

## Arquivos: entrada e saída de dados

O uso de arquivos pode ser indicado em vários casos, por exemplo:

- quando é necessário armazenar os dados entre as rodadas do programa
- quando o volume de dados é maior do que a capacidade da RAM
- quando os dados de entrada para o programa são gerados de outro programa ou dispositivo.

Existem dois tipos de arquivos: binários ou texto. Em ambos os casos um arquivo é uma sequência de bytes.

A linguagem C/C++ permite realização de várias operações com arquivos de alto e baixo nível.

### Abertura de arquivo

Para abrir um arquivo ou criar um arquivo novo é usada função **fopen()**. Quando chamada, essa função inicializa um objeto do tipo FILE, que contém a informação necessária para controle de entrada/saída de dados.

O protótipo da função **fopen()**:

```
FILE *fopen( const char * filename, const char * mode );
```

onde:

**filename** – é uma **string** que contém o nome do arquivo

**mode** – é uma variável que define a forma de acesso (as operações permitidas)

Mode	Descrição
r	read – abre um arquivo texto existente para leitura
w	write – abre um arquivo texto para entrada de dados. Caso o arquivo não existe – ele é criado. O programa começa a escrever os dados no início do arquivo.
a	append – abre o arquivo para acrescentar os dados. Caso o arquivo não existe – ele é criado. O programa começa a escrever os dados no final do arquivo.
r+	abre o arquivo texto para entrada e saída de dados.
w+	abre o arquivo texto para entrada e saída de dados. Primeiro o tamanho do arquivo é zerado (caso o arquivo tenha tamanho diferente de zero), se o arquivo não existia ele é criado.
a+	abre o arquivo texto para entrada e saída de dados. A leitura começa do início do arquivo, mas os dados serão acrescentados no final do arquivo.

No caso dos arquivos binários o modo de acesso será modificado para:

```
rb, wb, ab, rb+, r+b, wb+, w+b, ab+, a+b
```

### Fechamento de arquivo

O arquivo deve ser fechado usando a função **fclose()**.  
O protótipo dessa função tem a forma:

```
int fclose( FILE *fp );
```

Essa função retorna zero em caso de sucesso ou **EOF** se ocorrer um error na hora de fechar o arquivo.

A função passa todos os dados pendentes para o arquivo, fecha o arquivo e libera a memória.

**EOF** – é uma constante definida em **stdio.h**.

### Escrita de dados para arquivo

Para escrever os caracteres individuais pode ser usada a função:

```
int fputc( int c, FILE *fp );
```

Essa função escreve o valor do **c** para o arquivo referenciado pelo **fp**.  
Retorna o valor do caractere escrito em caso de sucesso ou **EOF** em caso de falha.

Para escrever uma sequencia de caracteres (que termina com null) pode ser usada a função:

```
int fputs( const char *s, FILE *fp );
```

Essa função escreve o string **s** para o arquivo referenciado pelo **fp**.  
Vai retornar o valor não nulo em caso de sucesso ou **EOF** em caso de falha.

A função **fprintf()** também pode ser usada para gravar uma sequencia de caracteres para o arquivo.

```
int fprintf(FILE *fp,const char *format, ...)
```

### Leitura de dados do arquivo

Para ler um caractere do arquivo é usada função **fgetc()**:

```
int fgetc( FILE * fp );
```

A função retorna o caractere lido ou **EOF** em caso de falha.

Para fazer a leitura de uma string de caracteres:

```
char *fgets( char *buf, int n, FILE *fp );
```

A função vai tentar fazer a leitura de **n-1** caracteres do arquivo referenciado pelo **fp**.  
Ela copia a sequencia lida para **buf** e acrescenta o caractere **null** para indicar o fim da sequencia de caracteres.

Se a função encontrar o símbolo de nova linha '\n' ou o fim do arquivo EOF antes de processar n-1 caracteres, ela vai copiar somente caracteres encontrados até esse momento.

Outra função que pode ser usada para leitura de dados é:

```
int fscanf(FILE *fp, const char *format, ...)
```

Essa função vai fazer a leitura de dados de forma similar, porém vai parar a leitura se encontrar um espaço em branco(' ').

### Funções de entrada/saída de dados para arquivos binários

Para fazer a entrada/saída de dados para arquivos binários são usadas funções:

```
size_t fread(void *ptr, size_t size_of_elements, size_t number_of_elements, FILE *a_file);
```

```
size_t fwrite(const void *ptr, size_t size_of_elements, size_t number_of_elements, FILE *a_file);
```

#### Exemplo 1 (4p): Criação do arquivo output.txt

```
1  #include<stdio.h>
2  //criação do arquivo output.txt
3
4  int main()
5  {
6      FILE *filePtr;
7      int i;
8
9      // criar um arquivo
10     filePtr =fopen("output.txt", "w");
11
12     if (filePtr == NULL)
13     {
14         printf("\n Erro! Não foi possível criar arquivo! \n ");
15         return 1; // código de erro
16     }
17
18     // gravar dados para arquivo
19     for (i = 1; i <= 10; i++)
20         fprintf(filePtr,"%d\n", i);
21
22     // fechar o arquivo
23     printf("\n O programa gravou números de [1,10] para arquivo output.txt \n ");
24     fclose(filePtr);
25
26     return 0;
27 }
```

O programa gravou números de [1,10] para arquivo output.txt

### Exemplo 2 (4p): Leitura de dados do arquivo **output.txt**

```
1  #include<stdio.h>
2  // leitura de dados do arquivo output.txt
3  // o arquivo output.txt deve estar presente na pasta atual
4  int main()
5  {
6      FILE *in_fPtr;
7      int num;
8
9      // abertura do arquivo para leitura
10     in_fPtr =fopen("output.txt", "r");
11
12     if (in_fPtr == NULL)
13     {
14         printf("\n Erro! Não foi possível abrir o arquivo! \n ");
15         return 1; // código de erro
16     }
17
18     // leitura de dados do arquivo
19     printf("\n Leitura de dados do arquivo output.txt \n");
20     while( fscanf(in_fPtr, "%d", &num ) == 1 )
21         printf("Número: %d\n", num);
22
23     // fechar o arquivo
24     fclose(in_fPtr);
25
26     return 0;
27 }
```

Leitura de dados do arquivo output.txt

Número: 1  
Número: 2  
Número: 3  
Número: 4  
Número: 5  
Número: 6  
Número: 7  
Número: 8  
Número: 9  
Número: 10

### Exemplo 3 (4p): Criação do arquivo **alunos.txt**

```
1 //criação do arquivo alunos.txt
2 #include<stdio.h>
3
4 int main()
5 {
6     FILE *filePtr;
7     int id, i = 1, ch;
8     float n;
9     char name[100];
10
11     // criar um arquivo
12     filePtr =fopen("alunos.txt", "w");
13
14     if (filePtr == NULL)
15     {
16         printf("\n Erro! Não foi possivel criar arquivo! \n ");
17         return 1; // código de eero
18     }
19
20     // ler dados e gravar para arquivo
21     printf("\n Programa grava dados para arquivo alunos.txt \n ");
22
23     while(i != 0)
24     {
25         printf("\n Digite matricula do aluno: ");
26         scanf("%i",&id);
27
28         // fflush(stdin);
29         printf("\n Digite nome do aluno: ");
30         scanf(" %[^\\t\\n]s",name);
31
32         printf("\n Digite a nota do aluno: ");
33         scanf("%f",&n);
34
35         printf("\n Voce digitou: ");
36         printf("\n Matricula: %i", id);
37         printf("\n Nome: %s", name);
38         printf("\n Nota: %.2f", n);
39
40         printf("\n Gravara para arquivo? 1- Sim; 0 - Não : ");
41         scanf("%i",&ch);
```

```

42
43     if( ch == 1)
44     { // gravar dados para arquivo
45         fprintf(filePtr, "%d\n", id);
46         fprintf(filePtr, "%s\n", name);
47         fprintf(filePtr, "%.2f\n", n);
48     }
49
50     printf("\n Deseja continuar? 1 - Sim; 0 - Não : ");
51     scanf("%i",&i);
52 }
53
54 fclose(filePtr);
55 return 0;
56 }

```

#### Exemplo 4 (4p): Leitura de dados do arquivo alunos.txt

```

1 //leitura do arquivo alunos.txt
2 #include<stdio.h>
3
4 int main()
5 {
6     FILE *filePtr;
7     int id;
8     float n;
9     char name[100];
10
11     // criar um arquivo
12     filePtr =fopen("alunos.txt", "r");
13
14     if (filePtr == NULL)
15     {
16         printf("\n Erro! Não foi possivel abrir arquivo! \n ");
17         return 1; // código de erro
18     }
19
20     // ler dados e gravar para arquivo
21     printf("\n Programa faz a leitura de dados do arquivo alunos.txt \n ");
22
23     while( fscanf(filePtr, "%i", &id) != EOF )
24     {
25         fscanf(filePtr, "%i", &id);
26         fgets(name, 100, filePtr );
27         fscanf(filePtr, "%f", &n);
28
29         printf("\n Leitura de dados: ");
30         printf("\n Matricula: %i", id);
31         printf("\n Nome: %s", name);
32         printf(" Nota: %.2f \n\n", n);
33     }
34
35     fclose(filePtr);
36     return 0;
37 }

```