Programação para Computação
13ª Aula

Prof. Max Santana Rolemberg Farias
2007

Arquivos em C

O sistema de E/S da linguagem C utiliza o conceito de ***streams*** e ***arquivos***, como um nível de abstração entre o programador e o dispositivo utilizado.

Uma ***stream*** é um dispositivo lógico que representa um arquivo ou dispositivo (terminal, disco, fita, etc).

A ***Stream*** é totalmente independente do dispositivo, dessa forma a função que manipula uma ***stream*** pode escrever tanto em um arquivo em disco ou em algum outro dispositivo, como terminal.

Existem dois tipos de ***stream***:

- ***Stream de Texto:*** É uma sequencia de caracteres, traduzida de acordo com o sistema.
- ***Stream Binário:*** É uma sequencia de bytes com uma correspondência de um para um com os bytes encontrados no dispositivo.

Arquivos

Na linguagem C, um arquivo pode ser qualquer coisa, desde um arquivo em disco até um terminal ou uma impressora.

Para utilizar um arquivo você deve associá-lo a uma ***stream*** e, então, manipular a ***stream***. Você associa um arquivo a uma ***stream*** através de uma **operação de abertura**.

Um arquivo é desassociado de uma **stream** através de uma **operação de fechamento**.

Se um arquivo aberto para saída por fechamento, o conteúdo de sua ***stream*** será escrito no dispositivo externo, garantindo que nenhuma informação seja acidentalmente deixada no ***buffer*** de disco.

A ***stream*** associa o arquivo a uma estrutura do tipo **FILE**. Esta estrutura é definida na biblioteca ***stdio.h***.

Funções Utilizadas

As principais funções para manipulação de arquivos. Estas funções estão na biblioteca ***stdio.h***.

FUNÇÃO	FINALIDADE
fopen( )	Abrir um arquivo
fclose( )	Fechar um arquivo
putc( )	Escrever um caractere em um arquivo
fputc( )	Escrever um caractere em um arquivo
getc( )	Ler um caractere em um arquivo
fgetc( )	Ler um caractere em um arquivo
fseek( )	Posicionar o ponteiro de arquivos num byte especifico
fprintf( )	É para o arquivo o que o printf é para o console
fscanf( )	É para o arquivo o que o scanf é para o console
feof( )	Retorna verdadeiro se o fim do arquivo foi atingido
ferror( )	Retorna verdadeiro se ocorreu um erro
rewind( )	Posiciona o ponteiro de arquivo no início
remove( )	Apagar um arquivo

Ponteiro de Arquivos

O ponteiro de arquivo é o meio comum que une o sistema de E/S.

Basicamente um ponteiro de arquivo identifica um arquivo especifico e é usado pela ***stream*** para direcionar as operações das funções de E/S.

Um ponteiro de arquivo é uma variável ponteiro do tipo **FILE**.

Para ler ou escrever em arquivos seu programa precisa usar os ponteiros de arquivos. Para declarar uma variável como ponteiro de arquivo use a seguinte sintaxe:

```
FILE *arquivo;
```

Abrindo um Arquivo

Para abriri uma ***stream*** e associá-la a um arquivo devemos usar a função ***fopen()***.

```
fopen(<arquivo>, <modo>)
```

O ***<arquivo>*** representa o nome do arquivo. O nome do arquivo pode ser um ***path***.

O ***<modo>*** é uma string que representa como o arquivo será aberto.

A função ***fopen*** retorna um ponteiro de arquivo, caso ocorra algum problema ela retorna **NULL**.

NULL é uma constante definida em ***stdio.h*** e define um ponteiro nulo.

Modo de Abertura de Arquivo

MODO	DECRIÇÃO
r	Abre um arquivo texto para leitura
w	Abre um arquivo texto para escrita. Se o arquivo existir será sobrescrito
a	Abre um arquivo texto para anexação. Se o arquivo não existir será criado
rb	Abre um arquivo binário para leitura
wb	Abre um arquivo binário para escrita. Se o arquivo existir será sobrescrito
ab	Abre um arquivo binário para anexação. Se o arquivo não existir será criado
r+ w+ a+	Abre um arquivo texto para leitura/escrita. Se o arquivo não existir será criado
r+b w+b a+b rb+ wb+ ab+	Abre um arquivo binário para leitura/escrita. Se o arquivo não existir será criado

Abrindo um Arquivo

Para abriri um arquivo texto chamado “teste” para escrita devemos fazer o seguinte:

```
FILE *arquivo;

arquivo = fopen(”teste”, ”w”);
```

É recomendado sempre testar se o arquivo foi aberto sem problema. Então sempre que for abrir um arquivo, devemos usar o código abaixo:

```
FILE *arquivo;

if ((arquivo = fopen(”teste”, ”w”)) == NULL){
  printf(”Erro ao abrir o arquivo\n”);
}
```

Este tipo de teste detecta problemas do tipo disco cheio ou protegido contra gravação antes da gravação.

Fechando um Arquivo

Para fechar uma ***stream*** devemos usar a função **fclose()**.

A função ***fclose*** escreve os dados do buffer de disco no arquivo e o fecha em nível de sistema operacional. Uma falha ao fechar uma ***stream*** pode provocar problemas tipo perda de dados, arquivo destruídos e erros intermitentes em seu programa.

fclose também libera o bloco de controle de arquivo associado à ***stream*** deixando-o disponível para reutilização.

Como, normalmente, há um limite para número de arquivos abertos ao mesmo tempo, devemos fechar um arquivo antes de abrir um outro.

A sintaxe é:

fclose(<arquivo>);

Onde **<arquivo>** é o ponteiro de arquivo retornado pela função ***fopen*** quando abrimos o arquivo. Caso o fechamento ocorra sem erro a função ***fclose*** retorna zero.

Escrevendo Caracteres

Para escrever um caractere em um arquivo aberto podemos usar duas funções ***putc*** ou ***fputc***. Elas são idênticas e existem para preservar a compatibilidade com versões antigas da linguagem.

Sintaxe:

```
putc(<caractere>, <arquivo>);

fputc(<caractere>, <arquivo>);
```

Onde **<caractere>** é o caractere a ser escrito e **<arquivo>** é o ponteiro de arquivo.

Se ocorrer tudo bem, a função retorna o caractere escrito, caso contrário ela retorna **EOF**.

EOF é uma constante definida na biblioteca ***stdio.h*** é geralmente é definida como -1 e retorna este valor quando uma função de entrada tenta ler além do final do arquivo.

Lendo Caracteres

Para ler um caractere em um arquivo aberto podemos usar duas funções getc ou fgetc. Elas são idênticas e existem para preservar a compatibilidade com versões antigas da linguagem.

Sintaxe:

```
getc(<arquivo>);
```

```
fgetc(<arquivo>);
```

Onde **<arquivo>** é o ponteiro de arquivo.

Quando o final do arquivo é alcançado a função retorna **EOF**.

Lendo Caracteres

Para ler o conteúdo de uma arquivo você poderia usar um trecho de código abaixo:

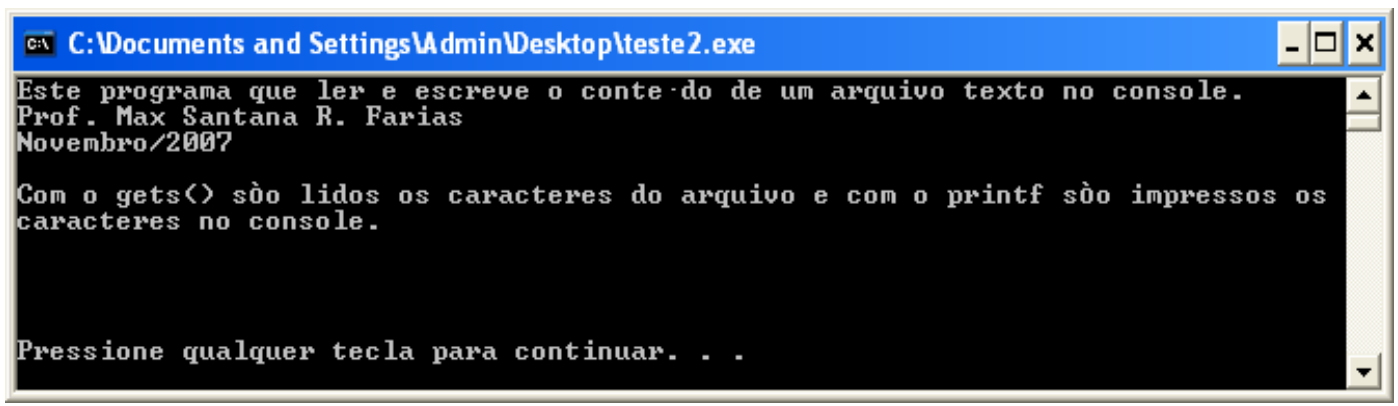
```
FILE *arquivo;

if ((arquivo = fopen(”teste”, ”r”)) == NULL){
  printf(”Erro ao abrir o arquivo\n”);
}
char caractere;
do{
  caractere = getc(arquivo);
} while(caractere != EOF);
```

Sendo **caractere** uma variável ***char*** e arquivo uma variável ponteiro para uma estrutura **FILE**.

É importante observar que ***getc*** também retorna **EOF** caso ocorra algum erro.

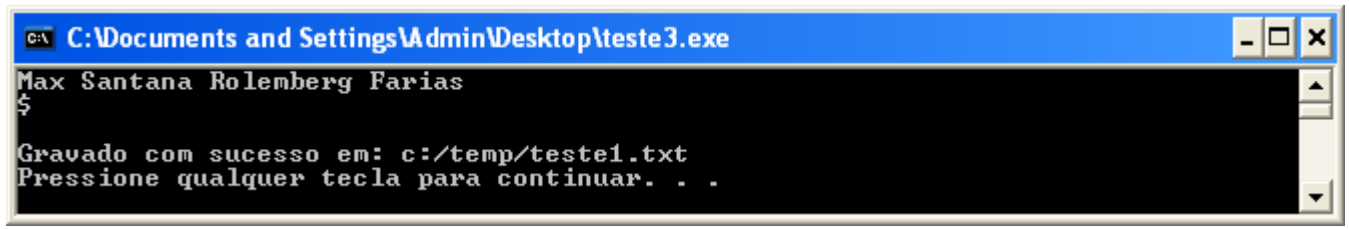
Exemplo: Lendo do Arquivo

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

int main(){
  char caractere;
  FILE *arquivo; // Ponteiro de arquivo
  // Testa e abre o arquivo para leirura
  if ((arquivo = fopen("c:/temp/teste.txt","r")) == NULL){
    printf("Erro ao tentar abrir o arquivo: c:/temp/teste.txt");
    system("PAUSE");
    return 0;
  }
  // ler cada caracter do arquivo e imprime no console
  do{
    caractere = getc(arquivo);
    printf("%c", caractere);
  }while(caractere != EOF);
  printf("\n\n");
  // Fecha o arquivo
  fclose(arquivo);
  system("PAUSE");
  return 0;
}
```



Exemplo: Escrevendo no Arquivo

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

int main(){
  char caractere;
  FILE *arquivo; // Ponteiro de arquivo
  // Testa e abre o arquivo para escrita
  if ((arquivo = fopen("c:/temp/teste1.txt","w")) == NULL){
    printf("Erro ao tentar abrir o arquivo: c:/temp/teste1.txt");
    system("PAUSE");
    return 0;
  }
  // ler cada caracter do arquivo e imprime no console
  do{
    caractere = getchar();
    putc(caractere, arquivo);
  }while(caractere != '$');
  // Fecha o arquivo
  fclose(arquivo);
  printf("\nGravado com sucesso em: c:/temp/teste1.txt\n");

  system("PAUSE");
  return 0;
}
```



Escrevendo e Lendo Strings

Para escrever e ler ***strings*** em um arquivo use as funções ***fputs*** e ***fgets***, da biblioteca ***stdio.h***.

fputs escreve uma string na ***stream*** especificada, sua sintaxe é:

```
fputs(<string>, <arquivo>);
```

Onde **<arquivo>** é um ponteiro de arquivo. Caso ocorra algum erro esta função retorna **EOF**.

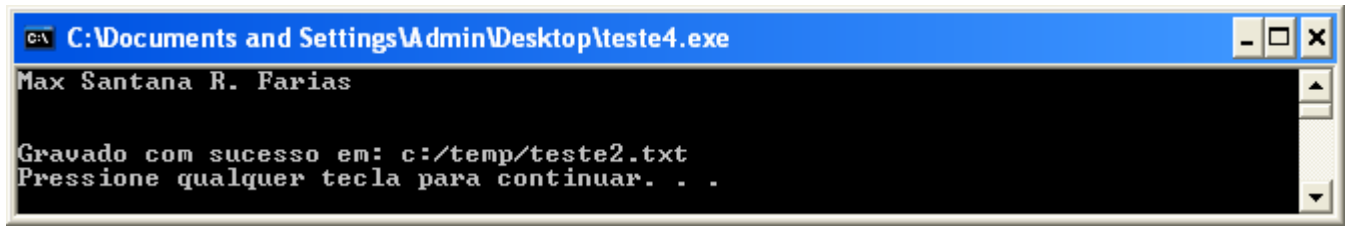
fgets lê uma string da ***stream*** especificada, sua sintaxe é:

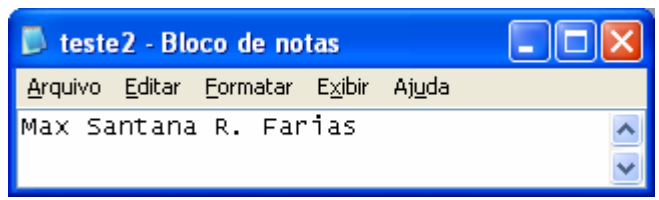
fgets(<string>, <tamanho>, <arquivo>);

A função ***fgets*** lê **<string>** até que um caractere de nova linha seja lido ou que <tamanho>-1 caracteres tenham sido lidos. Se uma nova linha é lida ela será parte da string. Caso ocorra tudo bem esta função retornará uma **string**, caso contrário retornará um ponteiro nulo.

Exemplo: Escrevendo Strings

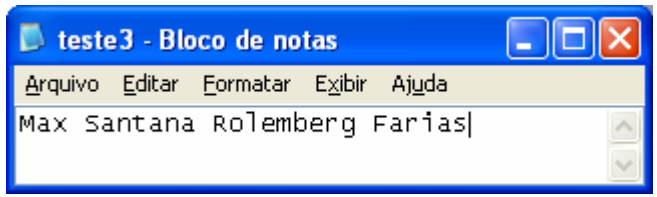
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(){
  char string[80];
  FILE *arquivo; // Ponteiro de arquivo
  // Testa e abre o arquivo para escrita
  if ((arquivo = fopen("c:/temp/teste2.txt","w")) == NULL){
    printf("Erro ao tentar abrir o arquivo: c:/temp/teste2.txt");
    system("PAUSE");
    return 0;
  }
  // ler uma string e armazena em um arquivo
  do{
    gets(string);
    strcat(string,"\n");
    fputs(string,arquivo);
  }while(*string != '\n');
  // Fecha o arquivo
  fclose(arquivo);
  printf("\nGravado com sucesso em: c:/temp/teste2.txt\n");
  system("PAUSE");
  return 0;
}
```

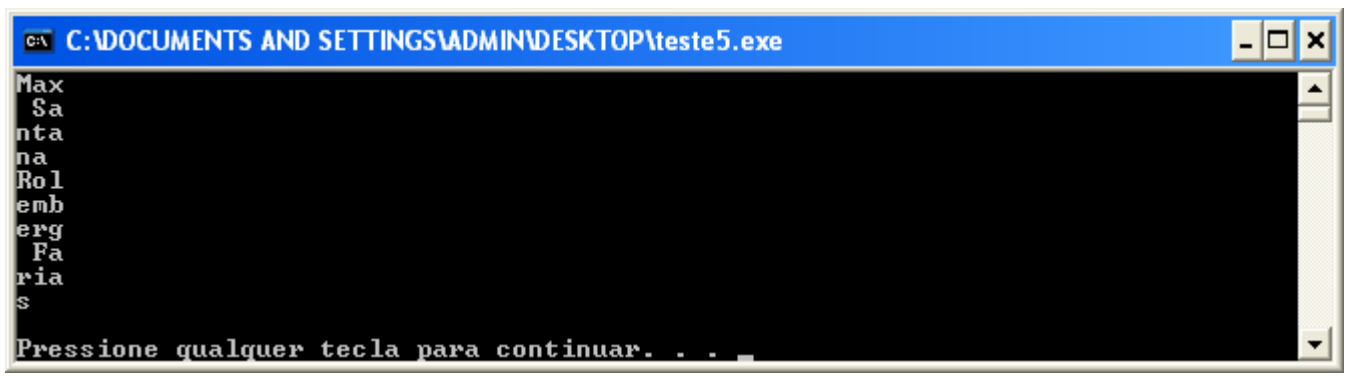




Exemplo: Lendo Strings

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

int main(){
  char string[80];
  FILE *arquivo; // Ponteiro de arquivo
  // Testa e abre o arquivo para escrita
  if ((arquivo = fopen("c:/temp/teste3.txt",“r")) == NULL){
    printf("Erro ao tentar abrir o arquivo: c:/temp/teste3.txt");
    system("PAUSE");
    return 0;
  }
  // ler strings de tamanho (4-1) até chegar no final do arquivo
  while (!feof(arquivo))  {
    // Lê uma string de tamanho (4-1) (inclusive com o '\n')
    fgets(string, 4, arquivo);
    printf("%s\n",string);
  }
  // Fecha o arquivo
  fclose(arquivo);
  system("PAUSE");
  return 0;
}
```





Entrada e Saída Formatada

Existem as funções ***fprintf*** e ***fscanf*** que são semelhantes a ***printf*** e ***scanf*** mas que direcionam os dados para arquivos.

Sintaxe:

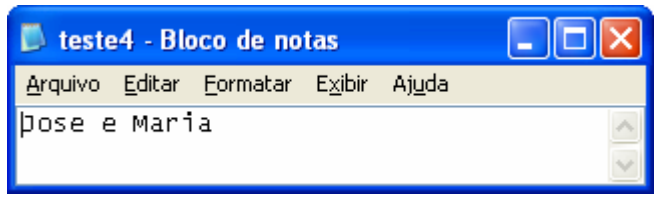
```
fprintf(<arquivo>, <especificador>, <variável>);

fscanf(<arquivo>, <especificador>, <variável>);
```

Onde **<arquivo>** é um ponteiro de arquivo para onde são direcionados os resultados das funções. O **<especificador>** é a string de formatação.

Embora sejam uma opção para ler e escrever dados em arquivo, esta funções devem ser evitadas pois trabalham com dados **ASCII**.

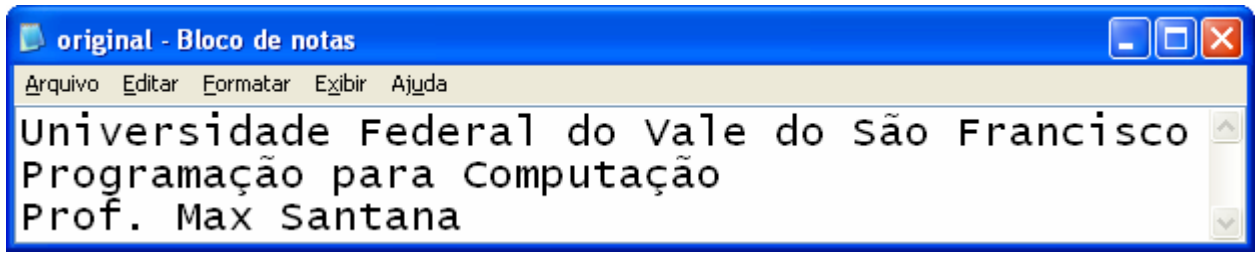
Exemplo: fprintf

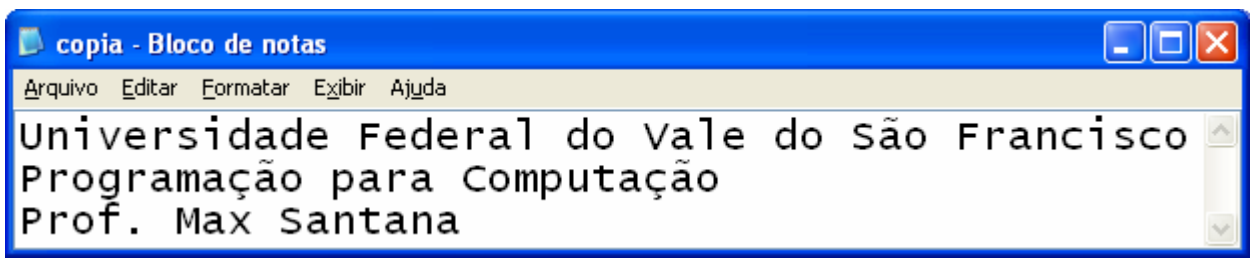
```
#include <stdio.h>
#include <stdlib.h>

int main(){
  char string[80];
  FILE *arquivo; // Ponteiro de arquivo
  // Testa e abre o arquivo para escrita
  if ((arquivo = fopen("c:/temp/teste4.txt","w")) == NULL){
    printf("Erro ao tentar abrir o arquivo: c:/temp/teste4.txt");
    system("PAUSE");
    return 0;
  }
  printf(“Exemplo do uso do fprintf\n”);
  printf(“Digite seu nome: ”);
  // Lê strings do teclado
  gets(string);
  // Escreve string no arquivo
  fprintf(arquivo,”%s”,string);
  // Fecha o arquivo
  fclose(arquivo);
  system("PAUSE");
  return 0;
}
```



Exemplo: Programa que Copia Arquivo

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
  char caractere;
  FILE *arqOriginal, *arqCopia;
  // Testa e abre o arquivo para escrita
  if ((arqOriginal = fopen("c:/temp/original.txt",“r")) == NULL){
    printf("Erro ao tentar abrir o arquivo: c:/temp/original.txt");
    system("PAUSE");
    return 0;
  }
  if ((arqCopia = fopen("c:/temp/copia.txt",“w")) == NULL){
    printf("Erro ao tentar abrir o arquivo: c:/temp/copia.txt");
    system("PAUSE");
    return 0;
  }
  while (!feof(arqOriginal)){
    caractere = getc(arqOriginal);
    if (!feof(arqOriginal))
      putc(caractere, arqCopia);
  }
  fclose(arqOriginal);
  fclose(arqCopia);
  system("PAUSE");
  return 0;
}
```





Função *fwrite( )*

A função ***fwrite()*** pode escrever qualquer tipo de dado e não apenas caracteres ou cadeias de caracteres. A sintaxe de ***fwrite*** é:

fwrite(<variável>, <tamanho>, <quantidade>, <arquivo>);

Onde **<variavel>** é o endereço da variável que receberá os dados a ser escrito no arquivo, **<tamanho>** é o número de bytes a ser escrito (para calcular o tamanho deve usar o operador ***sizeof***), **<quantidade>** indica quantos itens que serão escritos e **<arquivo>** é o ponteiro de arquivo.

Quando a função ***fwrite( )*** for bem-sucedida, vai retornar um número de gravações realizadas (o parâmetro **<quantidade>** descrito anteriormente). Caso contrário, quando ocorrer algum erro, o valor retornado será menor que **<quantidade>**.

```
FILE *arquivo;
char nome[50] = “Max Santana”;
...
fwrite(&nome,sizeof(nome),1,arquivo);
```

Função *fread( )*

A função ***fread( )*** pode ler qualquer tipo de dado e não apenas caracteres ou cadeias de caracteres. A sintaxe de ***fwrite*** é:

fread(<variável>, <tamanho>, <quantidade>, <arquivo>);

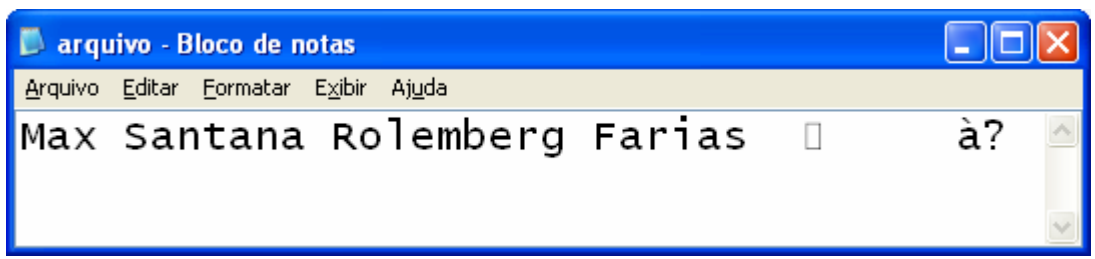
Onde **<variavel>** é o endereço da variável que receberá os dados lidos no arquivo, **<tamanho>** é o número de bytes a ser lidos (para calcular o tamanho deve usar o operador ***sizeof***), **<quantidade>** indica quantos itens que serão lidos e **<arquivo>** é o ponteiro de arquivo.

Quando a função ***fread( )*** for bem-sucedida, vai retornar um número de leituras realizadas (o parâmetro **<quantidade>** descrito anteriormente). Caso contrário, quando ocorrer algum erro ou quando o final do arquivo for encontrado, o valor retornado será menor que **<quantidade>**.

```
FILE *arquivo;
char nome[50];
...
fread(&nome,sizeof(nome),1,arquivo);
```

Exemplo: *fwrite( )*

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

int main(){
  char nome[30] = “Max Santana Rolemberg Farias”;
  int idade = 29;
  float altura = 1.75;
  FILE *arquivo;
  // Testa e abre o arquivo para escrita
  if ((arquivo = fopen("c:/temp/arquivo.txt",“w")) == NULL){
    printf("Erro ao tentar abrir o arquivo: c:/temp/arquivo.txt");
    system("PAUSE");
    return 0;
  }

  fwrite(&nome,sizeof(nome),1,arquivo);
  fwrite(&idade,sizeof(idade),1,arquivo);
  fwrite(&altura,sizeof(altura),1,arquivo);

  fclose(arquivo);
  system("PAUSE");
  return 0;
}
```

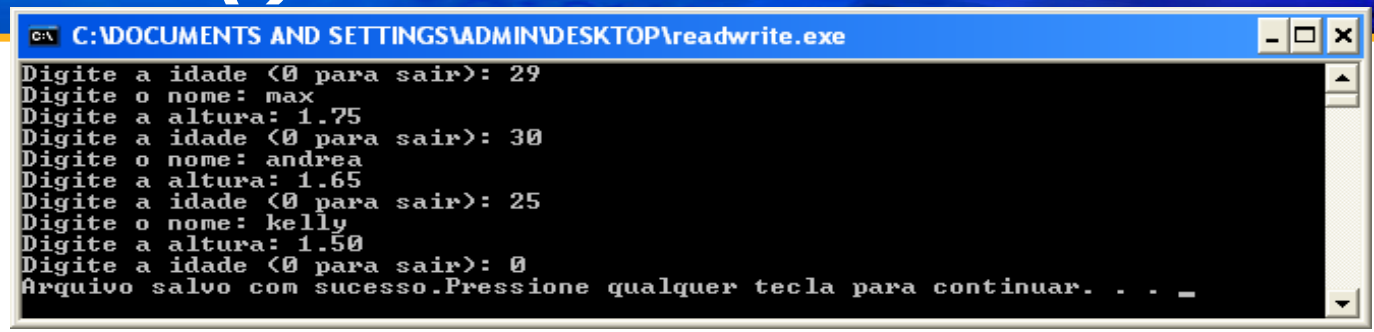


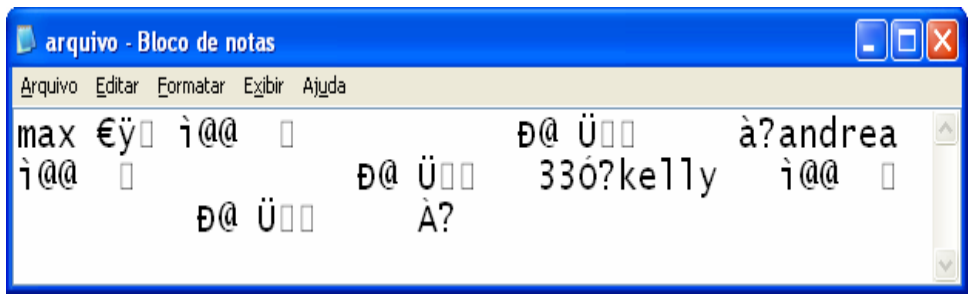
Exemplo: *fread( )*

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

int main(){
  char nome[30];
  int idade;
  float altura;
  FILE *arquivo;
  // Testa e abre o arquivo para escrita
  if ((arquivo = fopen("c:/temp/arquivo.txt",“r")) == NULL){
    printf("Erro ao tentar abrir o arquivo: c:/temp/arquivo.txt");
    system("PAUSE");
    return 0;
  }
  fread(&nome,sizeof(nome),1,arquivo);
  fread(&idade,sizeof(idade),1,arquivo);
  fread(&altura,sizeof(altura),1,arquivo);
  printf(“Nome: %s\n”, nome);
  printf(“Idade: %d\n”, idade);
  printf(“Altura: %.2f\n”, altura);
  fclose(arquivo);
  system("PAUSE");
  return 0;
}
```

Exemplo: *fwrite( )*

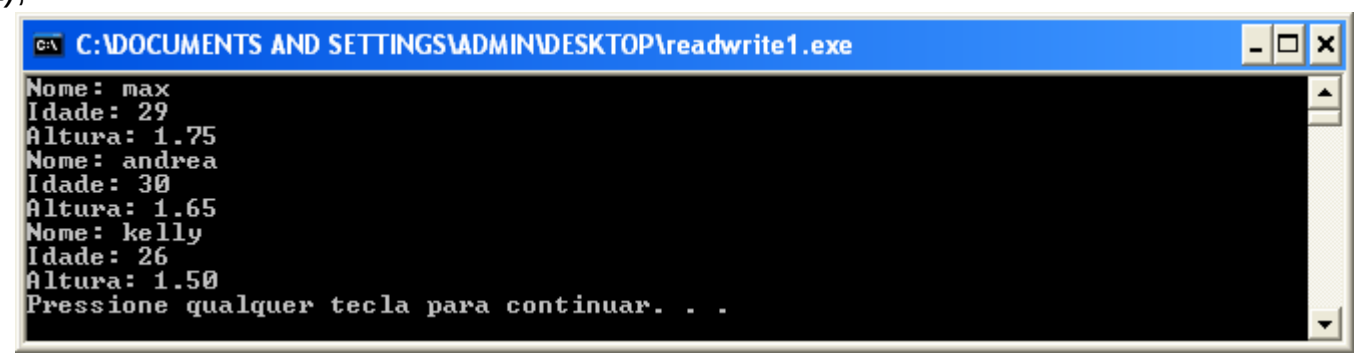
```
#include <stdio.h>
#include <stdlib.h>
int main(){
  char nome[30];
  int idade; float altura;
  FILE *arquivo;
  if ((arquivo = fopen("c:/temp/arquivo.dat","wb")) == NULL){
    printf("Erro ao tentar abrir o arquivo: c:/temp/arquivo.dat");
    system("PAUSE");
    return 0;
  }
  printf("Digite a idade (0 para sair): ");
  scanf("%d",&idade);
  while(idade!=0){
    printf("Digite o nome: ");  scanf("%s",&nome);
    printf("Digite a altura: ");  scanf("%f",&altura);
    fwrite(&nome,sizeof(nome),1,arquivo);
    fwrite(&idade,sizeof(idade),1,arquivo);
    fwrite(&altura,sizeof(altura),1,arquivo);
    printf("Digite a idade (0 para sair): "); scanf("%d",&idade);
  }
  fclose(arquivo);
  printf("Arquivo salvo com sucesso.");
  system("PAUSE");
  return 0;
}
```





Exemplo: *fwrite( )*

```
#include <stdio.h>
#include <stdlib.h>
int main(){
  char nome[30];
  int idade; float altura;
  FILE *arquivo;
  if ((arquivo = fopen("c:/temp/arquivo.dat",“rb")) == NULL){
    printf("Erro ao tentar abrir o arquivo: c:/temp/arquivo.dat");
    system("PAUSE");
    return 0;
  }
  while(!feof(arquivo)){
  fread(&nome,sizeof(nome),1,arquivo);
  fread(&idade,sizeof(idade),1,arquivo);
  fread(&altura,sizeof(altura),1,arquivo);
    if (!feof(arquivo)){
      printf("Nome: %s\n", nome);
      printf("Idade: %d\n", idade);
      printf("Altura: %.2f\n", altura);
    }
  }
  fclose(arquivo);
  system("PAUSE");
  return 0;
}
```



Função *fseek( )*

A função ***fseek()*** posiciona o cursor (ponteiro) em um endereço específico, tornando possível leituras e escritas aleatórias. A sintaxe de ***fseek*** é:

fseek(<arquivo>, <bytes>, <posição>);

Onde **<arquivo>** representa o ponteiro de arquivo, **<bytes>** representa a quantidade de bytes que será percorrida a partir de **<posição>**, **<posição>** representa o ponto a partir do qual a busca será executada, ao parâmetro **<posição>** poderá receber três valores (**SEEK_SET**, **SEEK_CUR**, **SEEK_END**).

SEEK_SET: permite a movimentação de **<bytes>** a partir da posição inicial do arquivo;
SEEK_CUR: permite a movimentação de **<bytes>** a partir da posição atual do arquivo;
SEEK_END: permite a movimentação de **<bytes>** a partir da posição final do arquivo.

fseek(arquivo,**sizeof**(nome),SEEK_SET);

Função *frewind( )*

A função ***frewind()*** posiciona o cursor (ponteiro) no início do arquivo. A sintaxe de ***frewind*** é:

frewind(<arquivo>);

Onde **<arquivo>** representa o ponteiro de arquivo.

frewind(arquivo);

Função *fremove( )*

A função ***fremove()*** apaga um arquivo. A sintaxe de ***fremove*** é:

fremove(<nome_arquivo>);

Onde **<nome_arquivo>** representa o nome do arquivo que será removido (podendo ser incluido o ***path***).

fremove(“c:/temp/teste.txt”);

Função *fflush( )*

A função ***fflush()*** escreve o conteúdo armazenado no buffer dentro de um determinado arquivo. A sintaxe de ***fflush*** é:

fflush(<arquivo>);

Onde **<arquivo>** representa o ponteiro de arquivo.

Caso a função ***fflush()*** não indique um arquivo especificamente, as gravações associadas a todos os arquivos abertos no momento serão efetuadas.

fflush(arquivo);