Arquivos em C

Arquivos

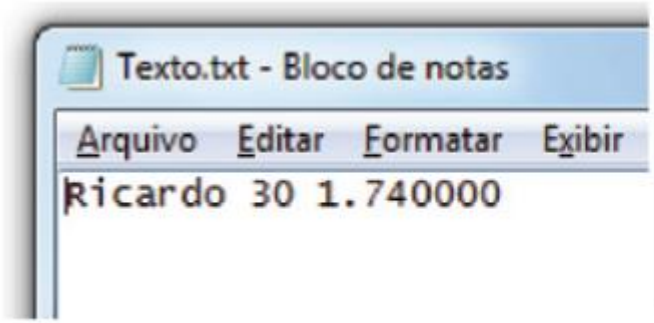
- Por que usar arquivos?
  - Permitem armazenar grande quantidade de informação;
  - Persistência dos dados (disco);
  - Acesso concorrente aos dados (mais de um programa pode usar os dados ao mesmo tempo).

Tipos de Arquivos

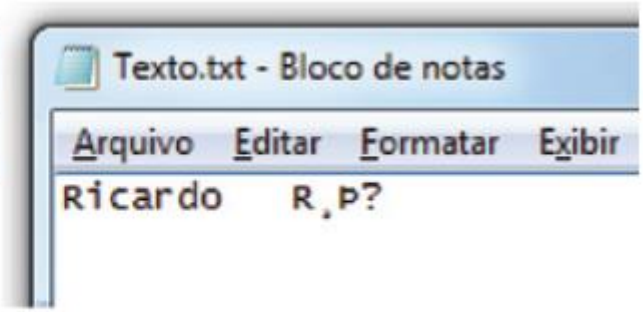
- Basicamente, a linguagem C trabalha com dois tipos de arquivos: de texto e binários.

Ex: Os dois trechos de arquivo abaixo possuem os mesmo dados:

```
char nome[20] = “Ricardo”;
int i = 30;
float a = 1.74;
```



Arquivo Texto



Arquivo Binário

Manipulando arquivos em C

- A linguagem C possui uma série de funções para manipulação de arquivos: **stdio.h**.
- 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".
- Declaração: FILE *p;
- p é o ponteiro para o arquivo que nos permitirá manipular arquivos no C.

Abrindo um arquivo

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

FILE *fopen(char *nome_arquivo,char *modo);

Abrindo um arquivo

- Um arquivo binário pode ser aberto para escrita utilizando o seguinte conjunto de comandos:

```
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    }
```

- A condição **fp==NULL** testa se o arquivo foi aberto com sucesso. No caso de erro **fopen()** retorna um ponteiro pra **NULL**.

Abrindo um arquivo

- No parâmetro nome_arquivo pode-se trabalhar com caminhos absolutos ou relativos.
- Caminho absoluto:
  - C:\\Projetos\\dados.txt
- Caminho relativo:
  - arq.txt
  - ..\\dados.txt

Modos de abertura

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").

Erro ao abrir um arquivo

- Caso o arquivo não tenha sido aberto com sucesso
  - Provavelmente o programa não poderá continuar a executar;
  - Nesse caso, utilizamos a função **exit(),** presente na biblioteca **stdlib.h**, para abortar o programa

void exit (int codigo_de_retorno);

Erro ao abrir um arquivo

```
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. Fim de programa.\n");
08        system("pause");
09        exit(1);
10    }
11    fclose(fp);
12    system("pause");
13    return 0;
14  }
```

Fechando um arquivo

- Sempre que terminamos de usar um arquivo que abrimos, devemos fechá-lo. Para isso usa-se a função **fclose()**

  int fclose (FILE *fp);

- O ponteiro **fp** passado à função **fclose()** determina o arquivo a ser fechado. A função retorna zero no caso de sucesso.

Fechando um arquivo

- Por que devemos fechar o arquivo?
  - Ao fechar um arquivo, todo caractere que tenha permanecido no "buffer" é gravado.
  - O "buffer“ é uma região de memória que armazena temporariamente os caracteres a serem gravados em disco imediatamente. Apenas quando o "buffer" está cheio é que seu conteúdo é escrito no disco.

Fechando um arquivo

- Por que utilizar um “buffer”?? Eficiência!
  - Para ler e escrever arquivos no disco temos que posicionar a cabeça de gravação em um ponto específico do disco.
  - Fazer isso a cada caractere lido/escrito, a leitura/escrita de um arquivo seria muita lenta.

Escrita/Leitura de Caracteres

- A função mais básica de entrada de dados é a função **fputc** (*put character*).

  int fputc (char ch, FILE *fp);

- Cada invocação dessa função grava um único caractere **ch** no arquivo especificado por **fp**.

Leitura/Escrita de Caracteres

```
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:");
      gets(string);
15    //Grava a string, caractere a caractere
16    for(i = 0; i < strlen(string); i++)
17      fputc(string[i], arq);
18    fclose(arq);
19    system("pause");
20    return 0;
21  }
```

Leitura/Escrita de Caracteres

- **fputc ('*', stdout)** exibe um * na tela do monitor (dispositivo de saída padrão).

Leitura/Escrita de Caracteres

- Cada chamada da função **fgetc** lê um único caractere do arquivo especificado.
  - Se **fp** aponta para um arquivo então **fgetc(fp)** lê o caractere atual no arquivo e se posiciona para ler o próximo caractere do arquivo.

```
int c;
c = fgetc(fp);
```

Leitura/Escrita de Caracteres

```
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  }
```

Leitura/Escrita de Caracteres

- **fgetc(stdin)** lê o próximo caractere digitado no teclado.

Leitura/Escrita de Caracteres

- O que acontece quando **fgetc** tenta ler o próximo caractere de um arquivo que já acabou?
  - Precisamos que a função retorne algo indicando o arquivo acabou.
- Porém, todos os 256 caracteres são "válidos"!

Leitura/Escrita de Caracteres

- Para evitar esse tipo de situação, **fgetc** não devolve um **char** mas um **int:**

  int fgetc (FILE *fp);

- Mais exatamente, **fgetc** devolve a constante **EOF** (*end of file*), que está definida na biblioteca **stdio.h**. Em muitos computadores o valor de EOF é −1.

```
if (c == EOF)
printf ("\nO arquivo terminou!");
```

Leitura/Escrita de Caracteres

```
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  }
```

Fim do arquivo

- Como visto, EOF ("End of file") indica o fim de um arquivo. No entanto, podemos também utilizar a função **feof** para verificar se um arquivo chegou ao fim.

  int feof (FILE *fp);

- Basicamente, a função retorna
  - Diferente de zero: se o arquivo NÃO chegou ao fim
  - Zero: se o arquivo chegou ao fim

Fim do arquivo

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

Arquivos pré-definidos

- Como visto anteriormente, os ponteiros **stdin** e **stdout** podem ser utilizados para acessar os dispositivo de entrada (geralmente o teclado) e saída (geralmente o vídeo) padrão.
- Na verdade, no início da execução de um programa, o sistema automaticamente abre alguns arquivos pré-definidos, entre eles **stdin** e **stdout**.

Arquivos pré-definidos

- **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)

Escrita/Leitura de Strings

- Funções para ler/escrever strings.

int fputs (char *str,FILE *fp);

char *fgets (char *str, int tamanho,FILE *fp);

Escrita/Leitura de Strings

```
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  }
```

Escrita/Leitura de Strings

- A função **fgets** recebe 3 parâmetros
  - **str**: aonde a lida será armazenada, **str**;
  - **tamanho**: o número máximo de caracteres a serem lidos;
  - **fp**: ponteiro que está associado ao arquivo de onde a string será lida.
- E retorna
  - NULL em caso de erro ou fim do arquivo;
  - O ponteiro para o primeiro caractere recuperado em **str**.

Escrita/Leitura de Strings

```
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  }
```

Escrita/Leitura de bloco de dados

- Temos duas funções
  - **fwrite()**
  - **fread()**

unsigned fwrite(void *buffer,int numero_de_bytes, int count, FILE *fp);

Escrita/Leitura de bloco de dados

- A função **fwrite** recebe 4 argumentos
  - **buffer:** ponteiro para a região de memória na qual estão os dados;
  - **numero_de_bytes:** tamanho de cada posição de memória a ser escrita;
  - **count:** total de unidades de memória que devem ser escritas;
  - **fp**: ponteiro para o arquivo.

Escrita/Leitura de bloco de dados

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  #include <string.h>
04  int main(){
05    FILE *arq;
06    arq = fopen("ArqGrav.txt","wb");
07    if(arq == NULL){
08      printf("Problemas na CRIACAO do arquivo\n");
09      system("pause");
10      exit(1);
11    }
12    char str[20] = "Hello World!";
13    float x = 5;
14    int v[5] = {1,2,3,4,5};
15    //grava a string toda no arquivo
16    fwrite(str,sizeof(char),strlen(str),arq);
17    //grava apenas os 5 primeiros caracteres da string
18    fwrite(str,sizeof(char),5,arq);
19    //grava o valor de x no arquivo
20    fwrite(&x,sizeof(float),1,arq);
21    //grava todo o array no arquivo (5 posições)
22    fwrite(v,sizeof(int),5,arq);
23    //grava apenas as 2 primeiras posições do array
24    fwrite(v,sizeof(int),2,arq);
25    fclose(arq);
26    system("pause");
27    return 0;
28  }
```

Escrita/Leitura de bloco de dados

- A função **fread** funciona como a sua companheira **fwrite**, porém lendo do arquivo.
- Como na função **fwrite, fread** retorna o número de itens escritos. Este valor será igual a **count** a menos que ocorra algum erro.
- Seu protótipo é:

```
unsigned fread (void *buffer, int
numero_de_bytes, int count, FILE *fp);
```

Escrita/Leitura de bloco de dados

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04    FILE *arq;
05    arq = fopen("ArqGrav.txt","rb");
06    if(arq == NULL){
07      printf("Problemas na ABERTURA do arquivo\n");
08      system("pause");
09      exit(1);
10    }
11    char str1[20],str2[20];
12    float x;
13    int i,v1[5],v2[2];
14    //lê a string toda do arquivo
15    fread(str1,sizeof(char),12,arq);
16    str1[12] = '\0';
17    printf("%s\n",str1);
```

Escrita/Leitura de bloco de dados

```
18    //lê apenas os 5 primeiros caracteres da string
19    fread(str2,sizeof(char),5,arq);
20    str2[5] = '\0';
21    printf("%s\n",str2);
22    //lê o valor de x do arquivo
23    fread(&x,sizeof(float),1,arq);
24    printf("%f\n",x);
25    //lê todo o array do arquivo (5 posições)
26    fread(v1,sizeof(int),5,arq);
27    for(i = 0; i < 5; i++)
28      printf("v1[%d] = %d\n",i,v1[i]);
29    fread(v2,sizeof(int),2,arq);
30    //lê apenas as 2 primeiras posições do array
31    for(i = 0; i < 2; i++)
32      printf("v2[%d] = %d\n",i,v2[i]);
33    fclose(arq);
34    system("pause");
35    return 0;
36  }
```

Escrita/Leitura por fluxo padrão

- Ex: **fprintf**

  printf (“Total = %d”,x);//escreve na tela

  fprintf (**fp**,“Total = %d”,x);//grava no arquivo **fp**

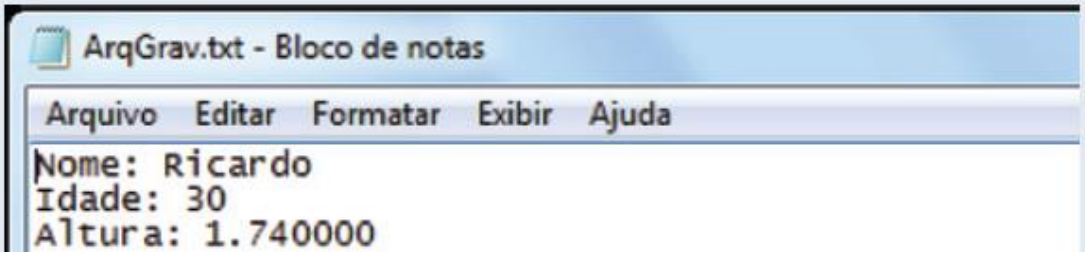
- Ex: **fscanf**

  scanf (“%d”,&x);//lê do teclado

  fscanf (**fp**,“%d”,&x);//lê do arquivo **fp**

Escrita por fluxo padrão

```
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  }
```



Leitura por fluxo padrão

```
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  }
```

Movendo-se pelo arquivo

- **Rewind:** Retorna para o seu início do arquivo

  void rewind (FILE *fp);

Movendo-se pelo arquivo

int fseek (FILE *fp,long numbytes,int origem);

- Esta função move a posição corrente de leitura ou escrita no arquivo em uma quantidade de bytes, a partir de um ponto especificado.

Movendo-se pelo arquivo

- A função **fseek** recebe 3 parâmetros
  - **fp**: o ponteiro para o arquivo;
  - **numbytes**: é o total de bytes a partir de **origem** a ser pulado;
  - **origem**: determina a partir de onde os **numbytes** de movimentação serão contados. Os valores possíveis são definidos por macros em **stdio.h**:

Movendo-se pelo arquivo

- Os valores possíveis para **origem** são definidos por macros em **stdio.h** e são:

Nome	Valor	Significado
SEEK_SET	0	Início do arquivo
SEEK_CUR	1	Ponto corrente no arquivo
SEEK_END	2	Fim do arquivo

- A função devolve 0 quando bem sucedida.

Movendo-se pelo arquivo

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  struct cadastro{ char nome[20], rua[20]; int idade;};
04  int main(){
05    FILE *f = fopen("arquivo.txt","wb");
06    if(f == NULL){
07       printf("Erro na abertura\n");
08       system("pause");
09       exit(1);
10    }
11    struct cadastro c,cad[4] = {"Ricardo","Rua 1",31,
12                                "Carlos","Rua 2",28,
13                                "Ana","Rua 3",45,
14                                "Bianca","Rua 4",32};
15    fwrite(cad,sizeof(struct cadastro),4,f);
16    fclose(f);
17    f = fopen("arquivo.txt","rb");
18    if(f == NULL){
19       printf("Erro na abertura\n");
20       system("pause");
21       exit(1);
22    }
23    fseek(f,2*sizeof(struct cadastro),SEEK _ SET);
24    fread(&c,sizeof(struct cadastro),1,f);
25    printf("%s\n%s\n%d\n",c.nome,c.rua,c.idade);
26    fclose(f);
27    system("pause");
28    return 0;
29  }
```

Apagando um arquivo

- Além de permitir manipular arquivos, a linguagem C também permite apagá-lo do disco. Isso pode ser feito utilizando a função **remove**:

  int remove (char *nome_do_arquivo);

- Diferente das funções vistas até aqui, esta função recebe o caminho e nome do arquivo a ser excluído, e não um ponteiro para FILE.
- Como retorno temos um valor inteiro, o qual será igual a 0 se o arquivo for excluído com sucesso.

Apagando um arquivo

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04    int status;
05    status = remove("ArqGrav.txt");
06    if(status != 0){
07      printf("Erro na remocao do arquivo.\n");
08      system("pause");
09      exit(1);
10    }else
11      printf("Arquivo removido com sucesso.\n");
12
13    system("pause");
14    return 0;
15  }
```