Manipulação de arquivos em C

As operações de entrada e saída do C, incluindo as relacionadas a arquivos, encontram-se na biblioteca *stdio.h*. Essa biblioteca também define várias macros, dentre elas NULL e EOF, que definem um ponteiro nulo e o fim de arquivo, respectivamente. Além disso, é nela que está definido o tipo FILE.

Na *Tabela 1* são listadas as principais funções relacionadas a manipulação de arquivos existentes na biblioteca *stdio.h*.

Tabela 1: Principais funções de manipulação de arquivos da biblioteca stdio.h

Função	O que faz?
fopen()	Abre um arquivo.
fclose()	Fecha o arquivo garantindo a transferência do *buffer*.
fflush()	Descarrega o *buffer*.
fscanf()	Leitura de entrada formatada (semelhante ao scanf()).
fprintf()	Escrita de saída formatada (semelhante ao printf()).
fgets()	Obtém uma string do arquivo.
fgetc()	Obtém um caracter do arquivo.
fputs()	Insere uma string no arquivo.
fputc()	Insere um caracter no arquivo.
fread()	Lê um bloco de dados do arquivo.
fwrite()	Escreve um bloco de dados no arquivo.
fseek()	Reposiciona o ponteiro.
rewind()	Reposiciona o ponteiro para o início do arquivo.
ftell()	Retorna a posição do ponteiro.

Associação do arquivo

O primeiro passo para trabalhar com um arquivo é fazer a associação do arquivo físico com um arquivo lógico. Para isso utilizamos o tipo FILE, definido na biblioteca *stdio.h*. A abertura/associação do arquivo é feita pela função fopen(const char* arquivo, const char* modo), em que arquivo é o diretório/nome do arquivo a ser aberto e modo é o modo que a associação é feita. Os tipos de associação estão descritos na *Tabela 2*.

Tabela 2: Modos de abertura de arquivos

”r”	Abre o arquivo somente para leitura, a partir do início. O arquivo deve existir.
”w”	Cria um arquivo vazio para escrita. Se já havia o arquivo, ele é perdido.
”a”	Adiciona no final do arquivo. Se o arquivo não existir, a função o cria.

”r+”	Abre o arquivo para leitura e escrita, a partir do início. O arquivo deve existir.
”w+”	Cria um arquivo vazio para leitura e escrita. Se já havia o arquivo, ele é perdido.
”a+”	Abre para adição ou leitura no final do arquivo. Se o arquivo não existir, a função o cria.

No Windows, o caracter “b” pode ser adicionado ao modo (ex: “ab”, “w+b”, etc) para especificar que o arquivo deve ser aberto no modo binário. Em sistemas POSIX (inclusive Linux), esse caracter é ignorado.Também é possível utilizar o caracter “t”, para abertura de no modo texto.

O código a seguir mostra um exemplo da associação.

```
#include <stdio.h>

main(int argc,char *argv[]) {

   FILE *fp;

   if ((fp=fopen (argv[1],"w"))==NULL)
      printf ("Erro na abertura do arquivo.");
   else
      printf("Arquivo aberto com sucesso.");

   fclose(fp);
}
```

Quando um programa encerra corretamente, com exit(0) por exemplo, os arquivos lógicos são liberados da memória. Porém, se o programa fechar com erro, o arquivo não é liberado. Para evitar que isso aconteça, é conveniente fechar o arquivo quando não for mais necessário o seu uso. Para isso basta usar a função fclose(arquivo).

Manipulação do conteúdo

Para ler um caracter do arquivo, basta utilizar a função fgetc(FILE * arquivo). De forma semelhante, para escrever um caracter no arquivo, basta utilizar a função fputc(FILE * arquivo). O código a seguir é um exemplo de leitura, que conta o número de letras ‘a’ no arquivo file.txt.

```
#include <stdio.h>

main(int argc,char *argv[]) {

   FILE *fp;
   char c;
   int n = 0;

   if ((fp=fopen (“file.txt”,"r")) != NULL) {
      while( (c=fgetc(fp)) !=EOF) {
         if (c=='a' || c=='A') n++;
      }
      fclose(fp);
      printf(“Existem %d letras a no arquivo.\n”, n);

}
```

Também é possível fazer leitura e escrita formatadas, com as funções fscanf(FILE * arquivo, const char* formato, ...) e fprintf(FILE * arquivo, const char* formato, ...). O funcionamento dessas funções são semelhantes às conhecidas scanf(formato, ...) e printf(formato, ...), mas direcionada para arquivos. O próximo código é um programa que lê dez nomes do teclado e escreve no arquivo nomes.txt.

```
#include <stdio.h>

main(int argc,char *argv[]) {

   FILE *fp;
   char nome[50];

   if ((fp=fopen (“nomes.txt”,"w")) != NULL) {
      for(int i=0; i<10; i++) {
         printf(“Escreva um nome: ”);
         gets(nome);
         fprintf(fp, “Nome %d: %s\n”, i+1, nome);
      }
   }
   fclose(fp);
}
```

Também é possível fazer leitura e escrita do arquivo em blocos. Para isso, devemos utilizar as funções fread(void * buffer, size_t tamanho, size_t cont, FILE * arquivo) e fwrite(void * buffer, size_t tamanho, size_t cont, FILE * arquivo), em que buffer contem o que se deseja escrever, tamanho indica o tamanho em bytes de cada elemento do buffer e cont indica quantos elementos são lidos/escritos. O código a seguir é um exemplo do uso do fwrite.

```
#include <stdio.h>

main(int argc,char *argv[]) {

   FILE *fp;
   char buffer[] = {‘x’, ‘y’, ‘z’};

   if ((fp=fopen (“nomes.txt”,"wb")) != NULL) {
      fwrite(buffer, 1, sizeof(buffer), fp);
   fclose(fp);

}
```

O funcionamento de outras funções, assim como exemplos de código, podem ser encontrados em http://www.cplusplus.com/reference/clibrary/cstdio/, local de referência para a montagem desse tutorial.