Streams

Prof. Fernando Figueira
(adaptado do material do Prof. Rafael Beserra Gomes)

UFRN

Material compilado em 1 de dezembro de 2025.

C trata os recursos de entrada e saída dos dados como **streams**

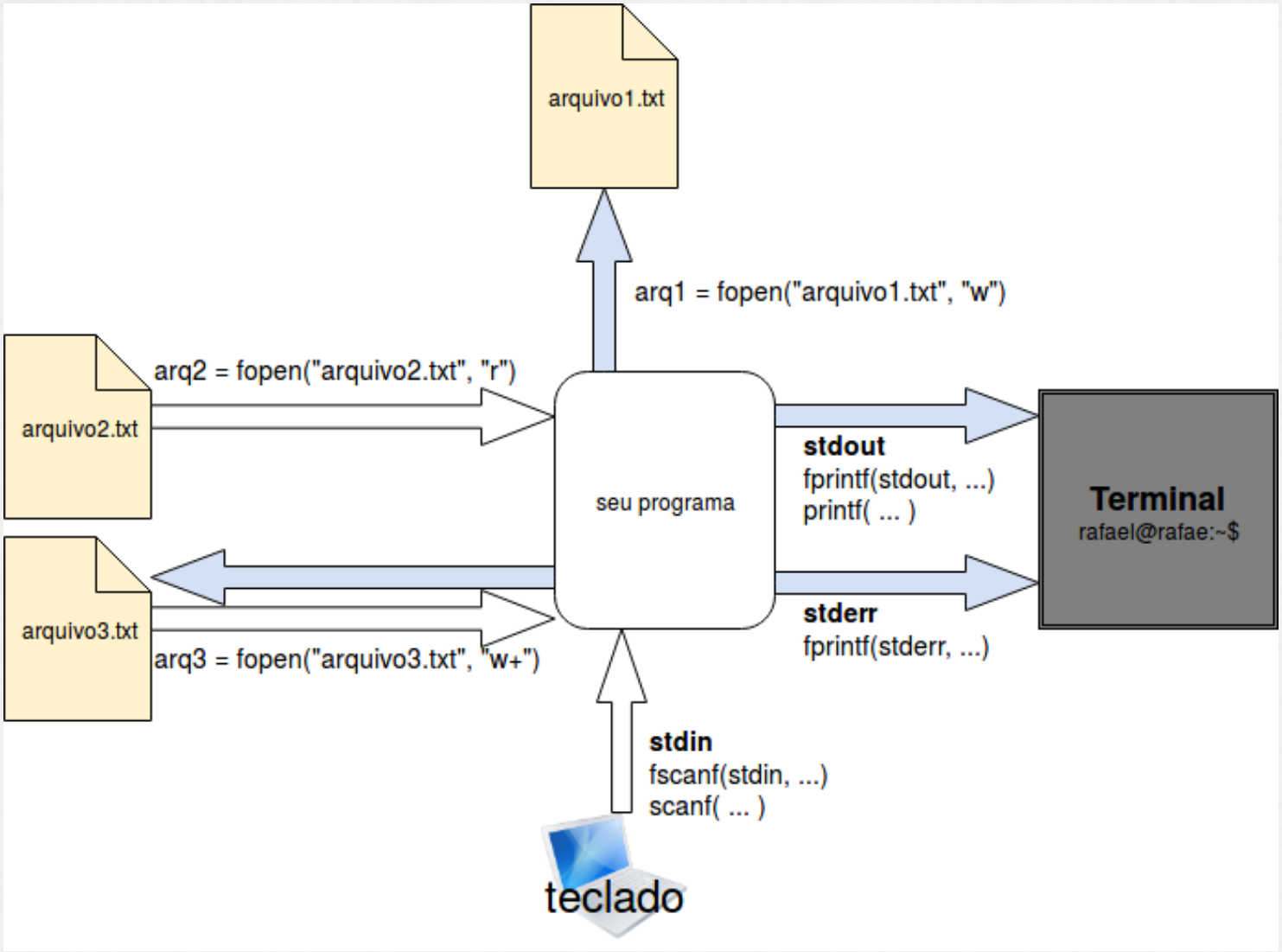
- arquivos
- leitura pelo teclado
- escrita no terminal

Um fluxo pode aceitar entrada de dados, saída de dados ou ambos.

- **stdio.h** (Standard Input/Output): funções para entrada e saída de dados
- stdio.h define qualquer fluxo em uma variável do tipo **FILE** (crie como ponteiro)

```
FILE *arq;
```

- há três fluxos padrão em C:
  - entrada padrão: **stdin** (em geral, o teclado)
  - saída padrão: **stdout** (em geral, o terminal)
  - erro padrão: **stderr** (em geral, o terminal)


arquivo1.txt
arq1 = fopen("arquivo1.txt", "w")
arquivo2.txt
arq2 = fopen("arquivo2.txt", "r")
seu programa
stdout
fprintf(stdout, ...)
printf(...)
Terminal
rafael@rafae:~$
arquivo3.txt
arq3 = fopen("arquivo3.txt", "w+")
stderr
fprintf(stderr, ...)
stdin
fscanf(stdin, ...)
scanf(...)
teclado

Definindo arquivo como stream

Abrir um arquivo usando C

- use a função *fopen* (de stdio.h); são dois parâmetros:
  1. nome do arquivo a ser aberto
  2. modo: a forma como o arquivo vai ser usado
- retorna NULL se não for possível abrir (o ideal é testar!)
- finalizando o uso, use a função *fclose* passando o arquivo como parâmetro

```
FILE *arq;
arq = fopen("teste.txt", "w");
if(arq == NULL) {
    printf("Nao foi possivel abrir o arquivo\n");
} else {
    //... processa o arquivo
}
fclose(arq);
```

Posição no arquivo

Quando um arquivo é aberto, o programa mantém uma **posição** (cursor) relativa ao início do arquivo:

```
2054 Indonesia Russia
8117 Indonesia Brazil
7569 Qatar Slovenia
```

```
2054 Indonesia Russia
8117 Indonesia Brazil
7569 Qatar Slovenia
```

```
2054 Indonesia Russia
8117 Indonesia Brazil
7569 Qatar Slovenia
```

O cursor avança à medida que os dados vão sendo lidos.

Modos de abertura

Modo	r	w	r+	w+[1]	a [2]	a+[3]
Se o arquivo não existe	erro	cria	erro	cria	cria	cria
Escreve?	não	sim	sim	sim	sim	sim
Lê?	sim	não	sim	sim	não	sim

[1] apaga o conteúdo no começo

[2] append, mantém a posição no arquivo sempre no fim

[3] funções para escrita alteram a posição para o fim do arquivo

Funções para stream

Escrevendo no fluxo

- use a função **fprintf**
- o primeiro parâmetro refere-se à **stream**
- os demais parâmetros são similares ao do **printf**
- exemplo:

```
FILE *arq;
arq = fopen("teste.txt", "w");
fprintf(arq, "%d %s", numero, texto);
fclose(arq);
```

✎ Exercício em sala

Escreva um programa em C que leia dois inteiros **a** e **b** (assuma $a < b$). Depois o programa deve escrever em um arquivo numeros.txt os números entre **a** e **b**.

Lendo do fluxo

- use a função **fscanf**
- o primeiro parâmetro refere-se ao **stream**
- os demais parâmetros são similares ao do **scanf**

```
FILE *arq;
arq = fopen("teste.txt", "w");
fscanf(arq, "%d %d", &numero1, &numero2);
fclose(arq);
```

- assim como scanf, retorna a quantidade de argumentos lidos com sucesso
- retorna EOF (constante definida em stdio.h) ao chegar no final do stream

```
FILE *arq;
arq = fopen("teste.txt", "w");
while(fscanf(arq, "%d %d", &numero1, &numero2) != EOF) {
    //... processa o par de numeros
}
fclose(arq);
```

✎ Exercício em sala

Escreva um programa em C que escreva na tela os números contidos no arquivo numeros.txt.

Reposicionando no arquivo

- fseek(FILE *stream, long int offset, int origin)
  - offset é dado em bytes
  - pode usar origin como:
    - SEEK_SET: começo do arquivo
    - SEEK_CUR: posição atual no arquivo
    - SEEK_END: final do arquivo
- ftell: retorna a posição no arquivo em bytes

Redirecionamento de streams

Redirecionamento da entrada

./a.out < input.txt
Mudando o fluxo stdin do teclado para o arquivo input.txt

Redirecionamento da saída

./a.out > output.txt
Mudando o fluxo stdout do terminal para o arquivo output.txt

Redirecionamento do erro

./a.out 2> erros.txt
Mudando o fluxo stderr do terminal para o arquivo erros.txt