
Fundamentos de Programação

Aula 13 - Arquivos - Parte 1

Arnaldo Barreto Vila Nova

Sumário

- Introdução
 - Declaração
 - Abertura
 - Fechamento
 - Gravação
 - Leitura
 - Exercícios
-

Introdução

- Imagine que você precisa cadastrar num sistema os dados de 100 pessoas. Nome, endereço, telefone, idade...
 - E depois de digitar os dados de 99 dessas pessoas, cai a energia e o computador desliga.
 - Então você descobre que todos os dados que você digitou não foram salvos e terá de preencher tudo novamente
 - Para evitar este tipo de situação, devemos entender como criar e manipular arquivos externos em nossos códigos
-

Introdução

- A manipulação de arquivos possibilita armazenar dados de forma permanente em uma memória secundária
 - Um arquivo é um conjunto de dados organizados, também chamados de registros
 - A manipulação depende do tipo de organização do arquivo
 - Arquivos de acesso sequencial
 - Arquivos de acesso aleatório
-

Introdução

- Na Linguagem C, um arquivo é considerado um fluxo contínuo de dados, carregado em um *buffer* normalmente denominado de *stream*.
 - Um *stream* associado a um arquivo é carregado na memória principal através de uma operação de abertura
 - As manipulações são então realizadas no *stream* para depois atualizar o arquivo através de uma operação de fechamento
-

Introdução

- Imagine o stream como se fosse uma seta ou marcação em um arquivo indicando em que parte do arquivo estamos visualizando.
 - Toda vez que uma informação é gravada ou lida, o stream é incrementado seguindo o fluxo do arquivo.
 - Existe também o caractere especial EOF (End Of File), que indica se o stream de um arquivo chegou ao final
 - Ou seja, se a informação lida a partir de um arquivo for igual a EOF, o stream chegou no final do arquivo.
-

Introdução

- Para trabalharmos com arquivos, temos 4 operações básicas para entender:
 - Abertura de arquivo: relaciona um stream com o arquivo desejado, ou seja, carrega na memória principal (RAM) o vínculo com o arquivo na memória secundária (HD) ;
 - Leitura de arquivo: lê determinado trecho do *stream*, ou seja, recebe determinadas informações do arquivo vinculado;
 - Escrita no arquivo: acrescenta ou altera o *stream*;
 - Fechamento de arquivo: Toda a informação no *stream* é atualizada no arquivo em disco e a área do *buffer* é liberada.
-

Declaração

- Usando a biblioteca <stdio.h> podemos declarar um *stream* onde iremos carregar um arquivo como um ponteiro de uma estrutura especial **FILE**:

```
FILE *arquivo;
```

Declaração

- Usando a biblioteca <stdio.h> podemos declarar um *stream* onde iremos carregar um arquivo como um ponteiro de uma estrutura especial **FILE**:

```
FILE *arquivo;
```

- Este ponteiro relaciona o sistema de E/S a um endereço de memória onde será carregado o arquivo
-

Abertura

- A função que abre um arquivo carregando ele no *stream* desejado é o **fopen**, que retorna um ponteiro associado ao arquivo (ou NULL, caso o arquivo não possa ser aberto)
-

Abertura

- A função que abre um arquivo carregando ele no *stream* desejado é o **fopen**, que retorna um ponteiro associado ao arquivo (ou NULL, caso o arquivo não possa ser aberto)
 - Essa função precisa de 2 parâmetros
 - nome físico do arquivo
 - modo de abertura
-

Abertura

- A função que abre um arquivo carregando ele no *stream* desejado é o **fopen**, que retorna um ponteiro associado ao arquivo (ou NULL, caso o arquivo não possa ser aberto)
- Essa função precisa de 2 parâmetros
 - nome físico do arquivo
 - modo de abertura

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

Abertura

- A tabela abaixo mostra os principais modos de abertura

Modo	Significado
"r"	Abre um arquivo texto para leitura. O arquivo deve existir antes de ser aberto.
"w"	Abrir um arquivo texto para gravação. Se o arquivo não existir, ele será criado. Se já existir, o conteúdo anterior será destruído.
"a"	Abrir um arquivo texto para gravação. Os dados serão adicionados no fim do arquivo ("append"), se ele já existir, ou um novo arquivo será criado, no caso de arquivo não existente anteriormente.
"rb"	Abre um arquivo binário para leitura. Igual ao modo "r" anterior, só que o arquivo é binário.
"wb"	Cria um arquivo binário para escrita, como no modo "w" anterior, só que o arquivo é binário.
"ab"	Acrescenta dados binários no fim do arquivo, como no modo "a" anterior, só que o arquivo é binário.
"r+"	Abre um arquivo texto para leitura e gravação. O arquivo deve existir e pode ser modificado.
"w+"	Cria um arquivo texto para leitura e gravação. Se o arquivo existir, o conteúdo anterior será destruído. Se não existir, será criado.
"a+"	Abre um arquivo texto para gravação e leitura. Os dados serão adicionados no fim do arquivo se ele já existir, ou um novo arquivo será criado, no caso de arquivo não existente anteriormente.
"r+b"	Abre um arquivo binário para leitura e escrita. O mesmo que "r+" acima, só que o arquivo é binário.
"w+b"	Cria um arquivo binário para leitura e escrita. O mesmo que "w+" acima, só que o arquivo é binário.
"a+b"	Acrescenta dados ou cria um arquivo binário para leitura e escrita. O mesmo que "a+" acima, só que o arquivo é binário.

Abertura


- Exemplos principais:

Aqui o arquivo está sendo
aberto somente para leitura

```
FILE *arquivo;  
arquivo = fopen("texto.txt", "r");  
  
if (arquivo == NULL)  
    printf("Ocorreu um erro de abertura");
```

Abertura

- Exemplos principais:

```
FILE *arquivo;  
arquivo = fopen("texto.txt", "r");  
  
if (arquivo == NULL)   
    printf("Ocorreu um erro de abertura");
```

Se o arquivo não existir, se o usuário não tiver
permissão para ler o arquivo ou se não tiver
memória RAM suficiente, o **fopen** retorna NULL

Abertura

- Exemplos principais:

Aqui o arquivo está sendo
aberto para escrita do zero

```
FILE *arquivo;  
arquivo = fopen("texto.txt", "w");  
  
if (arquivo == NULL)  
    printf("Ocorreu um erro de abertura");
```

Abertura

- Exemplos principais:

```
FILE *arquivo;
```

```
arquivo = fopen("texto.txt", "w");
```



```
if (arquivo == NULL)
```

```
    printf("Ocorreu um erro de abertura");
```

Se o arquivo não existir, ele será criado.

Se o usuário não tiver permissão de escrita, o **fopen** retorna NULL.

Se o arquivo existir e o usuário tiver permissão, o conteúdo será apagado e iniciado do zero

Abertura

- Exemplos principais: Aqui o arquivo está sendo aberto em modo de “escrita no final do arquivo”

```
FILE *arquivo;
```

```
arquivo = fopen("texto.txt", "a");
```



```
if (arquivo == NULL)
```

```
    printf("Ocorreu um erro de abertura");
```

Abertura

- Exemplos principais:

```
FILE *arquivo;
```



```
arquivo = fopen("texto.txt", "a");
```

```
if (arquivo == NULL)
```

```
    printf("Ocorreu um erro de abertura");
```

Se o arquivo não existir, ele será criado.

Se o usuário não tiver permissão de escrita, o **fopen** retorna NULL.

Se o arquivo existir e o usuário tiver permissão, o conteúdo será mantido e o stream é posicionado no final do arquivo para novas informações serem adicionadas.

Fechamento

- Para atualizar os dados do *stream* no disco é utilizada a função **fclose**, que retorna 0 se o arquivo foi fechado com sucesso ou EOF (end of file) caso tenha ocorrido algum erro
-

Fechamento

- Para atualizar os dados do *stream* no disco é utilizada a função **fclose**, que retorna 0 se o arquivo foi fechado com sucesso ou EOF (end of file) caso tenha ocorrido algum erro
 - Ele recebe como parâmetro o ponteiro do arquivo
-

Fechamento

- Para atualizar os dados do *stream* no disco é utilizada a função **fclose**, que retorna 0 se o arquivo foi fechado com sucesso ou EOF (end of file) caso tenha ocorrido algum erro
- Ele recebe como parâmetro o ponteiro do arquivo

```
FILE *arquivo;  
arquivo = fopen("texto.txt", "a"); /*...*/  
if (fclose(arquivo) != 0)  
    printf("ocorreu um erro de fechamento");
```

Gravação

- Algumas funções para gravação de dados em arquivos:
 - **fputc**: grava um caracter por vez no stream do arquivo
 - **fprintf**: semelhante ao printf, grava dados formatados no arquivo de acordo com o tipo de cada dado gravado
 - **fwrite**: grava um bloco de dados sequenciado (array, struct, ...), normalmente em formato binário
 - A gravação é realizada a partir do endereço no ponteiro do arquivo, cujo stream é incrementado a cada gravação
-

Gravação

- **fputc** tem como parâmetros o caractere a ser escrito e o *stream* no qual será escrito

fputc (<var> , <*stream>) ;

Gravação

- **fputc** tem como parâmetros o caractere a ser escrito e o *stream* no qual será escrito

fputc(<var>,<*stream>);

```
FILE *arquivo;  
arquivo = fopen("texto.txt", "w");  
fputc('a',arquivo);  
char c = 's';  
fputc(c,arquivo);  
fclose(arquivo);
```

Gravação

- **fprintf** grava os dados em uma determinada formatação de texto, de forma similar ao printf, só que adicionando o ponteiro do stream como parâmetro

```
fprintf(<*stream>, <formatação>, valor1, ...);
```

Gravação

- **fprintf** grava os dados em uma determinada formatação de texto, de forma similar ao printf, só que adicionando o ponteiro do stream como parâmetro

fprintf(<*stream>, <formatação>, valor1, ...);

```
FILE *arquivo;  
arquivo = fopen("texto.txt", "w");  
fprintf(arquivo, "O número %d é lido %s", 2, "dois");  
fclose(arquivo);
```

Gravação

- **fwrite** grava um bloco de dados. Recebe como parâmetros: o ponteiro de início do bloco, o tamanho de cada registro, o número de registros e o ponteiro do *stream*.

```
fwrite(<*p>, <tam_bytes>, <qtd>, <*stream>);
```

Gravação

- **fwrite** grava um bloco de dados. Recebe como parâmetros: o ponteiro de início do bloco, o tamanho de cada registro, o número de registros e o ponteiro do *stream*.

`fwrite(<*p>, <tam_bytes>, <qtd>, <*stream>);`

```
int vetor[5] = {0,1,2,3,4};  
FILE *arquivo;  
arquivo = fopen("texto.txt", "w");  
fwrite(vetor, sizeof(int), 5, arquivo);  
fclose(arquivo);
```

Leitura

- As principais funções para leitura de dados de um arquivo:
 - **fgetc**: lê um caractere do *stream* (ou 1 byte por vez)
 - **fgets**: lê uma sequência de caracteres, até que um caractere '\n' ou EOF seja encontrado, ou até o número de caracteres passado
 - **fscanf**: similar ao scanf, lê dados de acordo com uma formatação, podendo especificar o tipo de dado que será lido
 - **fread**: lê um bloco de dados do arquivo (array, struct, ...)
 - A leitura é feita a partir do endereço no ponteiro do arquivo, cujo stream é incrementado a cada leitura
-

Leitura

- `fgetc` faz a leitura de 1 byte no *stream* passado.

`fgetc(<*stream>) ;`

Leitura

- **fgetc** faz a leitura de 1 byte no *stream* passado.

fgetc(<*stream>) ;

```
FILE *arquivo;  
arquivo = fopen("texto.txt", "r");  
char c = fgetc(arquivo) ;  
printf("peguei o caractere %c", c);  
fclose(arquivo);
```

Leitura

- **fgets** faz a leitura de sequência de caracteres até uma quebra de linha ('\n') ou até a quantidade de caracteres passada, e armazena em um array de caracteres

`fgets(<*p>, <qtd>, <*stream>);`

Leitura

- **fgets** faz a leitura de sequência de caracteres até uma quebra de linha ('\n') ou até a quantidade de caracteres passada, e armazena em um array de caracteres


fgets(<*p>, <qtd>, <*stream>);

```
FILE *arquivo;  
arquivo = fopen("texto.txt", "r");  
char frase[20];  
fgets(frase, 20, arquivo);  
printf("peguei até 19 caracteres: %s", frase);  
fclose(arquivo);
```

Leitura

- **fgets** faz a leitura de sequência de caracteres até uma quebra de linha ('\n') ou até a quantidade de caracteres passada, e armazena em um array de caracteres

`fgets(<*p>, <qtd>, <*stream>);`

```
FILE *arquivo;  
arquivo = fopen("texto.txt", "r");  
char frase[20];  Lembre-se que o último caractere de uma string é o '/0'.  
fgets(frase, 20, arquivo);  
printf("peguei até 19 caracteres: %s", frase);  
fclose(arquivo);
```

Leitura

- **fscanf** faz a leitura de dados do arquivo e armazena na variável passada de acordo com a formatação passada

fscanf(<*stream>, <formatação>, <*var>);

Leitura

- **fscanf** faz a leitura de dados do arquivo e armazena na variável passada de acordo com a formatação passada

fscanf(<*stream>, <formatação>, <*var>);

```
FILE *arquivo;  
arquivo = fopen("texto.txt", "r");  
float x;  
fscanf(arquivo, "%f", &x);  
printf("valor que peguei do arquivo: %f", x);  
fclose(arquivo);
```

Leitura

- **fread** faz a leitura de um bloco de dados (array, struct, ...) de determinado tamanho e armazena esses dados no endereço de memória passado

```
fread(<*p>, <tam_bytes>, <qtd>, <*stream>);
```

Leitura

- **fread** faz a leitura de um bloco de dados (array, struct, ...) de determinado tamanho e armazena esses dados no endereço de memória passado

fread(<*p>, <tam_bytes>, <qtd>, <*stream>);

```
FILE *arquivo;  
arquivo = fopen("texto.txt", "r");  
float vetor[5];  
fread(vetor, 4, 5, arquivo);  
fclose(arquivo);
```

Exemplo

- Faça um programa que conte o número de quebras de linhas de um arquivo de texto.
-

Exemplo

```
int main()  
{
```

```
    int n, c;
```

```
    FILE *fp = fopen("texto.txt", "r");
```

```
    if (fp == NULL)
```

```
    {
```

```
        printf("erro de abertura");
```

```
        return 1;
```

```
    }
```

```
    c = fgetc(fp);
```

- Faça um programa que conte o número de quebras de linhas de um arquivo de texto.
-

Exemplo

- Faça um programa que conte o número de quebras de linhas de um arquivo de texto.

```
int main()
{
    int n, c;
    FILE *fp = fopen("texto.txt", "r");
    if (fp == NULL)
    {
        printf("erro de abertura");
        return 1;
    }
    c = fgetc(fp);

    while(c!= EOF)
    {
        if (c=='\n')
        {
            n++;
        }
        c = fgetc(fp);
    }
    fclose(fp);
}
```

Exemplo

- Faça um programa que receba o nome de uma pessoa e grave num arquivo de texto.
-

Exemplo

```
int main()
{
    char nome[20];
    FILE *fp = fopen("texto.txt",
"a+");
    if (fp == NULL)
    {
        printf("erro de abertura");
        return 1;
    }
    printf("Digite um nome: ");
    scanf("%s", &nome);
    fprintf(fp, "%s\n", nome);
    fclose(fp);
}
```

- Faça um programa que receba o nome de uma pessoa e grave num arquivo de texto.
-

Exercícios

- Faça um programa para contar quantas letras minúsculas ou maiúsculas um arquivo de texto tem.
 - Escreva um programa para receber do usuário o nome de 5 pessoas e guardá-los em um arquivo de texto.
 - Modifique o programa acima para guardar nomes até que o usuário diga que não há mais nomes a serem inseridos.
 - Faça um programa para guardar uma estrutura com nome e idade de 10 pessoas em um arquivo de texto.
-