



Disciplina:

# Programação Computacional

Prof. Fernando Rodrigues

e-mail: fernandorodrigues@sobral.ufc.br



## Aula 14: Programação em C

- ❖ Arquivos: Conceito de arquivos;
- ❖ Funções de manipulação de arquivos em C.

# Definição

---

- ▶ Um arquivo, de modo abstrato, nada mais é do que uma coleção de bytes armazenados em um dispositivo de armazenamento secundário, que é geralmente, um HD, um CD, um DVD etc.
- ▶ Essa coleção de bytes pode ser interpretada como: Caracteres, palavras ou frases de um documento de texto; Campos e registros de uma tabela de banco de dados; Pixels de uma imagem etc.
- ▶ O que define o significado de um arquivo em particular é a maneira como as estruturas de dados estão organizadas e as operações são usadas por um programa para processar (ler ou escrever) esse arquivo.

# Stream

---

- ▶ Embora cada dispositivo seja muito diferente, o sistema de arquivo com buffer transforma-os em um dispositivo lógico chamado de buffer.
- ▶ Todas as streams comportam-se de forma semelhante. Mesmo trabalhando com arquivos ou dispositivos diferentes.
- ▶ Existem dois tipos de streams
  - Binária
  - Texto

# Stream

---

## ► De texto:

- É uma sequência de caracteres;
- Um arquivo texto armazena caracteres que podem ser mostrados diretamente na tela ou modificados por um editor de texto simples como o Bloco de Notas;
- A “conversão” dos dados em texto faz com que os arquivos texto sejam maiores. Além disso, suas operações de escrita e leitura consomem mais tempo em comparação às dos arquivos binários.

## ► Binária:

- É uma sequência de bits / bytes com uma correspondência de um para um, com aqueles encontrados no dispositivo externo.
- Um arquivo binário armazena uma sequência de bits que está sujeita às convenções dos programas que o gerou.

# Arquivos Texto

---

- ▶ Os dados gravados em um arquivo texto são gravados exatamente como seriam impressos na tela. Por isso eles podem ser modificados por um editor de texto simples como o Bloco de Notas.
- ▶ No entanto, para que isso ocorra, os dados são gravados como caracteres de 8 bits utilizando a tabela ASCII. Ou seja, durante a gravação dos dados existe uma etapa de “conversão”.

## Arquivos Texto (Exemplo)

---

- ▶ Para entender essa conversão dos dados em arquivos texto, imagine um número inteiro com oito dígitos: 12345678.
- ▶ Esse número ocupa 32 bits (4 bytes) na memória. Porém, quando for gravado em um arquivo texto, cada dígito será convertido em seu caractere ASCII, ou seja, 8 bits por dígito.
- ▶ Como resultado final, esse número ocupará 64 bits no arquivo texto, o dobro do seu tamanho na memória como binário.

# Arquivos Binários

---

- ▶ Os dados gravados em um arquivo binário são gravados exatamente como estão organizados na memória do computador.
- ▶ Isso significa que não existe uma etapa de “conversão” dos dados.
- ▶ Portanto, suas operações de escrita e leitura são mais rápidas do que as realizadas em arquivos texto.

# Arquivos Binários (Exemplo)

---

- ▶ Voltemos ao nosso número inteiro com 8 dígitos: 12345678.
- ▶ Esse número ocupa 32 bits (ou 4 bytes) na memória. Quando for gravado em um arquivo binário, o conteúdo da memória será copiado diretamente para o arquivo, sem conversão.
- ▶ Como resultado final, esse número ocupará os mesmos 32 bits (4 bytes) no arquivo binário.



# Stream e arquivo

---

- ▶ O sistema de E/S de C provê um nível de abstração entre o programador e o dispositivo acessado.
- ▶ Essa abstração é chamada de stream, e o dispositivo real é chamado de arquivo.

# Arquivos

---

- ▶ Em C, um “arquivo” pode ser qualquer coisa, desde um arquivo em disco a uma impressora ou terminal.
- ▶ Você deve associar uma stream com um arquivo específico, realizando uma abertura de arquivo.
- ▶ Uma vez aberto, informações podem ser trocadas entre o arquivo e seu programa.

# Fundamentos do sistema de arquivos

---

- ▶ O sistema de arquivos é formado por um conjunto de diversas funções inter-relacionadas.
- ▶ Essas funções estão contidas na biblioteca “stdio.h”.

# Fundamentos do sistema de arquivos

---

- ▶ Já vimos como podemos exibir e ler dados para / do usuário através do teclado e da tela; (`printf`, `scanf` etc)
- ▶ Agora veremos também como ler e gravar dados em arquivos.
- ▶ Assim como as funções de entrada/saída padrão (teclado e tela), as funções de entrada/saída em arquivos estão declaradas no cabeçalho *stdio.h*.
  - ▶ "STanDard Input-Output".

# Fundamentos do sistema de arquivos

---

- ▶ Na manipulação de um arquivo, há basicamente três etapas que precisam ser realizadas:
  1. Abrir o arquivo;
  2. Ler e/ou gravar os dados desejados;
  3. Fechar o arquivo.
- ▶ Em C, todas as operações realizadas com arquivos envolvem seu *identificador de fluxo*, que é uma variável do tipo FILE \*.
- ▶ Todas as funções de manipulação de arquivos trabalham com o conceito de “ponteiro de arquivo”.

## Ponteiro de Arquivo: FILE \*

---

- No exemplo abaixo, “fp” é um ponteiro de arquivo, ou seja, é o ponteiro que nos permitirá manipular arquivos na linguagem C.

```
FILE *fp;           // não se esqueça do asterisco!
```

- Um ponteiro de arquivo nada mais é do que um ponteiro para uma área na memória chamada de “buffer”. Nela se encontram vários dados sobre o arquivo aberto, como o seu nome e a posição atual.

# Abrindo e fechando um arquivo

---

```
*fopen (char *nome_do_arquivo, char *modo_de_acesso);
```

- ▶ O nome do arquivo deve ser uma string com o caminho completo:
  - ▶ por exemplo, /usr/share/appname/app.conf
- ▶ ou C:\Documentos\nomes.txt
- ▶ ou o caminho em relação ao diretório atual (nomes.txt, ../app.conf) do arquivo que se deseja abrir ou criar.

# Abrindo e fechando um arquivo

```
*fopen (char *nome_do_arquivo, char *modo_de_acesso);
```

- ▶ O modo de acesso é uma string que contém uma sequência de caracteres que dizem se o arquivo será aberto para gravação ou leitura.
- ▶ Depois de aberto o arquivo, você só poderá executar os tipos de ação previstos pelo modo de acesso: não poderá ler de um arquivo que foi aberto somente para

Modo	Significado
r	Abre o arquivo somente para leitura. O arquivo deve existir. (O <i>r</i> vem do inglês <i>read</i> , ler)
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 não existir. (O <i>w</i> vem do inglês <i>write</i> , escrever)
w+	Abre o arquivo para escrita e leitura, apagando o conteúdo pré-existente.
a	Abre o arquivo para escrita no final do arquivo. Não apaga o conteúdo pré-existente. (O <i>a</i> vem do inglês <i>append</i> , adicionar, apender)
a+	Abre o arquivo para escrita no final do arquivo e leitura.



# Abrindo e fechando um arquivo

---

## ► Modo de acesso:

Modo	Arquivo	Função
"r"	Texto	Leitura. Arquivo deve existir.
"w"	Texto	Escrita. Cria arquivo, se não houver. Apaga o anterior, se ele existir.
"a"	Texto	Escrita. Os dados serão adicionados no fim do arquivo ("append").
"rb"	Binário	Leitura. Arquivo deve existir.
"wb"	Binário	Escrita. Cria arquivo se não houver. Apaga o anterior se ele existir.
"ab"	Binário	Escrita. Os dados serão adicionados no fim do arquivo ("append").
"r+"	Texto	Leitura/escrita. O arquivo deve existir e pode ser modificado.
"w+"	Texto	Leitura/escrita. Cria arquivo se não houver. Apaga o anterior se ele existir.
"a+"	Texto	Leitura/escrita. Os dados serão adicionados no fim do arquivo ("append").
"r+b"	Binário	Leitura/escrita. O arquivo deve existir e pode ser modificado.
"w+b"	Binário	Leitura/escrita. Cria arquivo se não houver. Apaga o anterior se ele existir.
"a+b"	Binário	Leitura/escrita. Os dados serão adicionados no fim do arquivo ("append").

- O arquivo deve sempre ser aberto em um modo que permita executar as operações desejadas.



# Abrindo e fechando um arquivo

---

- ▶ Abrindo / criando um arquivo binário para escrita e depois fechando o mesmo.

## Exemplo: abrir um arquivo binário para escrita

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04      FILE *fp;
05      fp = fopen("exemplo.bin","wb");
06      if(fp == NULL)
07          printf("Erro na abertura do arquivo.\n");
08
09      fclose(fp);
10      system("pause");
11      return 0;
12  }
```

# Abrindo e fechando um arquivo

---

```
*fopen (char *nome_do_arquivo, char *modo_de_acesso);
```

- ▶ O valor de retorno da função `fopen()` é muito importante! Ele é o identificador do fluxo que você abriu e é só com ele que você conseguirá ler e escrever no arquivo aberto.
- ▶ Se houver um erro na abertura/criação do arquivo, a função retornará o valor **NULL**. O erro geralmente acontece por duas razões:
  - ▶ O arquivo não existe, caso tenha sido requisitado para leitura.
  - ▶ O usuário atual não tem permissão para abrir o arquivo com o modo de acesso pedido. Por exemplo, o arquivo é somente-leitura, ou está bloqueado para gravação por outro programa, ou pertence a outro usuário e não tem permissão para ser lido por outros.

# Abrindo e fechando um arquivo

---

- ▶ Ao terminar de usar um arquivo, você deve fechá-lo. Isso é feito pela função `fclose( )`:

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

- ▶ O único argumento é o identificador do fluxo (retornado por `fopen`).
- ▶ O valor de retorno indica o sucesso da operação com o valor zero.

# Abrindo e fechando um arquivo

---

- ▶ Por que devemos fechar um arquivo aberto?
  - ▶ Ao fechar um arquivo, todo caractere que tenha permanecido no buffer é gravado.
  - ▶ O buffer é uma área intermediária entre o arquivo no disco e o programa em execução. Trata-se de uma região de memória que armazena temporariamente os caracteres a serem gravados em disco.
  - ▶ Apenas quando o buffer está cheio é que seu conteúdo é escrito (automaticamente) no disco.
- ▶ A função `exit()` fecha todos os arquivos que um programa tiver aberto.

# Abrindo e fechando um arquivo

---

- ▶ Por que utiliza-se buffer para escrever em arquivos?
  - ▶ O uso de um buffer é uma questão de eficiência.
  - ▶ Para ler e escrever arquivos no disco rígido é preciso posicionar a cabeça de gravação em um ponto específico do disco rígido. E isso consome tempo.
  - ▶ Se tivéssemos que fazer isso para cada caractere lido ou escrito, as operações de leitura e escrita de um arquivo seriam extremamente lentas.
  - ▶ Assim, a gravação só é realizada quando há um volume razoável de informações a serem gravadas ou quando o arquivo for fechado.

# Abrindo e fechando um arquivo

- ▶ Abrindo / criando um arquivo texto para escrita.

```
#include <stdio.h>

int main()
{
    FILE *fp;
    fp = fopen ("README", "w");
    if (fp == NULL) {
        printf ("Houve um erro ao abrir o arquivo.\n");
        return 1;
    }
    printf ("Arquivo README criado com sucesso.\n");
    fclose (fp);
    return 0;
}
```

# Arquivos pré-definidos

---

- ▶ Na biblioteca padrão do C, existem alguns fluxos pré-definidos que não precisam (nem devem) ser abertos nem fechados:
  - ▶ **stdin**: dispositivo de entrada padrão (geralmente o teclado)
  - ▶ **stdout**: dispositivo de saída padrão (geralmente o vídeo)
  - ▶ **stderr**: dispositivo de saída de erro padrão (geralmente o vídeo)
  - ▶ **stdaux**: dispositivo de saída auxiliar (em muitos sistemas, associado à porta serial)
  - ▶ **stdprn**: dispositivo de impressão padrão (em muitos sistemas, associado à porta paralela)





Disciplina:

# Programação Computacional

Prof. Fernando Rodrigues

e-mail: fernandorodrigues@sobral.ufc.br



## Aula 14.1: Programação em C

❖ Funções de manipulação de arquivos.

# Arquivos: Principais funções

---

<b>Nome</b>	<b>Função</b>
<code>fopen()</code>	Abre um arquivo
<code>fclose()</code>	Fecha um arquivo
<code>fputc()</code>	Escreve um caractere em um arquivo
<code>fgetc()</code>	Lê um caractere de um arquivo
<code>fseek()</code>	Posiciona o arquivo em um byte específico
<code>fprintf()</code>	É para um arquivo o que <code>printf()</code> é para o console
<code>fscanf()</code>	É para um arquivo o que <code>scanf()</code> é para o console
<code>feof()</code>	Devolve <b>verdadeiro</b> se o fim de arquivo for atingido
<code>ferror()</code>	Devolve <b>verdadeiro</b> se ocorreu um erro
<code>rewind()</code>	Recoloca o indicador de posição de arquivo no início do arquivo
<code>remove()</code>	Apaga um arquivo
<code>fflush()</code>	Descarrega um arquivo

---

# Escrevendo em arquivos

- Para escrever em arquivos, há quatro funções, das quais três são análogas às usadas para saída padrão:

Saída padrão	Arquivos	Explicação
putchar	fputc	Imprime apenas um caractere.
puts	fputs	Imprime uma string diretamente, sem nenhuma formatação.
printf	fprintf	Imprime uma string formatada.
N/A	fwrite	Grava dados binários para um arquivo.

```
void fputc (int caractere, FILE *fluxo);
```

```
void fputs (char *string, FILE *fluxo);
```

```
void fprintf (FILE *fluxo, char *formatação, ...);
```

```
int fwrite (void *dados, int tamanho_do_elemento, int num_elementos, FILE *fluxo);
```

# Escrevendo em arquivos - fputc()

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  #include <string.h>
04  int main(){
05      FILE *arq;
06      char string[100];
07      int i;
08      arq = fopen("arquivo.txt","w");
09      if(arq == NULL){
10          printf("Erro na abertura do arquivo");
11          system("pause");
12          exit(1);
13      }
14      printf("Entre com a string a ser gravada no arquivo:");
15      gets(string);
16      for(i = 0; i < strlen(string); i++)
17          fputc(string[i], arq);
18
19      fflush(arq);
20      fclose(arq);
21      system("pause");
22      return 0;
23 }
```



# Escrevendo em arquivos - fputs()

## Exemplo: escrevendo uma string em um arquivo com fputs()

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04      char str[20] = "Hello World!";
05      int result;
06      FILE *arq;
07      arq = fopen("ArqGrav.txt","w");
08      if(arq == NULL) {
09          printf("Problemas na CRIACAO do arquivo\n");
10          system("pause");
11          exit(1);
12      }
13      result = fputs(str,arq);
14      if(result == EOF)
15          printf("Erro na Gravacao\n");
16
17      fclose(arq);
18      system("pause");
19      return 0;
20  }
```

# Escrevendo em arquivos - fprintf()

## Exemplo: usando a função fprintf()

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04      FILE *arq;
05      char nome[20] = "Ricardo";
06      int I = 30;
07      float a = 1.74;
08      int result;
09      arq = fopen("ArqGrav.txt","w");
10      if(arq == NULL) {
11          printf("Problemas na ABERTURA do arquivo\n");
12          system("pause");
13          exit(1);
14      }
15      result = fprintf(arq,"Nome: %s\nIdade: %d\nAltura: %f\n",nome,i,a);
16      if(result < 0)
17          printf("Erro na escrita\n");
18      fclose(arq);
19      system("pause");
20      return 0;
21  }
```



# Escrevendo em arquivos - fwrite()

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04      FILE *arq;
05      arq = fopen("ArqGrav.bin","wb");
06      if(arq == NULL){
07          printf("Problemas na CRIACAO do arquivo\n");
08          system("pause");
09          exit(1);
10      }
11      int total_gravado, v[5] = {1,2,3,4,5};
12      //grava todo o array no arquivo (5 posições)
13      total_gravado = fwrite(v,sizeof(int),5,arq);
14      if(total_gravado != 5){
15          printf("Erro na escrita do arquivo!");
16          system("pause");
17          exit(1);
18      }
19      fclose(arq);
20      system("pause");
21      return 0;
22  }
```

# Lendo de arquivos

---

- ▶ Novamente, há quatro funções, das quais três se assemelham às usadas para a saída padrão:

Saída padrão	Arquivos	Explicação
getchar	fgetc	Recebe apenas um caractere.
gets	fgets	Lê uma string (geralmente uma linha inteira).
scanf	fscanf	Recebe uma string formatada.
N/A	fread	Lê dados binários de um arquivo.

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

```
void fgets (char *string, int tamanho, FILE *fluxo);
```

```
void fscanf (FILE *fluxo, char *formatação, ...);
```

```
int fread (void *dados, int tamanho_do_elemento, int num_elementos, FILE *fluxo);
```



# Lendo de arquivos - fgetc()



Cada chamada da função **fgetc()** lê um único caractere do arquivo especificado.

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04      FILE *arq;
05      char c;
06      arq = fopen("arquivo.txt","r");
07      if(arq == NULL){
08          printf("Erro na abertura do arquivo");
09          system("pause");
10          exit(1);
11      }
12      int i;
13      for(i = 0; i < 5; i++){
14          c = fgetc(arq);
15          printf("%c",c);
16      }
17      fclose(arq);
18      system("pause");
19      return 0;
20  }
```

# Lendo de arquivos – fgetc() e EOF



Quando atinge o final de um arquivo, a função **fgetc()** devolve a constante **EOF (End Of File)**, que está definida na biblioteca **stdio.h**. Em muitos computadores, o valor de **EOF** é definido como **-1**.

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04      FILE *arq;
05      char c;
06      arq = fopen("arquivo.txt","r");
07      if(arq == NULL){
08          printf("Erro na abertura do arquivo");
09          system("pause");
10          exit(1);
11      }
12      while((c = fgetc(arq)) != EOF)
13          printf("%c",c);
14      fclose(arq);
15      system("pause");
16      return 0;
17  }
```



# Lendo de arquivos - fgets()

## Exemplo: lendo uma string de um arquivo com fgets()

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04      char str[20];
05      char *result;
06      FILE *arq;
07      arq = fopen("ArqGrav.txt","r");
08      if(arq == NULL) {
09          printf("Problemas na ABERTURA do arquivo\n");
10          system("pause");
11          exit(1);
12      }
13      result = fgets(str,13,arq);
14      if(result == NULL)
15          printf("Erro na leitura\n");
16      else
17          printf("%s",str);
18
19      fclose(arq);
20      system("pause");
21      return 0;
22  }
```



# Lendo de arquivos - fgets()

---

- ▶ A função fgets() lê uma string do arquivo até que um caractere de nova linha (\n) seja lido ou “tamanho-1” caracteres tenham sido lidos.
- ▶ A string resultante de uma operação de leitura usando a função fgets() sempre terminará com a constante “\0” (por isso somente “tamanho-1” caracteres, no máximo, serão lidos).
- ▶ No caso de um caractere de nova linha (\n ou ENTER) ser lido antes de “tamanho-1” caracteres, ele fará parte da string.

# Lendo de arquivos - fscanf()

## Exemplo: usando a função fscanf()

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04      FILE *arq;
05      char texto[20], nome[20];
06      int i;
07      float a;
08      int result;
09      arq = fopen("ArqGrav.txt","r");
10      if(arq == NULL) {
11          printf("Problemas na ABERTURA do arquivo\n");
12          system("pause");
13          exit(1);
14      }
15      fscanf(arq,"%s%s",texto,nome);
16      printf("%s %s\n",texto,nome);
17      fscanf(arq,"%s %d",texto,&i);
18      printf("%s %d\n",texto,i);
19      fscanf(arq,"%s%f",texto,&a);
20      printf("%s %f\n",texto,a);
21      fclose(arq);
22      system("pause");
23      return 0;
24  }
```



# Lendo de arquivos - fread()



O valor do retorno da função **fread()** será igual ao valor de **count**, a menos que ocorra algum erro na leitura dos dados.

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04      FILE *arq;
05      arq = fopen("ArqGrav.bin","rb");
06      if(arq == NULL){
07          printf("Problemas na ABERTURA do arquivo\n");
08          system("pause");
09          exit(1);
10      }
11      int i,total_lido, v[5];
12      //lê 5 posições inteiras do arquivos
13      total_lido = fread(v,sizeof(int),5,arq);
14      if(total_lido != 5){
15          printf("Erro na leitura do arquivo!");
16          system("pause");
17          exit(1);
18      }else{
19          for(i = 0; i < 5; i++)
20              printf("v[%d] = %d\n",i,v[i]);
21      }
22      fclose(arq);
23      system("pause");
24      return 0;
25  }
```



# Exercícios I

---

- 1) Escreva um programa que leia do usuário o nome de um arquivo texto. Em seguida, mostre na tela quantas linhas esse arquivo possui.
- 2) Escreva um programa que leia do usuário os nomes de dois arquivos texto. Crie um terceiro arquivo texto com o conteúdo dos dois primeiros juntos (o conteúdo do primeiro seguido do conteúdo do segundo).
- 3) Crie um programa para calcular e exibir o número de palavras contido em um arquivo texto. O usuário deverá informar o nome do arquivo.
- 4) Faça um programa que leia 100 números. Esse programa deverá, em seguida, armazenar esses números em um arquivo binário.
- 5) Elabore um programa que leia um arquivo binário contendo 100 números. Mostre na tela a soma desses números.
- 6) Crie um programa que leia um arquivo binário contendo uma quantidade qualquer de números. O primeiro número lido indica quantos valores existem no arquivo. Mostre na tela o maior e o menor valor lido.

