

# Programação Estruturada/Programação II: Manipulação de Arquivos Binários em C

---

## Introdução

Manipular arquivos binários é essencial em C quando desejamos armazenar dados na forma "crua", exatamente como estão na memória, sem conversões para texto. Isso é útil para guardar estruturas, imagens ou qualquer conteúdo que não se restrinja a tipos de texto.

Assim como nos arquivos texto, usamos a biblioteca `<stdio.h>`, e as operações envolvem funções semelhantes, porém adaptadas para leitura e escrita de blocos de bytes.

## O ponteiro FILE\*

Para arquivos binários, continuamos utilizando a variável especial do tipo `FILE*`. Sua utilidade é a mesma: ser um "controle remoto" para acessar o arquivo e realizar operações de leitura ou escrita, de maneira abstrata e eficiente.

## O que é NULL?

Ao abrir um arquivo binário (por exemplo, para gravar dados de imagem), também devemos checar se a abertura foi bem sucedida. Se `fopen()` retorna `NULL`, houve uma falha (como ausência de permissões).

## Abrindo um Arquivo Binário

A diferença fundamental está no modo de abertura. Basta acrescentar um **'b'** na string do modo.

Exemplos:

- `"rb"` – Leitura binária (arquivo já deve existir).
- `"wb"` – Escrita binária (cria ou sobrescreve).
- `"ab"` – Append binário.
- `"rb+", "wb+", "ab+"` – Leitura e/ou escrita binárias.

Exemplo abrindo para escrita binária:

```
#include

int main() {
    FILE* arquivo = fopen("dados.bin", "wb");
    if (arquivo == NULL) {
        printf("Erro ao abrir o arquivo!\n");
        return 1;
    }

    printf("Arquivo binário aberto com sucesso!\n");
    fclose(arquivo);
}
```

```
    return 0;
}
```

## Gravando Dados em Arquivo Binário

Para salvar bytes diretamente (como inteiros, floats ou structs), usamos `fwrite()`. Sua chamada básica é:

```
fwrite(ponteiro, tamanho-do-elemento, qtd-elementos, arquivo);
```

Por exemplo, para guardar um array de inteiros:

```
#include

int main() {
    FILE* arquivo = fopen("numeros.bin", "wb");
    if (arquivo == NULL) {
        printf("Erro ao abrir o arquivo!\n");
        return 1;
    }

    int numeros[] = {10, 20, 30, 40, 50};
    fwrite(numeros, sizeof(int), 5, arquivo);

    fclose(arquivo);
    return 0;
}
```

## Lendo Dados de Arquivo Binário

Para ler, usamos `fread()`:

```
// fread(destino, tamanho-do-elemento, qtd-elementos, arquivo);
```

Exemplo lendo os inteiros salvos acima:

```
#include

int main() {
    FILE* arquivo = fopen("numeros.bin", "rb");
    if (arquivo == NULL) {
        printf("Erro ao abrir o arquivo!\n");
        return 1;
    }

    int numeros[5];
```

```

    fread(numeros, sizeof(int), 5, arquivo);

    // Exibir valores lidos
    for (int i = 0; i < 5; i++) {
        printf("%d\n", numeros[i]);
    }

    fclose(arquivo);
    return 0;
}

```

## Gravando e Lendo Estruturas

Imaginemos a seguinte struct:

```

typedef struct {
    char nome[50];
    int idade;
} Pessoa;

```

Gravando e lendo structs num arquivo binário:

```

// Escrevendo
Pessoa p = {"Alice", 25};
fwrite(&p, sizeof(Pessoa), 1, arquivo);

// Lendo
Pessoa pLido;
fread(&pLido, sizeof(Pessoa), 1, arquivo);

```

## Observações Importantes

- Arquivos binários NÃO são legíveis por humanos.
- O método `fgets()` e `fprintf()` são específicos para arquivos de texto e NÃO funcionam corretamente com arquivos binários.
- O acesso pode ser feito por registros de tamanho fixo, facilitando armazenamento de structs.

## Exercícios de Fixação

### 1. Grave e leia uma struct em um arquivo binário:

- Crie um programa em C que solicite nome e idade do usuário, armazene-os numa struct e salve no arquivo "usuario.bin".
- Feche o arquivo. Depois, abra para leitura binária, recupere os dados e exiba na tela.

### 2. Crie e leia um vetor de inteiros binário:

- Crie um array de 100 inteiros (valores de 1 a 100) e salve em "vetor.bin".
- Em outro programa, leia esse arquivo binário e exiba todos os números.

### 3. **Salve e leia múltiplos registros:**

- Escreva um programa para cadastrar 3 pessoas (nome/idade), salvar no arquivo "pessoas.bin" e outro programa para ler e mostrar os 3 registros gravados.
-