Métodos e Técnicas de Programação
.:: Entrada e Saída de arquivos ::.

Prof. Igor Peretta

2018-2

Contents

1 Manipulação de arquivos (streams)

Similares às funções de entrada/saída padrão (teclado e tela), as funções de entrada/saída em arquivos estão declaradas em stdio.h. As funções para manipulação de arquivos são semelhantes às usadas para entrada/saída padrão. A manipulação de arquivos também se dá por meio de fluxos (streams).

São três etapas principais:

1. Abrir o arquivo;
2. Ler e/ou gravar dados;
3. Fechar o arquivo.

Em C, as operações realizadas com arquivos envolvem uma variável do tipo FILE * (ponteiro, também conhecido como identificador de fluxo).

1.1 Abrir ou fechar arquivos

Para abrir um stream (arquivo), protótipo:

```
FILE * fopen (char * filename, char * access_mode);
```

Table 1: Modos de acesso para abertura de arquivo

Modo	Significado
r	Somente para leitura
r+	Leitura e escrita
w	Escrita, ponteiro no início do arquivo
w+	Escrita e leitura, apaga o conteúdo ou cria novo
a	Escrita, ponteiro no final do arquivo
a+	Escrita, ponteiro no final do arquivo, e leitura
b	Modo binário, em alternativa ao modo texto

Nota: Se `fopen()` retornar NULL significa que algum erro impediu o programa de abrir o arquivo.

1.2 Fechando arquivos

Para fechar um stream (arquivo), protótipo:

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

1.3 Streams pré-definidos

Esses streams não devem ser abertos ou fechados.

Table 2: Streams padrão

Stream	Significado
stdin	dispositivo de entrada padrão (teclado)
stdout	dispositivo de saída padrão (vídeo)
stderr	dispositivo de saída de erro padrão (vídeo)
stdaux	dispositivo de saída auxiliar (porta serial)
stdprn	dispositivo de impressão padrão (porta paralela)

```
#include <stdio.h>
#include <stdlib.h>
int main() {
  FILE * arquivo;
  arquivo = fopen("acorda.igor2", "w");
  if(arquivo) {
    printf("Foi!\n");
    fclose(arquivo);
  } else {
    printf("Arquivo não pode ser aberto!\n");
  }
  return 0;
}
```

```
Foi!
```

2 Escrita e Leitura em arquivos

2.1 Escrita em streams (arquivos)

2.1.1 Escreve o próximo caractere (unsigned)

Protótipo:

```
void fputc (int byte, FILE * stream);
```

2.1.2 Escreve uma string

Protótipo:

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

2.1.3 Escreve uma string formatada

Protótipo:

```
   void fprintf (FILE * stream, char * format, ...);
```

```
fprintf(stdout, "Olá mundo de novo!\n");
```

```
Olá mundo de novo!
```

```
#include <stdio.h>
int main() {
  FILE * arquivo; int x = 123;
  arquivo = fopen("ave.tonta", "w");
  if(arquivo) {
    fprintf(arquivo, "C\t144\t3.1415\t%d\n", x);
    fprintf(arquivo, "Este é um teste mesmo...\n");
    fclose(arquivo);
  } else {
    fprintf(stderr, "Arquivo não pode ser aberto!\n");
  }
  return 0;
}
```

2.1.4 Escreve um bloco de dados

Protótipo:

```
   size_t fwrite (void * buffer, size_t size, size_t count, FILE
* stream);
```

fwrite() retorna o número de elementos escritos no arquivo e pode ser usado em arquivos binários.

```
#include <stdio.h>
typedef
    struct X {
int i; char c; float f; }
X;
int main() {
  FILE * arquivo;
  X var[3];
  var[0].i = 1; var[1].i = 2; var[2].i = 3;
  var[0].c = 'C'; var[1].c = 'B'; var[2].c = 'A';
```

```
  var[0].f = 1.1f; var[1].f = 1.2f; var[2].f = 1.3f;
  if(arquivo = fopen("arq.bin","wb")) {
    fwrite(&var, sizeof(X), 3, arquivo);
    fclose(arquivo);
  }
  return 0;
}
```

2.2 Leitura de arquivos

2.2.1 Lê (recupera) o próximo caracter (unsigned)

Protótipo:

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

2.2.2 Lê uma linha de um stream específico

Protótipo:

```
void fgets (char * string, int maxsize, FILE * stream);
```

Armazena na string apontada em seu argumento. A leitura é interrompida quando foram lidos: 1. (maxsize-1) caracteres; 2. O caracter de nova linha; 3. o caractere "fim de arquivo" (EOF).

2.2.3 Lê entradas formatadas de um stream

Protótipo:

```
void fscanf (FILE * stream, char * format, ...);
```

`fscanf(stdin,"%d",&x)` é equivalente a `scanf("%d",&x)`.

Exemplo: `fscanf(table,"%s\t%d\t%f",&nome,&idade,&salario);` lê de um arquivo onde foi gravado com `fprintf(table,"%s\t%d\t%f",nome,idade,salario);`.

```
#include <stdio.h>
int main() {
  FILE * arquivo;
  char c; int i,j; float f; char s[256];
  arquivo = fopen("ave.tonta", "r");
  if(arquivo) {
    fscanf(arquivo,"%c\t%i\t%f\t%i\n", &c, &i, &f, &j);
    fgets(s, 256, arquivo);
    fclose(arquivo);
    fprintf(stdout,"%c\t%i\t%g\t%i\n%s", c, i, f, j, s);
```

```
  } else {
    fprintf(stderr, "Arquivo não pode ser aberto!\n");
  }
  return 0;
}
```

```
C 144 3.1415 123
Este é um teste mesmo...
```

2.2.4 Leitura de bloco de dados

Protótipo:

```
   size_t fread (void * buffer, size_t buffsize, int numel, FILE
* stream);
```

A função `fread()` também retorna o número de elementos escritos no arquivo, além de poder ser usado com arquivos binários.

```
#include <stdio.h>
typedef
    struct X {
int i; char c; float f; }
X;
int main() {
  FILE * arquivo; int i;
  X var[3];
  if(arquivo = fopen("arq.bin","rb")) {
    fread(&var, sizeof(X), 3, arquivo);
    for(i = 0; i < 3; i++) printf("%i, %c, %f\n",
var[i].i, var[i].c, var[i].f);
    fclose(arquivo);
  }
  return 0;
}
```

```
1, C, 1.100000
2, B, 1.200000
3, A, 1.300000
```

2.3 Fim de arquivo

O final de um arquivo pode ser verificado de duas maneiras:

- Ou pela captura do caractere `EOF` (macro de `stdio.h`);
- Ou pela função de teste `feof()`, que retornará 0 (falso) ou qualquer outra coisa (verdadeiro). Protótipo: `int feof (FILE * stream);`.

3 Movimentando-se pelo arquivo (ponteiros)

3.1 Acesso aleatório (busca)

Para buscas e acessos aleatórios em arquivos, usa-se a função `fseek()`. Esta move a posição corrente de leitura ou escrita no arquivo de um valor especificado (OFFSET), a partir de um ponto especificado. Seu protótipo é:

```
int fseek (FILE * stream, long offset, int origin);
```

O parâmetro origem determina a partir de onde os numbytes de movimentação serão contados. Os valores possíveis são definidos por macros em stdio.h e são:

Table 3: Macros de posição no arquivo

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

3.2 "Rebobinar"

Volta para o começo do arquivo de um fluxo. Protótipo:

```
void rewind (FILE * stream)
```

```
#include <stdio.h>
int main() {
  FILE * arquivo;
  int i;
  arquivo = fopen("sequencia.txt", "w");
  if(arquivo) {
    for(i = 0; i < 10; i++)
      fprintf(arquivo,"%i\n",i);
    fclose(arquivo);
  } else {
    fprintf(stderr, "Arquivo não pode ser aberto!\n");
```

```
  }
  return 0;
}

#include <stdio.h>
int main() {
  FILE * arquivo;
  int i, j;
  arquivo = fopen("sequencia.txt", "r");
  if(arquivo) {
    fseek(arquivo, 6, SEEK_SET);
    do {
      fscanf(arquivo, "%d\n", &j);
      fprintf(stdout, "%d\n", j);
    } while(!feof(arquivo));
    rewind(arquivo);
    fscanf(arquivo, "%d\n", &j);
    fprintf(stdout, "* %d\n", j);
    fclose(arquivo);
  } else {
    fprintf(stderr, "Arquivo não pode ser aberto!\n");
  }
  return 0;
}

3
4
5
6
7
8
9
* 0
```

4 Outras funções

4.1 Apaga arquivo

Para apagar um arquivo, protótipo:

```
int remove(const char *filename)
```

Retorna zero se bem sucedido. Você precisa fechar o stream antes de removê-lo.

4.2 Verifica erros

Protótipo:

```
int ferror (FILE *fp);
```

A função retorna zero, se nenhum erro ocorreu e um número diferente de zero se algum erro ocorreu durante o acesso ao arquivo. Uma função que pode ser usada em conjunto com `ferror()` é a função `perror()` (*print error*), cujo argumento é uma string que normalmente indica em que parte do programa o problema ocorreu.

5 Exemplo de arquivo binário

Gravação e leitura de dados em binário (breve discussão sobre protocolo).

```
#include <stdio.h>
int main() {
  FILE * arquivo;
  struct Ponto { int x; int y; } P[3];
  P[0].x = -32; P[0].y = 32;
  P[1].x = 65; P[1].y = 67;
  P[2].x = 78; P[2].y = -1024;
  arquivo = fopen("bin.dat", "wb");
  if(arquivo) {
    fwrite(P, sizeof(struct Ponto), 3, arquivo);
    fclose(arquivo);
  } else {
    fprintf(stderr, "Arquivo não pode ser aberto!\n");
  }
  return 0;
}
```

```
#include <stdio.h>
int main() {
  FILE * arquivo;
  struct Ponto { int x; int y; } P[3];
  arquivo = fopen("bin.dat", "rb");
  if(arquivo) {
```

```
    fread(P, sizeof(struct Ponto), 3, arquivo);
    fclose(arquivo);
  } else {
    fprintf(stderr, "Arquivo não pode ser aberto!\n");
  }
  fprintf(stdout,"(%d,%d)\n", P[0].x, P[0].y);
  fprintf(stdout,"(%d,%d)\n", P[1].x, P[1].y);
  fprintf(stdout,"(%d,%d)\n", P[2].x, P[2].y);
  return 0;
}
```

```
(-32,32)
(65,67)
(78,-1024)
```

```
#include <stdio.h>
int main() {
  FILE * arquivo;
  unsigned char uc;
  arquivo = fopen("bin.dat", "rb");
  if(arquivo) {
    do {
      fread(&uc, sizeof(char), 1, arquivo);
      fprintf(stdout, "%02X ", uc);
    } while(!feof(arquivo));
    printf("\n");
    fclose(arquivo);
  } else {
    fprintf(stderr, "Arquivo não pode ser aberto!\n");
  }
  return 0;
}
```

```
E0 FF FF FF 20 00 00 00 41 00 00 00 43 00 00 00 4E 00 00 00 00 FC FF FF FF
```