Manipulação de Arquivos (Entrada e Saída)

Nesta semana, vamos explorar a **manipulação de arquivos** em C, que é uma habilidade essencial para armazenar dados de forma permanente e recuperar informações de arquivos. Você aprenderá a trabalhar com arquivos de texto e binários, ler e escrever neles, além de ver como aplicar essas técnicas em soluções práticas.

1. Introdução aos Arquivos

Arquivos são usados para armazenar dados fora da memória volátil (RAM), permitindo que os dados permaneçam disponíveis após a execução do programa. Existem dois tipos principais de arquivos em programação:

- **Arquivos de Texto**: Contêm dados em formato legível por humanos. Cada linha é separada por um caractere de nova linha (`\n`).
- **Arquivos Binários**: Contêm dados em formato binário (não legível por humanos), usados para armazenar dados mais complexos, como imagens, vídeos, ou até mesmo números em formatos compactados.

2. Tipos de Arquivos

Arquivos de Texto

Em arquivos de texto, os dados são gravados como caracteres simples. Eles podem ser lidos e manipulados facilmente. Em C, os arquivos de texto são manipulados utilizando funções como `fopen`, `fscanf`, `fprintf`, e `fclose`.

Arquivos Binários

Em arquivos binários, os dados são armazenados como bytes. Esse tipo de arquivo é mais eficiente em termos de armazenamento e manipulação de dados complexos (como estruturas de dados, arrays de números, etc.). Em C, arquivos binários são manipulados com funções como `fopen`, `fread`, `fwrite`, e `fclose`.

---

3. Abertura, Leitura e Escrita de Arquivos

Abrindo Arquivos

Em C, usamos a função `fopen` para abrir arquivos. O primeiro parâmetro é o nome do arquivo, e o segundo é o modo de abertura:

- `"r"`: Abre para leitura (arquivo deve existir).
- `"w"`: Abre para escrita (cria o arquivo ou apaga o conteúdo existente).
- `"a"`: Abre para escrita no final do arquivo (não apaga o conteúdo existente).
- `"rb"` e `"wb"`: Para arquivos binários, respectivamente, leitura e escrita.

Exemplo de abertura de arquivo para leitura:

```
FILE *arquivo = fopen("exemplo.txt", "r");
if (arquivo == NULL) {
    printf("Erro ao abrir o arquivo para leitura\n");
    return 1;
}
```

Leitura de Arquivos

Para ler dados de um arquivo, podemos usar funções como `fscanf` (para ler dados formatados) ou `fgets` (para ler uma linha de texto).

Exemplo de leitura de um arquivo de texto:

```
char linha[100];
FILE *arquivo = fopen("exemplo.txt", "r");
if (arquivo == NULL) {
    printf("Erro ao abrir o arquivo\n");
    return 1;
}

while (fgets(linha, sizeof(linha), arquivo)) {
    printf("%s", linha); // Exibe cada linha do arquivo
}

fclose(arquivo); // Fecha o arquivo após a leitura
```

Escrita de Arquivos

Para escrever dados em arquivos, usamos funções como `fprintf` (para escrita formatada) ou `fputs` (para escrever uma linha de texto).

Exemplo de escrita em um arquivo de texto:

```
FILE *arquivo = fopen("saida.txt", "w");
if (arquivo == NULL) {
    printf("Erro ao abrir o arquivo\n");
    return 1;
}

fprintf(arquivo, "Este é um exemplo de escrita em arquivo.\n");

fclose(arquivo); // Fecha o arquivo após a escrita
```

4. Manipulação de Arquivos Binários

Abertura de Arquivos Binários

A principal diferença na manipulação de arquivos binários é que o modo de abertura deve incluir os sufixos `b` (de binário), como `"rb"` para leitura e `"wb"` para escrita.

Exemplo de abertura de um arquivo binário:

```
FILE *arquivo = fopen("dados.bin", "wb");
if (arquivo == NULL) {
    printf("Erro ao abrir o arquivo binário para escrita\n");
    return 1;
}
```

Leitura e Escrita em Arquivos Binários

A leitura e escrita em arquivos binários é feita com as funções `fread` e `fwrite`, que manipulam dados em blocos de bytes.

Exemplo de gravação de dados binários:

```
#include <stdio.h>

typedef struct {
    int idade;
    float salario;
} Funcionario;

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

    Funcionario func = {25, 3500.50};
    fwrite(&func, sizeof(Funcionario), 1, arquivo); // Escreve o struct no arquivo
binário

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

Não se preocupe com a presença de structs. Será o último tópico que estudaremos.

Exemplo de leitura de dados binários:

```
#include <stdio.h>

typedef struct {
    int idade;
```

```
    float salario;
} Funcionario;

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

    Funcionario func;
    fread(&func, sizeof(Funcionario), 1, arquivo); // Lê o struct do arquivo
binário
    printf("Idade: %d\n", func.idade);
    printf("Salário: %.2f\n", func.salario);

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

5. Funções de Manipulação de Arquivos

- `fopen`: Abre um arquivo.
- `fclose`: Fecha um arquivo.
- `fscanf`: Lê dados formatados de um arquivo.
- `fprintf`: Escreve dados formatados em um arquivo.
- `fgets`: Lê uma linha de um arquivo.
- `fputs`: Escreve uma linha em um arquivo.
- `fread` e `fwrite`: Usadas para manipulação de arquivos binários.

6. Aplicações Práticas

Leitura de Arquivo de Texto e Contagem de Palavras

Crie um programa que leia um arquivo de texto e conte o número de palavras. Aqui está um exemplo básico para realizar isso:

```
#include <stdio.h>
#include <ctype.h>

int main() {
    FILE *arquivo = fopen("texto.txt", "r");
    if (arquivo == NULL) {
        printf("Erro ao abrir o arquivo\n");
        return 1;
    }
```

```
    int palavras = 0;
    char c;
    int dentroPalavra = 0; // Flag para verificar se está dentro de uma palavra

    while ((c = fgetc(arquivo)) != EOF) {
        if (isalpha(c)) {
            if (!dentroPalavra) {
                palavras++;
                dentroPalavra = 1;
            }
        } else {
            dentroPalavra = 0; // Quando encontra um espaço ou pontuação
        }
    }

    printf("Numero de palavras: %d\n", palavras);
    fclose(arquivo);

    return 0;
}
```

Gravação de Resultados em Arquivos de Saída

Suponha que você tenha um programa que calcule a média de notas de um aluno e grave o resultado em um arquivo de saída:

```
#include <stdio.h>

int main() {
    FILE *arquivo = fopen("resultado.txt", "w");
    if (arquivo == NULL) {
        printf("Erro ao abrir o arquivo\n");
        return 1;
    }

    float notas[5] = {8.5, 7.0, 9.2, 6.8, 7.5};
    float soma = 0;
    for (int i = 0; i < 5; i++) {
        soma += notas[i];
    }
    float media = soma / 5;

    fprintf(arquivo, "Média das notas: %.2f\n", media);

    fclose(arquivo);
    printf("Resultado gravado em resultado.txt\n");

    return 0;
}
```

7. Exercícios práticos

1. **Leitura de Arquivo de Texto**: Crie um programa que leia um arquivo de texto e exiba seu conteúdo na tela.
2. **Contagem de Palavras**: Desenvolva um programa que leia um arquivo de texto e conte quantas palavras ele contém.
3. **Gravação de Resultados em Arquivo**: Crie um programa que leia as notas de 5 alunos e grave as médias em um arquivo de saída.
4. **Manipulação de Arquivos Binários**: Escreva um programa que leia e escreva dados numéricos em um arquivo binário.
5. **Verificação de Existência de Arquivo**: Crie um programa que verifique se um arquivo existe e informe o status.

8. Conclusão

Com a prática da manipulação de arquivos, você será capaz de criar programas que armazenam e recuperam dados de forma persistente. A manipulação de arquivos binários oferece uma forma eficiente de trabalhar com dados estruturados, enquanto os arquivos de texto são ideais para manipulação de dados legíveis. Experimente os exercícios propostos e aprenda a usar a leitura e escrita de arquivos para resolver problemas práticos!