

# Modularização- Arquivos

Prof. Gabriel Barbosa da Fonseca

Adaptado de Rosilane Mota

## Conceitos Básicos

---

Conjuntos de informações mantidas no disco (memória secundária)

Informações são “persistidas” em comparação com a memória RAM (memória primária) que eram “temporárias”

Sistema Operacional (OS) faz um “buffer” das informações lidas/gravadas

---

---

Em C, arquivos podem ser gravados de duas maneiras:

- Modo texto (conjunto de caracteres)  
Arquivo texto pode ser tratado por qualquer editor (e.g., bloco de notas, terminal, word, etc.)
  - Modo binário (conjunto de bytes)  
Ex: Grandes quantidades de informações de forma eficiente
-

## Operações

---

- **Abertura do arquivo** (localização, alocação do buffer)
  - **Leitura do arquivo** (informações do buffer são disponibilizadas)
  - **Gravação do arquivo** (alteração de dados preexistentes ou acréscimo de novos dados)
  - **Fechamento do arquivo** (atualização do buffer e liberação de memória alocada)
- 

---

Para trabalhar com arquivos, a linguagem C oferece um pacote de funções de biblioteca divididas em grupos:

- **Grupo1:** Ler e escrever um caractere por vez (funções `fputc()` e `fgetc()`)
  - **Grupo2:** Ler e escrever linha a linha (funções `fputs()` e `fgets()`)
  - **Grupo3:** Ler e escrever dados formatados (funções `fprintf()` e `fscanf()`)
-

# Modos de Acesso

modo_de_acesso	Significado
r	Abre o arquivo somente para leitura. O arquivo deve existir.
r+	Abre o arquivo para leitura e escrita. O arquivo deve existir.
w	Abre o arquivo somente para escrita no início do arquivo. Apagará o conteúdo do arquivo se ele já existir, criará um arquivo novo se ele não existir.
w+	Abre o arquivo para escrita e leitura, apagando o conteúdo pré-existente.
a	Abre o arquivo para escrita no final do arquivo. Não apagará o conteúdo pré-existente.
a+	Abre o arquivo para escrita no final do arquivo e leitura.

# Abrindo arquivos

---

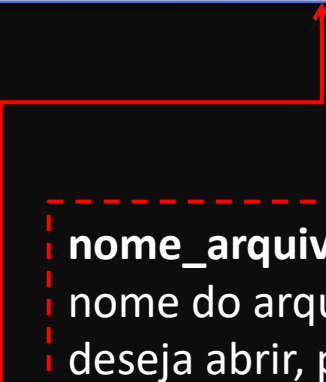
A função para abrir arquivos é a `fopen()`  
Essa função executa duas tarefas:

- Cria e preenche uma estrutura `FILE`, com as informações necessárias
- Retorna um ponteiro do tipo `FILE` que aponta para a localização na memória dessa estrutura criada

Resumindo, a função `fopen()` abre um arquivo, retornando o ponteiro associado ao arquivo

---

```
FILE * arq = fopen (nome_arquivo, modo_abertura);
```



**nome\_arquivo:** representa o nome do arquivo que se deseja abrir, podendo conter inclusive o caminho.

**modo\_abertura:** representa como o arquivo será aberto.

---

## Exemplo 1

---

Programa que cria o arquivo1.txt para escrita no mesmo diretório em que o projeto está sendo executado.

---

```
#include <stdio.h>

int main()
{
    FILE *arquivo;
    arquivo = fopen("arquivo1.txt","w");
    printf("Arquivo criado.");
    return 0;
}
```

---

## Exemplo 2

---

Programa que cria o arquivo2.txt para escrita no diretório c:\temp\

Se diretório temp não existe, fopen não indica erro ao criar e não cria o arquivo.

O diretório pode ser criado com o comando **system** e com os argumentos contendo os comandos do **prompt** como o “make dir” (md).

```
system("md c:\\temp");
```

---

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

```
int main()
```

```
{
```

```
    FILE *arquivo;
```

```
    system("md c:\\temp"); // Tem que ser \\
```

```
    arquivo = fopen("c:/temp/arquivo2.txt","w");
```

```
    printf("Arquivo 2 criado.");
```

```
    return 0;
```

```
}
```

---

## Exemplo 3

---

Programa que cria o arquivo3.txt para escrita no diretório c:\temp\

Pode-se utilizar \\ para indicar apenas uma \ e não confundir com caracteres especiais.

---

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

```
int main()
```

```
{
```

```
    FILE *arquivo;
```

```
    arquivo = fopen("c:\\temp\\arquivo3.txt","w");
```

```
    printf("Arquivo 3 criado.");
```

```
    return 0;
```

```
}
```

---



## Um caractere por vez

---

A função `fputc()` escreve um caractere por vez. Essa função retorna o caractere gravado ou EOF se acontecer algum erro.

```
int fputc (char ch, FILE *arq);
```

---

A função `fgetc()` realiza a leitura a partir de um arquivo texto ou EOF se não for possível ler.

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

---

## Exemplo 4

---

Programa que cria o arquivo4.txt para escrita no diretório do código

Grava “teste” no arquivo e “1” em outra linha, encerrando a manipulação deste modo de escrita.

---

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

int main()
{
    FILE *arquivo;
    arquivo = fopen("arquivo4.txt","w");
    fputc('t',arquivo);
    fputc('e',arquivo);
    fputc('s',arquivo);
    fputc('t',arquivo);
    fputc('e',arquivo);
    fputc('\n',arquivo);
    fputc('1',arquivo);
    fclose(arquivo);

    printf("Arquivo 4 criado.");
    return 0;
}
```

## Exemplo 5

---

Programa que lê o arquivo4.txt existente no diretório do código

---

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

int main()
{
    FILE *arquivo;
    char letra;
    arquivo = fopen("arquivo4.txt","r");
    while ((letra = fgetc(arquivo))!= EOF)
    {
        printf("%c",letra);
    }
    fclose(arquivo);
    return 0;
}
```

## Exemplo 6

---

Programa que lê o arquivo4.txt existente no diretório do código

---

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

int main()
{
    FILE *arquivo;
    char letra;
    arquivo = fopen("arquivo4.txt","r");
    while (!feof(arquivo))
    {
        letra = fgetc(arquivo);
        printf("%c",letra);
    }
    fclose(arquivo);
    return 0;
}
```

## Dados formatados

---

A função `fprintf( )` escreve no arquivo uma sequência de dados formatados, retornando a quantidade de caracteres escritos.

É similar à função `printf()`, exceto pelo fato de que um ponteiro para `FILE` deverá ser incluído como primeiro argumento.

```
int fprintf ( FILE * f, const char * formato,  
             [argumentos] );
```

A função `fscanf( )` faz a leitura do arquivo uma sequência de dados formatados, retornando a quantidade de itens lidos.

É similar à função `scanf()`, exceto pelo fato de que um ponteiro para `FILE` deverá ser incluído como primeiro argumento.

```
int fscanf ( FILE * f, const char * formato,  
            [argumentos] );
```

## Exemplo 7

---

Programa que escreve no arquivo5.txt no diretório do código

---

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

int main()
{
    FILE *arquivo;
    int a = 5, b = 6;
    arquivo = fopen("arquivo5.txt", "w");
    fprintf(arquivo, "%d %d\n", a, b);
    fclose(arquivo);
    return 0;
}
```

## Exemplo 8

---

Programa que lê do arquivo5.txt  
existente no diretório do código

---

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

int main()
{
    FILE *arquivo;
    int a = 5, b = 6;
    arquivo = fopen("arquivo5.txt", "r");
    while(fscanf(arquivo, "%d %d", &a, &b) != EOF)
    {
        printf("Leu %d e %d\n", a, b);
    }
    fclose(arquivo);
    return 0;
}
```

# Modularização- Arquivos

Prof. Gabriel Barbosa da Fonseca

Adaptado de Rosilane Mota