UnidadeIII - Manipulação de Arquivo em Linguagem C

Sumário

Introdução

A manipulação de arquivos em C é uma tarefa importante para muitos programas, pois permite que os dados sejam armazenados e recuperados de forma persistente. Neste tópico estudaremos os conceitos sobre arquivos em linguagem C, incluindo a abertura, leitura, escrita, fechamento, criação de arquivos.

A manipulação de arquivos em C é uma habilidade essencial para processar entrada/saída de dados de forma eficiente e confiável e incluem importantes elementos, tais como:

1. **Armazenamento de Dados:** Permite armazenar dados de forma persistente em arquivos, o que é essencial para muitos tipos de aplicativos, como bancos de dados, processamento de texto, jogos, entre outros.
2. **Portabilidade:** O acesso a arquivos é uma operação comum em sistemas operacionais modernos. Ao manipular arquivos em C, você pode criar programas portáteis que funcionam em diferentes sistemas operacionais sem grandes modificações.
3. **Processamento de Grandes Dados:** Arquivos podem conter grandes volumes de dados que podem não caber totalmente na memória RAM disponível. Manipular esses arquivos permite processar esses dados em partes, evitando problemas de falta de memória.
4. **Persistência de Dados:** Os arquivos permitem que os dados sejam armazenados permanentemente mesmo após o encerramento do programa. Isso é útil para manter informações importantes entre diferentes execuções do programa.
5. **Interoperabilidade:** A manipulação de arquivos é fundamental para a comunicação entre diferentes programas. Por exemplo, um programa pode escrever dados em um arquivo que outro programa pode ler e processar posteriormente.

Além das necessidades relacionadas ao processamento de arquivos textos apresentados acima, o estudo também é importante para o aprendizado de programação por várias razões, como:

1. **Conceitos Fundamentais**: A manipulação de arquivos em C permite aos iniciantes entenderem conceitos fundamentais de entrada/saída (I/O) de dados, como abrir, ler, escrever e fechar arquivos.
2. **Aplicabilidade Prática**: É uma habilidade prática que os programadores usam em uma variedade de projetos do mundo real. Ao aprender a manipular arquivos em C, é possível aplicar imediatamente esse conhecimento em projetos pessoais ou profissionais, além de desenvolver habilidades importantes, como

gerenciamento de memória, tratamento de erros e compreensão de ponteiros, que são essenciais para se tornar um programador competente em C ou em outras linguagens.

3. **Resolução de Problemas**: Trabalhar com arquivos em C é importante, também, para resolver problemas relacionados ao processamento de dados, armazenamento e recuperação de informações. Isso ajuda no desenvolvimento da capacidade de resolução de problemas.
4. **Entendimento do Funcionamento Interno**: Ao manipular arquivos em C, compreende-se mais profundamente como os sistemas operacionais lidam com operações de E/S e como os dados são armazenados fisicamente no disco.

Concluindo...

A manipulação de arquivos em C é uma parte essencial do aprendizado de programação, pois proporciona uma base sólida em conceitos fundamentais, desenvolve habilidades práticas e promove uma compreensão mais profunda dos sistemas de computadores e da programação em geral.

1 Manipulação de Arquivos em C

Em muitos programas de computador, é essencial a capacidade de ler dados de arquivos armazenados no disco rígido, bem como escrever dados em arquivos para persistência de informações. A linguagem de programação C fornece um conjunto robusto de funções para manipulação de arquivos, permitindo que os programadores realizem uma ampla variedade de operações de entrada e saída de dados.

Ao trabalhar com arquivos em C, é possível abrir, criar, ler, escrever e fechar arquivos de maneira eficiente e confiável. Isso não apenas permite que os programas interajam com dados armazenados externamente, mas também oferece uma maneira de preservar informações entre diferentes execuções do programa, assim então, exploraremos as principais funções para manipulação de arquivos em C, desde a abertura de arquivos até operações de leitura e escrita e, também o fechamento correto dos arquivos. Veremos como usar essas funções para realizar operações comuns de manipulação de arquivos e como lidar com possíveis erros durante essas operações.

Ao compreender e dominar as funções para manipulação de arquivos em C, os será possível criar algoritmos mais poderosos e flexíveis, capazes de lidar com uma variedade de situações do mundo real que envolvem o armazenamento e o processamento de dados em arquivos.

1.1 Funções para manipulação de arquivo:

Para iniciar a manipulação de arquivo texto em C, vamos começar explorando essas funções essenciais e descobrir como usá-las para manipular arquivos em programas escritos em C. A lista abaixo apresenta as principais funções de manipulação de arquivos da biblioteca stdio.h ou cstdio.

Tabela de Funções da biblioiteca stdio.h:

Função	Descrição
fopen()	Abre um arquivo.
fclose()	Fecha o arquivo garantindo a transferência do buffer.
fflush()	Descarrega o buffer.
fscanf()	Leitura de entrada formatada (semelhante ao scanf()).
fprintf()	Escrita de saída formatada (semelhante ao printf()).
fgets()	Obtém uma string do arquivo.
fgetc()	Obtém um caracter do arquivo.
fputs()	Insere uma string no arquivo.
fputc()	Insere um caracter no arquivo.
fread()	Lê um bloco de dados do arquivo.
fwrite()	Escreve um bloco de dados no arquivo.
fseek()	Reposiciona o ponteiro.
rewind()	Reposiciona o ponteiro para o início do arquivo.

ftell()	Retorna a posição do ponteiro.

2. Criando um arquivo

Para criar um arquivo novo, usa-se a função fopen() com o modo "w" (escrita).

Sintaxe:

Fopen(“nomearquivotxt.txt”,mododeabertura”)

Exemplo: Trecho de código para criar um arquivo.

```
 int main() {
FILE *arquivo = fopen("exemplo.txt", "w");
if (arquivo != NULL) {
   cout<<"Arquivo criado com sucesso.\n";
   fclose(arquivo); }
else {
    cout<< “Erro ao criar o arquivo.\n”;
}
return 0;
}
```

2.1 Modo de abertura de arquivo:

Em C, a biblioteca padrão <stdio.h> oferece vários modos de abertura de arquivo texto para atender às diferentes necessidades de manipulação de arquivos. Abaixo estão relacionados os modos mais comuns de abertura de arquivo texto:

- r - abre um arquivo no modo leitura
- w - abre ou cria um arquivo de texto no modo de escrita
- a - abre um arquivo no modo de inclusão (append)
- r+ - abre um arquivo nos modos de leitura e escrita
- a+ - abre um arquivo nos modos de leitura e escrita
- w+ - abre um arquivo nos modos de leitura e escrita

2.1.2 Exemplos de uso do modo de abertura de arquivo

1. **"r" - Modo de Leitura (Read)**:
    - Este modo é usado para abrir um arquivo existente para leitura.
    - Se o arquivo não existir, a função **fopen()** falhará.
    - O ponteiro do arquivo será colocado no início do arquivo.

```
FILE *arquivo = fopen("exemplo.txt", "r");
```

2. **"w" - Modo de Escrita (Write)**:
    - Este modo é usado para criar um novo arquivo para escrita.
    - Se o arquivo já existir, seu conteúdo será apagado.
    - Se o arquivo não existir, será criado.
    - O ponteiro do arquivo será colocado no início do arquivo.

```
FILE *arquivo = fopen("exemplo.txt", "w");
```

3. **"a" - Modo de Anexação (Append)**:
    - Este modo é usado para abrir um arquivo para escrita, adicionando dados ao final do arquivo.
    - Se o arquivo não existir, será criado.
    - O ponteiro do arquivo será colocado no final do arquivo, permitindo que novos dados sejam adicionados sem apagar o conteúdo existente.

```
FILE *arquivo = fopen("exemplo.txt", "a");
```

4. **"r+" - Modo de Leitura/Gravação (Read/Write)**:
    - Este modo é usado para abrir um arquivo existente para leitura e escrita.
    - O ponteiro do arquivo será colocado no início do arquivo.
    - Se o arquivo não existir, a função **fopen()** falhará.

```
FILE *arquivo = fopen("exemplo.txt", "r+");
```

5. **"w+" - Modo de Leitura/Gravação (Write/Read)**:
    - Este modo é usado para criar um novo arquivo para leitura e escrita.
    - Se o arquivo já existir, seu conteúdo será apagado.
    - Se o arquivo não existir, será criado.
    - O ponteiro do arquivo será colocado no início do arquivo.

```
FILE *arquivo = fopen("exemplo.txt", "w+");
```

6. **"a+" - Modo de Anexação/Leitura (Append/Read)**:
    - Este modo é usado para abrir um arquivo para leitura e escrita, adicionando dados ao final do arquivo.
    - Se o arquivo não existir, será criado.
    - O ponteiro do arquivo será colocado no final do arquivo.

```
FILE *arquivo = fopen("exemplo.txt", "a+");
```

Estes são os modos básicos de abertura de arquivo texto em C. Cada modo oferece diferentes funcionalidades e comportamentos, permitindo uma manipulação flexível de arquivos de texto de acordo com os requisitos do programa.

2.1.3 Arquivo .dat e Arquivo .txt

Em linguagem C, existem várias formas de armazenar dados em arquivos, cada uma adequada para diferentes necessidades e tipos de dados. Alguns dos tipos de arquivos comuns para armazenar dados em C incluem:

1. **Arquivo .dat (Dados Binários)**:
    - Um arquivo .dat geralmente armazena dados em formato binário, ou seja, os dados são armazenados diretamente em sua representação numérica, sem formatação especial.
    - Não há nenhuma estruturação específica para delimitar caracteres ou linhas no arquivo. Os dados são armazenados em uma sequência de bytes, e a interpretação desses bytes depende do programa que os lê e escreve.
    - Os arquivos .dat são frequentemente usados quando é necessário preservar a estrutura e a integridade dos dados

originais, como em arquivos de imagens, áudio, vídeos, bancos de dados, entre outros.

- Como os dados são armazenados em sua forma bruta, os arquivos .dat podem ocupar menos espaço em disco e ser mais eficientes para certos tipos de dados.

2. **Arquivo .txt (Texto Simples)**:
    - Um arquivo .txt armazena dados em formato de texto simples, onde os caracteres são codificados em um conjunto de caracteres específico, como ASCII ou UTF-8.
    - Os dados são organizados em linhas de texto, onde cada linha é terminada por um caractere de nova linha ('\n' ou '\r\n', dependendo do sistema operacional).
    - Os arquivos .txt são frequentemente usados para armazenar dados legíveis por humanos, como documentos de texto, scripts, configurações de programas, entre outros.
    - Como os dados são armazenados em texto simples, os arquivos .txt podem ser facilmente editados em qualquer editor de texto e são amplamente interoperáveis entre diferentes sistemas operacionais e aplicativos.

A principal diferença entre criar um arquivo .dat e um arquivo .txt está no formato dos dados armazenados e na forma como esses dados são interpretados. Arquivos .dat são usados para armazenar dados binários brutos, enquanto arquivos .txt são usados para armazenar dados em formato de texto simples.

2.1.4 Manipulando arquivo binário

Um arquivo binário é um tipo de arquivo que armazena dados em uma representação binária, ou seja, os dados são armazenados em uma sequência de bits. Esses arquivos podem conter qualquer tipo de dados, como texto, números, imagens, áudio, vídeos, entre outros.

A principal característica dos arquivos binários é que eles não possuem uma estrutura específica para delimitar caracteres ou linhas, como é o caso dos arquivos de texto. Em vez disso, os dados são armazenados diretamente na forma de bytes, sem interpretação ou conversão automática.

A interpretação dos dados em um arquivo binário depende do programa que os lê e escreve. Por exemplo, um programa pode interpretar os bytes de um arquivo binário como números inteiros, enquanto outro programa pode interpretá-los como caracteres de texto.

Os arquivos binários são amplamente utilizados em computação para armazenar dados de forma eficiente e preservar sua estrutura original. Eles são frequentemente usados em situações em que a representação exata dos dados é crucial, como em arquivos de imagem, áudio, banco de dados, executáveis de programas, entre outros.

2.1.4.1 Abrindo arquivo .dat com modo rb

fopen("arquivo.dat","rb")

Essa linha de código está utilizando a função **fopen()** para abrir um arquivo chamado "arquivo.dat" no modo de leitura binária (**"rb"**).

Detalhes da linha de código:

- **fopen()**: Esta é uma função da biblioteca padrão em C que é usada para abrir um arquivo. Ela retorna um ponteiro para um objeto FILE, que é usado posteriormente para realizar operações de entrada e saída no arquivo.
- **"arquivo.dat"**: Este é o nome do arquivo que será aberto. No caso, "arquivo.dat" é o nome do arquivo que será manipulado.
- **"rb"**: Este é o modo de abertura do arquivo. No caso, **"rb"** indica que o arquivo será aberto para leitura binária. Sendo:
    - **"r"**: Indica que o arquivo será aberto para leitura.
    - **"b"**: Indica que o arquivo será aberto em modo binário. Isso significa que o arquivo será lido ou gravado em sua forma binária, sem realizar conversões automáticas de nova linha ou caracteres especiais.

Essa linha de código está abrindo o arquivo "arquivo.dat" em modo de leitura binária, o que significa que o programa poderá ler os dados do arquivo em sua forma binária sem realizar qualquer conversão automática de formato de nova linha ou caracteres especiais.

2.1.4.2 Abrindo arquivo .dat com modo wb

fopen("arquivo.dat","wb")

Essa linha de código está usando a função **fopen()** para abrir (ou criar) um arquivo chamado "arquivo.dat" em modo de escrita binária ("wb").

Detalhes da linha de código:

- **fopen()**: Esta é uma função da biblioteca padrão em C, usada para abrir arquivos. Ela retorna um ponteiro para um objeto **FILE**, que é usado para realizar operações de entrada e saída no arquivo.
- **"arquivo.dat"**: Este é o nome do arquivo que será aberto ou criado. No caso, "arquivo.dat" é o nome do arquivo com o qual estamos lidando. Se o arquivo já existir, ele será aberto; caso contrário, um novo arquivo com esse nome será criado.
- **"wb"**: Este é o modo de abertura do arquivo. No caso, "wb" indica que o arquivo será aberto para escrita binária. Sendo:
    - **"w"**: Indica que o arquivo será aberto para escrita (write). Se o arquivo já existir, seu conteúdo será apagado antes que os novos dados sejam escritos nele. Se o arquivo não existir, um novo arquivo será criado. Atenção ao usar este modo para não apagar os dados do arquivo, então, para não teremos que usar de uma validação e verificação.
    - **"b"**: Indica que o arquivo será aberto em modo binário. Isso significa que os dados serão escritos ou lidos em sua forma bruta, sem conversões automáticas ou interpretação de caracteres especiais. Em sistemas que diferenciam entre modos binários e de texto, como o Windows, o modo "b" é necessário ao lidar com arquivos binários. Em sistemas Unix-like, como Linux e macOS, o modo "b" é ignorado, mas é comum incluí-lo por portabilidade.

Portanto, essa linha de código está abrindo o arquivo "arquivo.dat" em modo de escrita binária, permitindo que dados binários sejam escritos no arquivo. Se o arquivo já existir, seu conteúdo será apagado; se não existir, um novo arquivo será criado.

2.1.4.2 Exemplo:

O trecho de código, abaixo, apresenta um algoritmo de manipulação de arquivo contendo algumas das funcionalidades iniciais essenciais, a saber: criar, abrir, armazenar e fechar um arquivo de dados no modo binário. Desta forma temos nas linhas iniciais o uso da função de manipulação de arquivo fopen() com os modos de abertura “rb” e “wb”. Após a criação e abertura do arquivo, temos o fopen() com o modo “a+b” para permitir a abertura do arquivo para leitura de dados e na sequência temos a função fwrite() que permitirá a inserção/escrita de dados no arquivo criado e aberto. No final do trecho de código temos a função fclose() que fecha o arquivo para evitar vazamento de recursos e possíveis problemas de acesso concorrente afim de manter a segurança dos dados. O arquivo de dados poderá ser acessado novamente através do programa de manipulação construído para este propósito.

```
#include<iostream>
#include<string.h>
#include<cstdio>
#include<stdlib.h>
using namespace std;

struct cadastro{
    char nome[30];
    int ano;
};

int main(){
    cadastro cad;
    int i;
    char aux[30];
    FILE *fp;

    if ((fp = fopen("arqcad.dat", "rb")) == NULL){
        fp = fopen ("arqcad.dat", "wb");
        cout<<"Novo arquivo criado com sucesso!"<<endl;
        system("Pause");
        system("cls");
    }
    fp = fopen("arqcad.dat", "a+b");
    fflush(stdin);
    cout<<"Nome: ";
    gets(aux);
    strcpy(cad.nome, aux);
    fflush(stdin);
    cout<<"Ano: "; cin>>cad.ano;
    fwrite(&cad,sizeof(cad),1,fp);
    fclose(fp);
    return 0; }
```