



Arquivos em linguagem C

Luciano Antunes



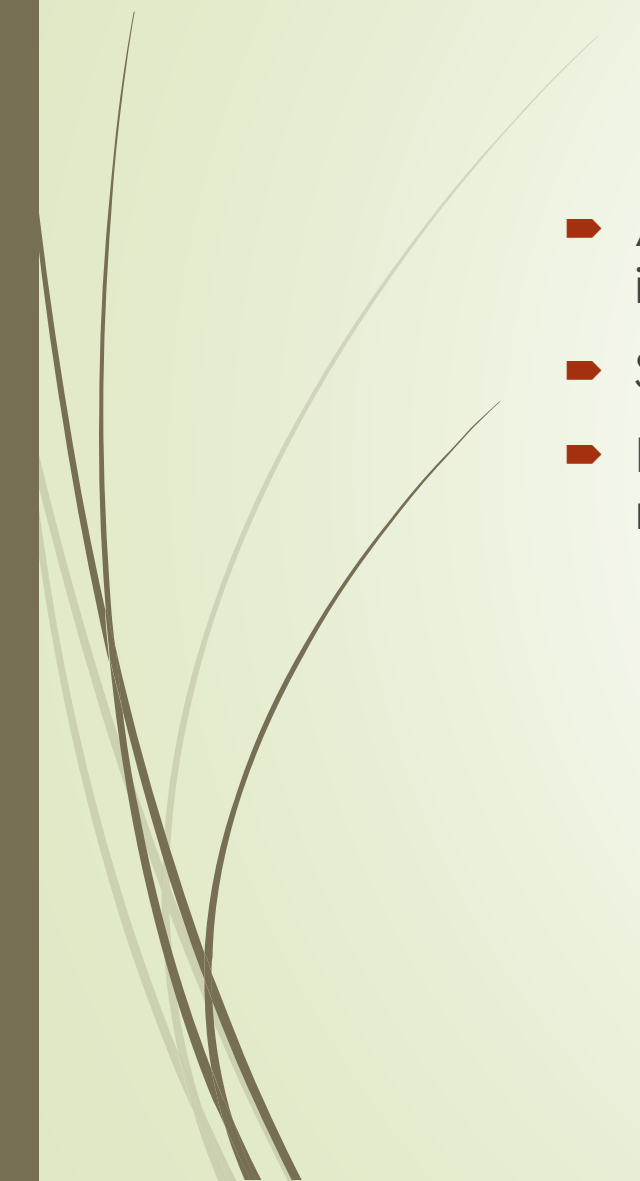
Manipulando arquivos em C

- A linguagem C possui uma série de funções para manipulação de arquivos, cujos protótipos estão reunidos na biblioteca padrão de entrada e saída:

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




Manipulando arquivos em C

- A linguagem C não possui funções que automaticamente leiam todas as informações de um arquivo.
 - Suas funções se limitam a abrir/fechar e ler caracteres/bytes
 - É tarefa do programador criar a função que lerá um arquivo de uma maneira específica.
- 



Manipulando arquivos em C

- Todas as funções de manipulação de arquivos trabalham com o conceito de "ponteiro de arquivo".
- Um ponteiro de arquivo pode ser declarado da seguinte forma:
 - `FILE *p;`
- `p` é o ponteiro para arquivos que permite manipular arquivos no C.




Arquivo - abertura

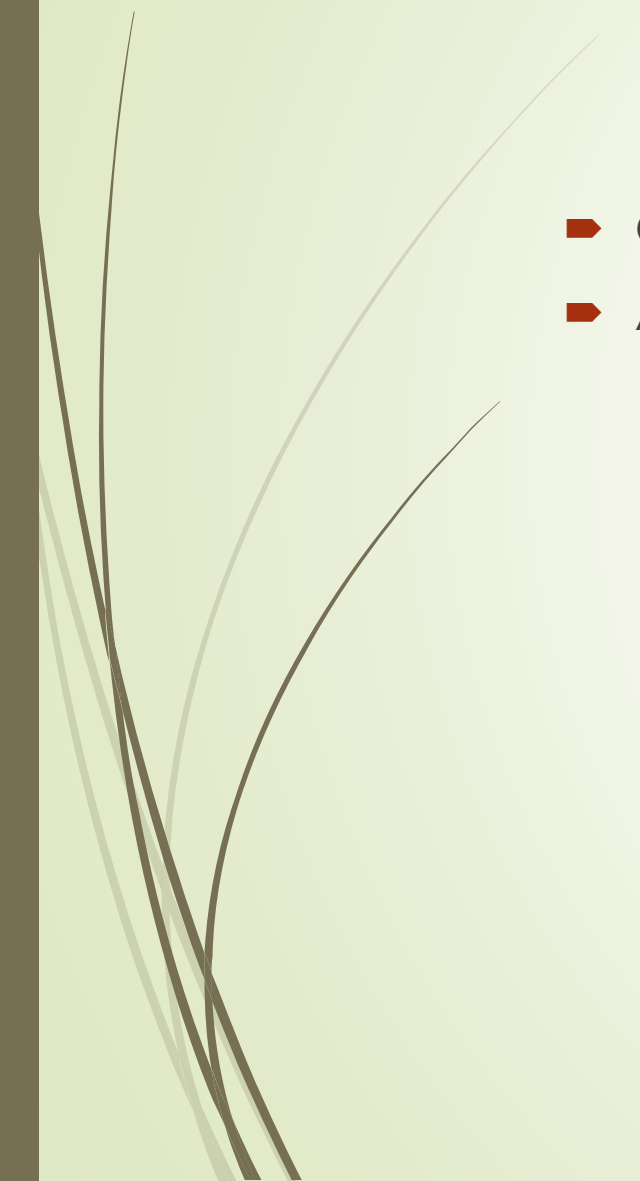
- Para a abertura de um arquivo, usa-se a função fopen

< ponteiro > = fopen(“nome do arquivo”, “tipo de abertura”);

- O parâmetro arquivo determina qual arquivo deverá ser aberto, sendo que o mesmo deve ser válido no sistema operacional que estiver sendo utilizado.



Arquivo - abertura

- O modo de abertura determina que tipo de uso será feito do arquivo.
 - A tabela a seguir mostra os modo válidos de abertura de um arquivo.
- 

Arquivo - 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.



O ponteiro para arquivo

Em C, o arquivo é manipulado através de um ponteiro especial para o arquivo.

A função deste ponteiro é “apontar” a localização de um registro.

Sintaxe:

FILE < *ponteiro >

O tipo FILE está definido na biblioteca stdio.h.

Exemplo de declaração de um ponteiro para arquivo em C:

FILE *pont_arq;

Lembrando que FILE deve ser escrito em letras maiúsculas.



Fechamento de arquivo

- Sintaxe de fechamento de arquivo
- **`fclose< ponteiro >;`**



Problemas na abertura de arquivos

Na prática, nem sempre é possível abrir um arquivo. Podem ocorrer algumas situações que impedem essa abertura, por exemplo:

- Você está tentando abrir um arquivo no modo de leitura, mas o arquivo não existe;
- Você não tem permissão para ler ou gravar no arquivo;
- O arquivo está bloqueado por estar sendo usado por outro programa.
- Quando o arquivo não pode ser aberto a função `fopen` retorna o valor `NULL`.
- É altamente recomendável criar um trecho de código a fim de verificar se a abertura ocorreu com sucesso ou não.



Problemas na abertura de arquivos

Exemplo:

```
if (pont_arq == NULL)
{
    printf("ERRO! O arquivo não foi aberto!\n");
}
else
{
    printf("O arquivo foi aberto com sucesso!");
}
```

Gravando dados em arquivos

- A função **fprintf** armazena dados em um arquivo. Seu funcionamento é muito semelhante ao printf, a diferença principal é a existência de um parâmetro para informar o arquivo onde os dados serão armazenados.
- **Sintaxe:**

fprintf(nome_do_ponteiro_para_o_arquivo, “%s”,variavel_string)




Leitura de arquivos

- **Leitura caracter por caracter – Função `getc()`**
- Faz a leitura de um caracter no arquivo.
- Sintaxe:
- `getc(ponteiro_do_arquivo);`
- Para realizar a leitura de um arquivo inteiro caracter por caracter podemos usar `getc` dentro de um laço de repetição.



Leitura de Arquivos - exemplo

```
do {  
    //faz a leitura do character no arquivo apontado por pont_arq  
    c = getc(pont_arq);  
    //exibe o character lido na tela  
    printf("%c" , c);  
}while (c != EOF);
```



Leitura de strings – Função fgets()

- É utilizada para leitura de strings em um arquivo. Realiza a leitura dos caracteres até o final da linha quando encontra o caracter \n.
- A leitura é efetuada de tal forma que a string lida é armazenada em um ponteiro do tipo char.
- A função pode ser finalizada quando encontrar o final do arquivo, neste caso retorna o endereço da string lida. Se ocorrer algum erro na leitura do arquivo, a função retorna NULL.

Leitura de strings – Função fgets()

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    FILE *pont_arq;
    char texto_str[20];
    //abrindo o arquivo_frase em modo "somente leitura"
    pont_arq = fopen("arquivo_palavra.txt", "r");
    //enquanto não for fim de arquivo o looping será executado
    //e será impresso o texto
    while(fgets(texto_str, 20, pont_arq) != NULL)
        printf("%s", texto_str);
    //fechando o arquivo
    fclose(pont_arq);
    getch();
    return(0);
}
```