Arquivos em C

Notas de Aula

Prof. Francisco Rapchan
www.geocities.com/chicorapchan
rapchan@terra.com.br

Há várias definições para **arquivos** em computação. Uma das mais usadas é a de que um arquivo é um **conjunto de registros** (estruturas) onde cada registro é formado por um conjunto de **campos** e cada campo representa um dado (ou informação) nele contido. Uma ficha que contenha os campos para preenchimento é um exemplo de registro. O conjunto dessas fichas é um arquivo. As informações nome, endereço, telefone e idade são exemplos de campos de um registro de cliente. O conjunto de todos os clientes seria o arquivo de cliente.

A principal vantagem na utilização de arquivos está no fato dos dados armazenados poderem ser utilizados a qualquer momento (em varias execuções diferentes do programa) não se perdendo quando o computador é desligado. Outra vantagem encontrada na utilização de arquivos é o fato dele poder armazenar um número maior de registros do que em uma tabela em memória, estando apenas limitado ao tamanho do meio físico utilizado para a sua gravação (o disco, a fita magnética, etc.).

É comum referir-se a **arquivos** como "**meios persistentes**" pois os dados ficam gravados em meio físico e não se perdem quando o sistema (computador) é desligado ou reiniciado.

Existem 2 tipos de arquivos:

- *Arquivos de formato texto*. Nesses arquivos são gravados apenas os caracteres como se fosse no vídeo e de onde são lidos caracteres como se estivessem sendo digitados no teclado. Por exemplo, os programas fonte que são digitados, são exemplos de arquivos de texto. Um arquivo texto pode ser visualizado com editor de texto.
- *Arquivos de formato Binário.* Nesses arquivos os dados são gravados como na memória. De forma geral não é possível ver claramente o conteúdo de um arquivo binário usando um editor de texto comum.

As principais declarações, operações e funções usadas na manipulação de arquivos são as seguintes:

FILE *arq;
fopen ("dados.txt", "w");
getc(arq)
fprintf (arq,"Bom dia!");
fscanf (arq,"%i",&numero)
fclose (arq);
EOF
arq = = NULL
sizeof (t_aluno)
fwrite (&dado, sizeof (t_aluno),1,arq);
fread (&dado, sizeof (t_aluno),1,arq);
ftell(arq)
fseek (arq, posicao, SEEK_SET);
fflush (arq);

Arquivos Texto

Exemplo 1. Faça um programa para criar um arquivo chamado “dados.txt” que armazene o texto “Bom dia!”.

```
#include <stdio.h>
main()
{
   FILE *arq;

   arq = fopen ("dados.txt", "w");

   fprintf (arq,"Bom dia!");

   fclose (arq);
}
```

Esse programa mostra a estrutura básica para manipulação de arquivos.

Para manipular arquivos precisamos:

1. Criar um ponteiro para FILE.
2. Abrir o arquivo.
3. Ler ou gravar dados no arquivo.
4. Fechar o arquivo.

Use um editor de texto para ver o conteúdo de dados.txt.

Abaixo descrevemos os detalhes das operações usadas no programa anterior.

`FILE *arq;`

Cria um ponteiro chamado arq para uma estrutura FILE (o asterisco indica que arq é um ponteiro). A estrutura FILE é definida em stdio.h e contém dados que permitem ao programa manipular arquivos.

`arq = fopen ("dados.txt", "w");`

A função fopen permite abrir um arquivo. Quando abrimos o arquivo passamos como parâmetros o nome do arquivo (dados.txt) e a forma como queremos que ele seja aberto. Neste caso colocamos “w” indicando que queremos abrir dados.txt para gravação (write).

Abaixo temos uma tabela com os formatos para abertura de arquivos texto.

"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.
"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.

`fprintf (arq,"Bom dia!");`

A função fprintf funciona como o printf comum, a diferença é que o primeiro parâmetro indica o arquivo onde os dados serão gravados.

`fclose (arq);`

Ao final precisamos fechar o arquivo com fclose. Dessa forma esse arquivo fica livre para ser usado por outros arquivos.

Exemplo 2. Faça um programa para criar um arquivo com o nome que o usuário quiser. Depois leia um texto do teclado e grave nesse arquivo.

```
#include <stdio.h>
main()
{
   FILE *saida;
   char nome [41], texto [81];

   // Pergunta o nome do arquivo
   printf ("Nome do arquivo: ");
   scanf  (" %40[^\n]",nome);

   // Abre o arquivo para gravação
   saida = fopen (nome, "w");

   // Lê o texto que será gravado
   printf ("Texto: ");
   scanf  (" %81[^\n]",texto);

   // Grava o texto no arquivo
   fprintf (saida,texto);

   // Fecha o arquivo
   fclose (saida);
}
```

Neste programa chamamos de **saida** o ponteiro para FILE.

Criamos duas strings:

- **nome [41]** que irá conter o nome do arquivo.
- **texto [81]** que irá conter o texto que será gravado no arquivo.

Após perguntarmos o nome do arquivo abrimos saida para gravação (parâmetro “w” no fopen).

Depois, perguntamos o texto que o usuário quer colocar no arquivo. Então, usamos fprintf para gravar esse texto no arquivo.

Ao final fechamos o arquivo saida com fclose.

Exemplo 3. Faça um programa em que o usuário digite o nome de um arquivo texto e seja mostrado na tela o conteúdo desse arquivo.

```
#include <stdio.h>
main()
{
   FILE *arquivo;
   char c;
   char nome [41];

   // Pergunta o nome do arquivo
   printf ("Nome do arquivo: ");
   scanf  (" %40[^\n]",nome);

   // Abre o arquivo para leitura
   arquivo = fopen (nome, "r");

   // Lê caracteres até o fim do arquivo
   c = getc(arquivo);
   while(c != EOF){
      printf("%c", c);
      c = getc(arquivo);
   }

   // Fecha o arquivo
   fclose (arquivo);

}
```

Observe que neste programa estamos abrindo o arquivo para leitura (veja a opção “r” de read em fopen).

Após abrirmos o arquivo, passamos a ler cada caractere do arquivo usando a função getc. Esta função retorna um a um os caracteres do arquivo. Quando chega ao final ela retorna EOF.

Em C o segmento de código abaixo:

```
c = getc(arquivo);
while(c != EOF){
   printf("%c", c);
   c = getc(arquivo);
}
```

Pode ser escrito da seguinte forma:

```
while( (c = getc(arquivo) ) != EOF)
   printf("%c", c);
```

Exemplo 4. Faça um programa que leia os números reais de um arquivo texto chamado "numeros.txt" e mostre na tela a soma desses números. Observação: o arquivo "numeros.txt" contendo os números que serão somados, pode ser construído usando um editor de texto qualquer (kwrite, notepade, kedit, etc.).

```
#include <stdio.h>
main()
{
   float numero;
   float soma;

   FILE *arq;

   // Abre o arquivo para leitura (read)
   arq = fopen ("numeros.txt", "r");

   // Lê os números até o fim do arquivo
   soma = 0;
   while (fscanf (arq,"%f",&numero)!= EOF)
      soma = soma + numero;

   // Mostra a soma dos números
   printf ("Soma: %10.2f\n",soma);

   // Fecha o arquivo
   fclose (arq);
}
```

Este é um programa muito simples. Ele exemplifica como pode ser útil poder ler arquivos textos.

Neste caso o programa lê um arquivo texto que contém uma série de números reais. Os números podem estar em uma mesma linha (separados por um espaço) ou cada um em uma linha.

Exemplo 5. Faça um programa que leia o número e a média cada aluno cadastrado em um arquivo chamado "notas.txt". Coloque os alunos com média maior ou igual a 6.0 em um arquivo chamado "aprovados.txt" e os demais em um arquivo chamado "recuperacao.txt". Observação: o arquivo "alunos.txt", contendo o número e as média de cada aluno, pode ser construído usando um editor de texto qualquer (kwrite, notepade, kedit, etc.).

```
#include <stdio.h>
main ()
{
   FILE *notas, *aprov, *recup;
   int numero;
   float media;

   // Abre os arquivos
   notas  = fopen ("notas.txt", "r");
   aprov  = fopen ("aprovados.txt", "w");
   recup  = fopen ("recuperacao.txt", "w");

   // Lê os dados e grava no arquivo
   while (fscanf (notas,"%i %f",&numero,&media)!= EOF){
      if (media >= 6)
         fprintf (aprov,"%i %4.1f\n",numero,media);
      else
         fprintf (recup,"%i %4.1f\n",numero,media);
   }

   // Fecha os arquivos
   fclose (notas);
   fclose (aprov);
   fclose (recup);
}
```

A novidade neste programa é o uso de mais de um arquivo ao mesmo tempo.

Neste caso temos 3 arquivos sendo manipulados: notas, aprov e reprov.

O programa lê o registro de cada aluno (número e média). Se o aluno estiver aprovado, seus dados serão gravados em aprov. Caso contrário será gravado em recuper.

Exemplo 6. Faça um programa que grave os seguintes dados em um arquivo chamado “notas.txt”:

Número	Nome	Média
1	Maria Luiza	7.0
2	Ana Maria	8.0

```
#include <stdio.h>
main()
{  // Cria a estrutura t_aluno
   typedef struct {
      int   numero;
      char  nome [41];
      float media;
   } t_lista ;

   // Cria a[2] e já preenche os dados.
   t_lista a[2] = {{1,"Maria Luiza",7},{2,"Ana Maria",8}};

   FILE *arq;

   // Abre o arquivo para gravação (write)
   arq = fopen ("notas.txt", "w");

   // Grava os dados no arquivo de maneira formatada
   fprintf (arq,"%10i %-40s %8.2f\n",a[0].numero, a[0].nome, a[0].media);
   fprintf (arq,"%10i %-40s %8.2f\n",a[1].numero, a[1].nome, a[1].media);

   // Fecha o arquivo
   fclose (arq);
}
```

Observe que estamos criando o vetor a[2] e já estamos preenchendo seus dados. Essa atribuição é possível apenas quando a variável é criada (no corpo do programa não é possível fazer esse tipo de atribuição).

Depois de executar este programa, podemos usar um editor de texto qualquer para ver o conteúdo do arquivo notas.txt.

Exemplo 7. Faça um programa leia e mostre na tela os dados do programa anterior.

```
#include <stdio.h>

main()
{  int   numero;
   char  nome [41];
   float media;
   int   c;
   FILE *arq;

   // Abre o arquivo para leitura
   arq = fopen ("notas.txt", "r");

   // Lê os dados do arquivo e mostra na tela
   for (c = 0; c < 2; c++){
      fscanf (arq,"%i %40[^\n] %f",&numero, nome, &media);
      printf ("%10i %-40s %8.2f\n",numero, nome, media);
   }

   // Fecha o arquivo
   fclose (arq);
}
```

Exemplo 8. Faça um programa permita gravar o número, o nome e a média de N alunos em um arquivo.

```
#include <stdio.h>
main()
{ int   qtde;       // quantidade de registros
  int   numero;     // número do aluno
  char  nome [41];  // nome do aluno
  float media;      // média do aluno
  FILE *arq;        // ponteiro para o arquivo
  int   c;          // contador

  // Abre o arquivo para gravação
  arq = fopen ("teste.txt", "w");

  // Usuário informa o número de registros
  printf ("Numero de alunos: "); scanf (" %i",&qtde);

  // Grava no arquivo o número de registros
  fprintf (arq, "%i\n",qtde);

  // Usuário digita os dados
  for (c = 0; c < qtde; c++){
    printf  ("\nRegistro %i\n",c+1);
    printf  ("Numero: "); scanf (" %i",&numero);
    printf  ("Nome  : "); scanf (" %40[^\n]",nome);
    printf  ("Media : "); scanf (" %f",&media);

    // Grava os dados no arquivo
    fprintf(arq,"%10i %-40s %8.2f\n",numero,nome,media);
  }

  // Fecha o arquivo
  fclose (arq);
}
```

Um dos aspectos mais interessantes desse programa é o fato dele armazenar dentro do arquivo o número de registros que irá conter (chamamos aqui de qtde).

Dessa forma um outro programa pode ler esse número para saber quantos registros terá que ler até chegar ao fim do arquivo.

Exemplo 9. Faça um programa que permita ler e mostrar os dados do programa anterior.

```
#include <stdlib.h>
#include <stdio.h>
main()
{ int   qtde;       // quantidade de registros
  int   numero;     // número do aluno
  char  nome [41];  // nome do aluno
  float media;      // média do aluno
  FILE  *arq;       // ponteiro para o arquivo
  int   c;          // contador

  // Abre o arquivo para leitura
  arq = fopen ("teste.txt", "r");

  // Verifica se o arquivo foi aberto com sucesso
  if (arq == NULL){
     printf ("Erro ao abrir o arquivo\n");
     exit (1);
  }
  // Lê o número de registros do arquivo
  fscanf (arq, "%i",&qtde);

  // Lê os dados do arquivo e mostra na tela
  for (c = 0; c < qtde; c++){
    fscanf (arq,"%i %40[^\n] %f",&numero,nome,&media);
    printf ("%10i %-40s %8.2f\n",numero, nome, media);
  }
  fclose (arq);     // Fecha o arquivo
}
```

Uma novidade nesse programa é o teste feito para saber se o arquivo foi aberto com sucesso. Caso a função fopen não consiga abrir o arquivo ela retorna NULL.

Assim, se arq for igual a NULL, é mostrada uma mensagem na tela e o programa é encerrado através da função exit( ).

A função exit( ) está disponível em <stdlib.h>. Ela faz com que o programa termine e retorne, para o sistema operacional, o código de retorno.

A convenção mais usada é que um programa retorne zero no caso de um término normal e retorne um número não nulo no caso de ter ocorrido um problema.

Arquivos Binários

Exemplo 10. Faça um programa que grave em um arquivo binário chamado "teste.dat" o nome e a média de N alunos.

```
#include <stdio.h>
#include <stdlib.h>
main(){
  // Define o tipo t_aluno
  typedef struct {
     char  nome [41];
     float media;
  } t_aluno;

  t_aluno a;   // Cria variável a do tipo t_aluno
  int   qtde;  // quantidade de registros
  FILE  *arq;  // ponteiro para o arquivo
  int   c;     // contador

  // Abre o arquivo binário para gravação
  arq = fopen ("teste.dat", "wb");
  if (arq == NULL){
     printf("Problemas na criação do arquivo\n");
     exit (1);
  }

  // Usuário digita o número de registros
  printf ("Numero de alunos: "); scanf (" %i",&qtde);

  // Usupario digita os dados
  for (c = 0; c < qtde; c++){
    printf  ("\nRegistro %i\n",c+1);
    printf  ("Nome  : "); scanf (" %40[^\n]",a.nome);
    printf  ("Media : "); scanf (" %f",&a.media);

    // Grava os dados no arquivo
    fwrite (&a, sizeof (t_aluno),1,arq);
  }

  fclose (arq);  // Fecha o arquivo
}
```

Neste programa estamos abrindo o arquivo "teste.dat" para gravação de um **arquivo binário** ("wb" – *write binary*).

Para gravarmos os dados, usamos a função **fwrite**:

fwrite (&a, sizeof (t_aluno),1,arq);

que recebe os seguintes **parâmetros**:

- um ponteiro para um buffer, ou seja, uma região da memória onde os dados lidos serão armazenados;
- o número de bytes de um bloco que será escrito no arquivo;
- a quantidade destes blocos;
- um ponteiro para o arquivo.

Para identificarmos o número de bytes de um bloco usamos o operador **sizeof**.

Exemplo 11. Faça um programa que leia e mostre na tela os dados do arquivo "teste.dat" do exemplo anterior.

```
#include <stdio.h>
#include <stdlib.h>
main(){
   typedef struct {
      char  nome [41];
      float media;
   } t_aluno;

   t_aluno a;
   FILE *arq;

   // Abre o arquivo binário para leitura
   arq = fopen ("teste.dat", "rb");

   // Lê os dados no arquivo
   while (fread (&a, sizeof (t_aluno),1,arq)!= 0)
      printf ("%-40s %4.1f\n",a.nome, a.media);

   fclose (arq);   // Fecha o arquivo
}
```

Neste programa, abrimos o arquivo binário "teste.dat" para leitura ("rb" – *read binary*);

A função **fread** serve para ler dados em um arquivo binário. Seus argumentos são equivalentes aos da função fwrite.

Observe que o programa continua lendo dados enquanto o resultado de fread não for zero (indicando que chegou ao fim do arquivo).

Exemplo 12. Escreva um programa que multiplique por dois as notas gravadas no arquivo teste.dat dos exemplos anteriores.

```
#include <stdio.h>
#include <stdlib.h>
main(){
   typedef struct {
      char  nome [41];
      float media;
   } t_aluno;

   t_aluno a;
   FILE *arq;
   int posicao;

   // Abre o arquivo binário para leitura e gravação
   if ((arq = fopen ("teste.dat", "r+b")) == NULL){
      printf("Problemas na criação do arquivo\n");
       exit (1);
   }

   // Le os dados e os atualiza
   while (fread (&a, sizeof (t_aluno),1,arq)!= 0){
      a.media = a.media * 2;
      posicao = ftell(arq)-sizeof (t_aluno);
      fseek (arq, posicao, SEEK_SET);
      fwrite (&a, sizeof (t_aluno),1,arq);
      fflush (arq);
   }

   // Fecha o arquivo
   fclose (arq);
}
```

Neste programa o arquivo teste.dat é aberto para leitura e gravação de dados binários (**"r+b"**). O algoritmo usado é o seguinte:

```
Abre o arquivo.
Enquanto houver dados, leia um registro.
       Altere o valor da média
       Posicione o ponteiro do arquivo no registro anterior
       Grave o novo dado
Fim do enquanto.
```

A função **ftell** indica a posição corrente do ponteiro do arquivo. Essa posição deve ser deslocada um registro para trás para poder sobrescrever o registro atualizado. A função **fseek** é que faz esse re-posicionamento.

A função **fflush** garante que o registro será imediatamente gravado. Caso essa função não seja usada, corre o risco dos dados ficarem temporariamente armazenados em um buffer de disco, descontrolando o algoritmo.