# LPG0002 – Linguagem de Programação

## Manipulação de Arquivos de Texto

Profª Luciana Rita Guedes
Departamento de Ciência da Computação
UDESC / Joinville

Material elaborado por: Prof. Rui Jorge Tramontin Junior

# Introdução

- A linguagem C oferece funções para leitura e gravação no disco;

# Introdução

- A linguagem C oferece funções para leitura e gravação no disco;

- Tais operações são executadas sobre **arquivos**;

# Introdução

- A linguagem C oferece funções para leitura e gravação no disco;
- Tais operações são executadas sobre **arquivos**;
- Arquivos podem ser manipulados de diversas formas, e podem ser abertos de dois modos:

# Introdução

- A linguagem C oferece funções para leitura e gravação no disco;
- Tais operações são executadas sobre **arquivos**;
- Arquivos podem ser manipulados de diversas formas, e podem ser abertos de dois modos:
  - Modo texto;

# Introdução

- A linguagem C oferece funções para leitura e gravação no disco;
- Tais operações são executadas sobre **arquivos**;
- Arquivos podem ser manipulados de diversas formas, e podem ser abertos de dois modos:
  - Modo texto;
  - Modo binário.

# Estrutura FILE

- As informações a respeito de um arquivo aberto por um programa em C ficam armazenadas em uma estrutura do tipo **`FILE`**;

;

# Estrutura FILE

- As informações a respeito de um arquivo aberto por um programa em C ficam armazenadas em uma estrutura do tipo **`FILE`**;

- A estrutura **`FILE`** é oferecida pela biblioteca ***stdio.h***;

# Estrutura FILE

- As informações a respeito de um arquivo aberto por um programa em C ficam armazenadas em uma estrutura do tipo **`FILE`**;

- A estrutura **`FILE`** é oferecida pela biblioteca ***stdio.h***;

- A função **`fopen()`** é utilizada para abrir um arquivo, e retorna um ponteiro para **`FILE`**;

# Estrutura FILE

- As informações a respeito de um arquivo aberto por um programa em C ficam armazenadas em uma estrutura do tipo **`FILE`**;
- A estrutura **`FILE`** é oferecida pela biblioteca ***stdio.h***;
- A função **`fopen()`** é utilizada para abrir um arquivo, e retorna um ponteiro para **`FILE`**;
- Tal ponteiro é utilizado pelo programa para manipular o arquivo.

# Função *fopen()*

```
FILE *fopen( char *nome_arq, char *modo );
```

# Função *fopen()*

```
FILE *fopen( char *nome_arq, char *modo );
```

- **`nome_arq`** é uma *string* que contém o nome do arquivo a ser aberto;

# Função *fopen()*

```
FILE *fopen( char *nome_arq, char *modo );
```

- **`nome_arq`** é uma *string* que contém o nome do arquivo a ser aberto;
- Caso tenha somente o nome do arquivo, significa que o mesmo deve estar no mesmo diretório do programa;
  - Ex.: `"saida.txt"`

# Função *fopen()*

```
FILE *fopen( char *nome_arq, char *modo );
```

- **`nome_arq`** é uma *string* que contém o nome do arquivo a ser aberto;
- Caso tenha somente o nome do arquivo, significa que o mesmo deve estar no mesmo diretório do programa;
  - Ex.: `"saida.txt"`
- É possível também fazer uma referência absoluta ao nome do arquivo;
  - Ex. `"C:\\temp\\saida.txt"`

# Modos de abertura de arquivo

- `"r"` Leitura (*read*). O arquivo deve existir;

# Modos de abertura de arquivo

- `"r"` Leitura (*read*). O arquivo deve existir;

- `"w"` Gravação (*write*). Se existir, o arquivo será destruído e reinicializado; se não existir, será criado;

# Modos de abertura de arquivo

- `"r"` Leitura (*read*). O arquivo deve existir;
- `"w"` Gravação (*write*). Se existir, o arquivo será destruído e reinicializado; se não existir, será criado;
- `"a"` Gravação em anexo (*append*). Os dados são adicionados ao final do arquivo existente; ou um novo arquivo é criado.

# Modificadores dos modos de abertura

Além dos modos, os seguintes modificadores podem ser utilizados:

# Modificadores dos modos de abertura

Além dos modos, os seguintes modificadores podem ser utilizados:

- `"+"` Permite acesso de leitura e escrita;

# Modificadores dos modos de abertura

Além dos modos, os seguintes modificadores podem ser utliizados:

- `"+"` Permite acesso de leitura e escrita;
- `"b"` Abre em *modo binário*;

# Modificadores dos modos de abertura

Além dos modos, os seguintes modificadores podem ser utilizados:

- `"+"` Permite acesso de leitura e escrita;
- `"b"` Abre em *modo binário*;
- `"t"` Abre em *modo texto*.

# Exemplos

```
// Abre em modo texto para leitura

FILE *f = fopen( "entrada.txt", "rt" );
```

# Exemplos

```
// Abre em modo texto para leitura

FILE *f = fopen( "entrada.txt", "rt" );


// Abre em modo texto para append

FILE *f = fopen( "log.txt", "at" );
```

# Considerações sobre *fopen()*

- Caso o arquivo possa ser aberto, a função retorna um ponteiro para **`FILE`**;

# Considerações sobre *fopen()*

- Caso o arquivo possa ser aberto, a função retorna um ponteiro para **`FILE`**;

- Caso contrário, retorna ***NULL*** (***é preciso testar!***);

# Considerações sobre *fopen()*

- Caso o arquivo possa ser aberto, a função retorna um ponteiro para **`FILE`**;
- Caso contrário, retorna ***NULL*** (***é preciso testar!***);
- Alguns fatores que levam a erro na abertura:

# Considerações sobre *fopen()*

- Caso o arquivo possa ser aberto, a função retorna um ponteiro para **`FILE`**;
- Caso contrário, retorna ***NULL*** (***é preciso testar!***);
- Alguns fatores que levam a erro na abertura:
  - Arquivo inexistente (no caso de leitura);

# Considerações sobre *fopen()*

- Caso o arquivo possa ser aberto, a função retorna um ponteiro para **`FILE`**;
- Caso contrário, retorna ***NULL*** (***é preciso testar!***);
- Alguns fatores que levam a erro na abertura:
  - Arquivo inexistente (no caso de leitura);
  - Espaço insuficiente em disco (para gravação);

# Considerações sobre *fopen()*

- Caso o arquivo possa ser aberto, a função retorna um ponteiro para **`FILE`**;

- Caso contrário, retorna ***NULL*** (***é preciso testar!***);

- Alguns fatores que levam a erro na abertura:
  - Arquivo inexistente (no caso de leitura);
  - Espaço insuficiente em disco (para gravação);
  - S.O. não dá permissão de acesso ao arquivo.

# Fechamento do arquivo

- Após o uso, o arquivo deve ser fechado;

# Fechamento do arquivo

- Após o uso, o arquivo deve ser fechado;
- O fechamento é fundamental, sobretudo após operações de gravação;

# Fechamento do arquivo

- Após o uso, o arquivo deve ser fechado;
- O fechamento é fundamental, sobretudo após operações de gravação;
- Para fechar um aquivo, basta chamar a função `fclose()`, passando o ponteiro `FILE` como parâmetro;

# ARQUIVOS DE TEXTO

# Arquivos de texto

- Quando aberto em modo texto, um arquivo é interpretado como sequências de caracteres agrupadas em linhas;

# Arquivos de texto

- Quando aberto em modo texto, um arquivo é interpretado como sequências de caracteres agrupadas em linhas;
  - Mudança de linha é feita pelo caractere *line feed* (`'\r'`);

# Arquivos de texto

- Quando aberto em modo texto, um arquivo é interpretado como sequências de caracteres agrupadas em linhas;
  - Mudança de linha é feita pelo caractere *line feed* (`'\r'`);

- Em sistemas DOS/Windows, a mudança de linha pode ser feita por dois caracteres;
  - *carriage return* (`'\n'`) e *line feed* (`'\r'`);

# Arquivos de texto

- Quando aberto em modo texto, um arquivo é interpretado como sequências de caracteres agrupadas em linhas;
  - Mudança de linha é feita pelo caractere *line feed* (`'\r'`);
- Em sistemas DOS/Windows, a mudança de linha pode ser feita por dois caracteres;
  - *carriage return* (`'\n'`) e *line feed* (`'\r'`);
- Entretanto, o código em C trata isso de forma independente (convertendo para somente um caractere durante o processamento).

# Funções de Manipulação

- Entrada/saída **formatada**;
  - Funções `fscanf()` e `fprintf();`

# Funções de Manipulação

- Entrada/saída **formatada**;
  - Funções `fscanf()` e `fprintf();`

- Entrada/saída por **linha** (***strings***);
  - Funções `fgets()` e `puts()`;

# Funções de Manipulação

- Entrada/saída **formatada**;
  - Funções `fscanf()` e `fprintf();`
- Entrada/saída por **linha** (***strings***);
  - Funções `fgets()` e `puts()`;
- Entrada/saída por **caractere individual**;
  - Funções `getc()` e `putc()`.

# Entrada/Saída Formatada

- As funções `fscanf()` e `fprintf()` são variações das funções já conhecidas para manipulação via *console*;

# Entrada/Saída Formatada

- As funções `fscanf()` e `fprintf()` são variações das funções já conhecidas para manipulação via *console*;

- A diferença é que há um parâmetro a mais (1º parâmetro) que é o ponteiro para `FILE`;

# Entrada/Saída Formatada

- As funções `fscanf()` e `fprintf()` são variações das funções já conhecidas para manipulação via *console*;

- A diferença é que há um parâmetro a mais (1º parâmetro) que é o ponteiro para `FILE`;

- Portanto, a entrada e saída formatada de arquivos é similar ao que é feito no *console*.

# Exemplo 1

```
float v[] = { 1.5, 2.75, -7, 0, 45.99 };
int i;
```

# Exemplo 1

```
float v[] = { 1.5, 2.75, -7, 0, 45.99 };
int i;

FILE *f = fopen( "exemplo1.txt", "wt" );
```

# Exemplo 1

```
float v[] = { 1.5, 2.75, -7, 0, 45.99 };
int i;

FILE *f = fopen( "exemplo1.txt", "wt" );
if( f == NULL){
   printf("Erro no arquivo!\n");
   return -1;
}
```

# Exemplo 1

```
float v[] = { 1.5, 2.75, -7, 0, 45.99 };
int i;

FILE *f = fopen( "exemplo1.txt", "wt" );
if( f == NULL){
   printf("Erro no arquivo!\n");
   return -1;
}

for( i = 0 ; i < 5 ; i++ )
```

# Exemplo 1

```
float v[] = { 1.5, 2.75, -7, 0, 45.99 };
int i;

FILE *f = fopen( "exemplo1.txt", "wt" );
if( f == NULL){
   printf("Erro no arquivo!\n");
   return -1;
}

for( i = 0 ; i < 5 ; i++ )
   fprintf( f, "%.3f\n", v[i] );
```

# Exemplo 1

```
float v[] = { 1.5, 2.75, -7, 0, 45.99 };
int i;

FILE *f = fopen( "exemplo1.txt", "wt" );
if( f == NULL){
   printf("Erro no arquivo!\n");
   return -1;
}

for( i = 0 ; i < 5 ; i++ )
   fprintf( f, "%.3f\n", v[i] );

fclose( f );
```

# Exemplo 1: arquivo gerado

```
1.500
2.750
-7.000
0.000
45.990
```

# Exemplo 2

```
float valor;
int cont = 0;
```

# Exemplo 2

```
float valor;
int cont = 0;
FILE *f = fopen( "exemplo1.txt", "rt" );
```

# Exemplo 2

```
float valor;
int cont = 0;
FILE *f = fopen( "exemplo1.txt", "rt" );
if( f == NULL){
   printf("Erro no arquivo!\n");
   return -1;
}
```

# Exemplo 2

```
float valor;
int cont = 0;
FILE *f = fopen( "exemplo1.txt", "rt" );
if( f == NULL){
   printf("Erro no arquivo!\n");
   return -1;
}
int x = fscanf( f, "%f", &valor );
```

# Exemplo 2

```
float valor;
int cont = 0;
FILE *f = fopen( "exemplo1.txt", "rt" );
if( f == NULL){
   printf("Erro no arquivo!\n");
   return -1;
}
int x = fscanf( f, "%f", &valor );
while( x == 1 ){



}
```

# Exemplo 2

```
float valor;
int cont = 0;
FILE *f = fopen( "exemplo1.txt", "rt" );
if( f == NULL){
   printf("Erro no arquivo!\n");
   return -1;
}
int x = fscanf( f, "%f", &valor );
while( x == 1 ){
   cont++;
   printf( "%d° valor: %.3f\n", cont, valor );

}
```

# Exemplo 2

```
float valor;
int cont = 0;
FILE *f = fopen( "exemplo1.txt", "rt" );
if( f == NULL){
   printf("Erro no arquivo!\n");
   return -1;
}
int x = fscanf( f, "%f", &valor );
while( x == 1 ){
   cont++;
   printf( "%d° valor: %.3f\n", cont, valor );
   x = fscanf( f, "%f", &valor );
}
```

# Exemplo 2

```
float valor;
int cont = 0;
FILE *f = fopen( "exemplo1.txt", "rt" );
if( f == NULL){
   printf("Erro no arquivo!\n");
   return -1;
}
int x = fscanf( f, "%f", &valor );
while( x == 1 ){
   cont++;
   printf( "%d° valor: %.3f\n", cont, valor );
   x = fscanf( f, "%f", &valor );
}
fclose( f );
```

# Exemplo 2

```
float valor;
int cont = 0;
FILE *f = fopen( "exemplo1.txt", "rt" );
if( f == NULL){
   printf("Erro no arquivo!\n");
   return -1;
}
int x = fscanf( f, "%f", &valor );
while( x == 1 ){
   cont++;
   printf( "%d° valor: %.3f\n", cont, valor );
   x = fscanf( f, "%f", &valor );
}
fclose( f );
printf("Foram lidos %d valores.\n", cont);
```

# Exemplo 2: saída no *console*

```
1° valor: 1.500
2° valor: 2.750
3° valor: -7.000
4° valor: 0.000
5° valor: 45.990
Foram lidos 12 valores.
```

# Exemplos práticos

- Arquivo de entrada contendo 3 valores *float* por linha; programa lê arquivo e gera como saída outro arquivo contendo, por linha, a média dos 3 valores;

- Salvar/salvar vetor de estruturas do tipo pessoa no arquivo texto.