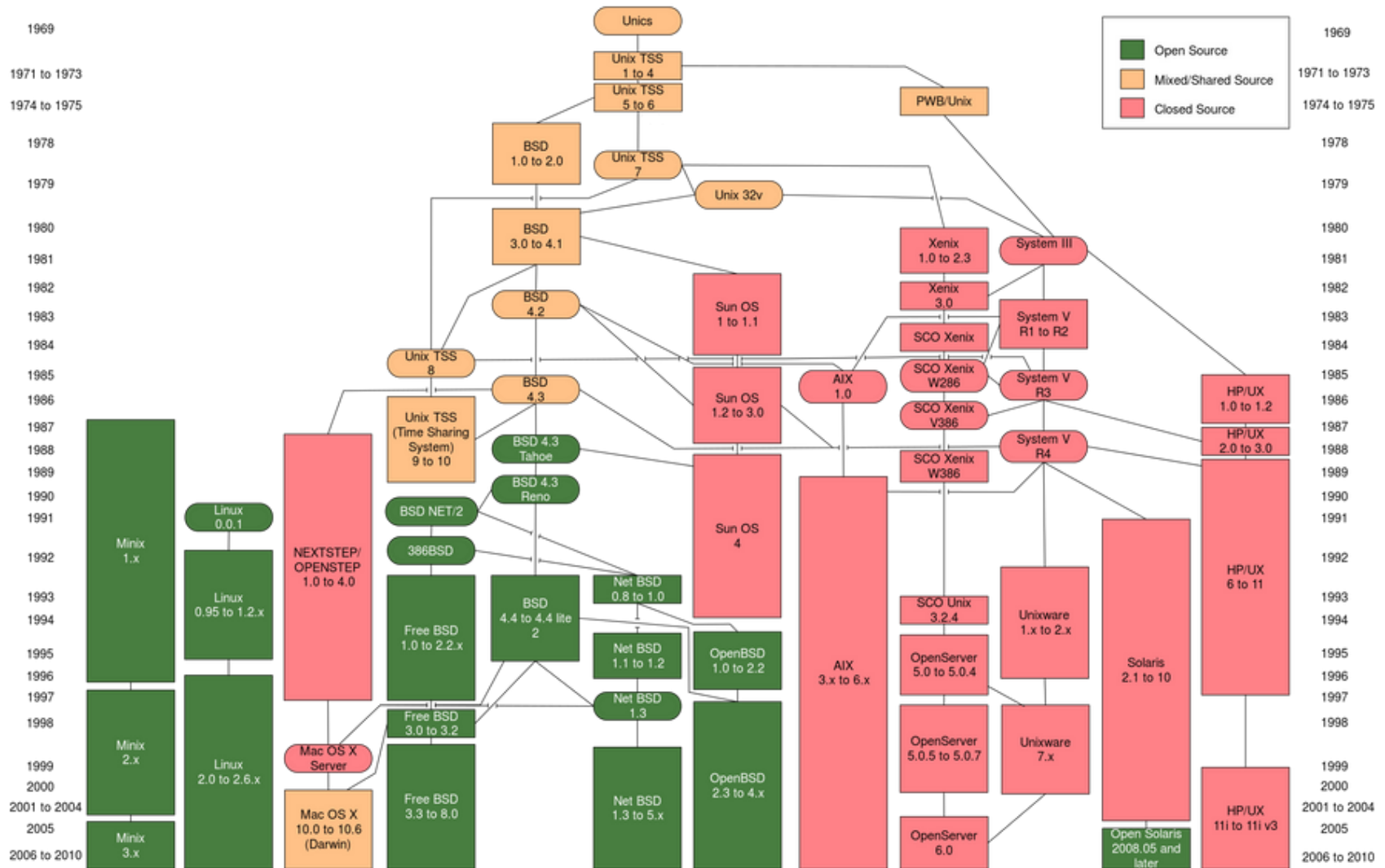
# Linux

- Sistema operacional criado por Linus Torvalds em 1991
  - Basicamente o sistema operacional utilizado para sustentar a Web
  - "Pai" do Android
  - SO baseado no sistema Unix

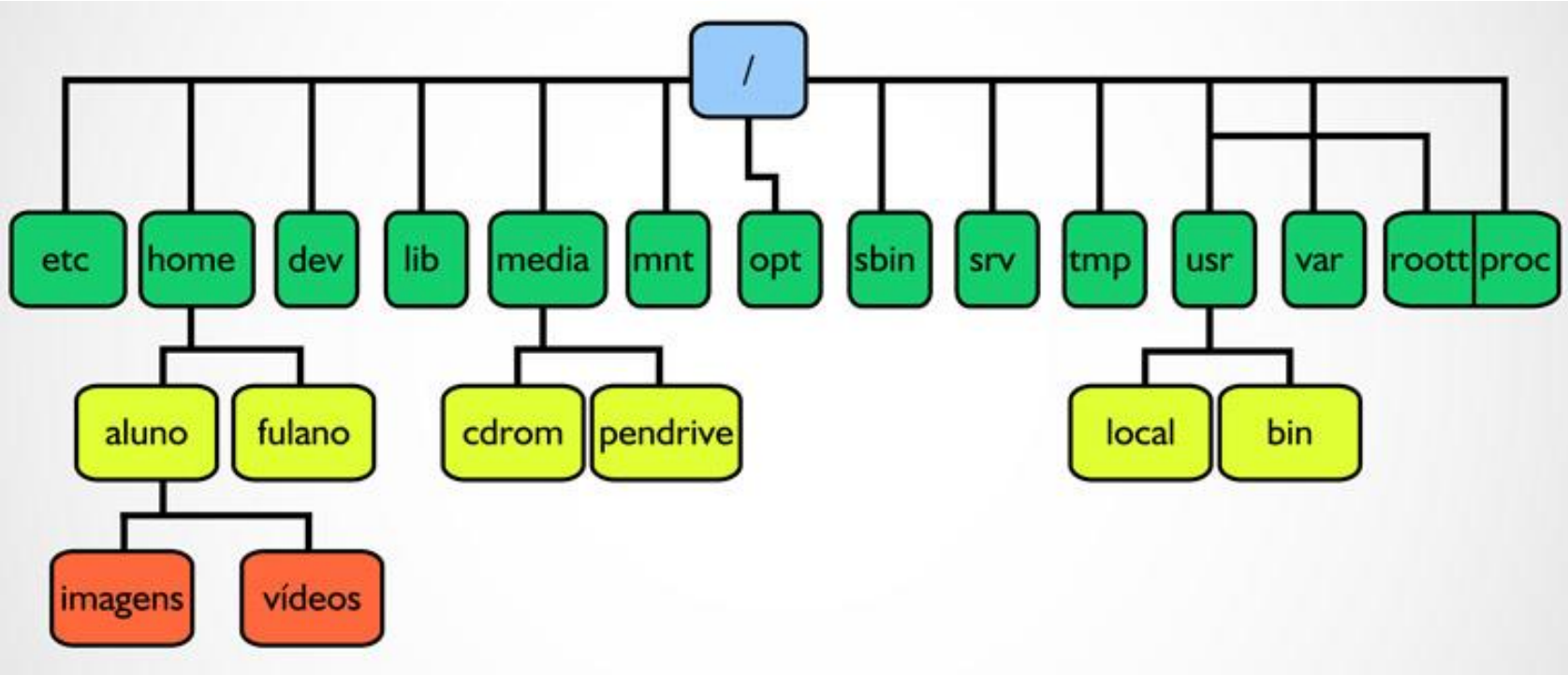


# UNIX HISTORY

1969 - 2010



# Estrutura de Diretórios





# Comandos

- Comandos essenciais: ls, pwd, cd, mkdir, cp, mv, rm, cat
- Use o tab para completar





# Roteiro

1. Criar pasta 'Projeto'
2. Criar pastas
  1. 'Projeto/lab01'
  2. 'Projeto/lab02'
  3. 'Projeto/lab03'
3. Crie o arquivo hello.c na pasta 'Projeto/lab01'
4. Copie o arquivo hello.c para a pasta 'Projeto/lab02'
5. Renomeie o arquivo hello.c da pasta 'Projeto/lab02' para 'ola.c'
6. Remova a pasta 'Projeto/lab02'