INSTITUTO FEDERAL
Maranhão

Programação Estruturada

- Arquivos (cont.) -

Prof. Ulysses Santos Sousa

ulyssessousa@ifma.edu.br

Aula 11

Roteiro

- Leitura e gravação em blocos de bytes
- stdin, stdout e stderr
- Gravando estruturas em disco
- Funções: rewind, fflush(), fseek(), ftell(), ferror(), perror().

Leitura e gravação de blocos de bytes

- **Funções *fwrite()* e *fread()***
  - Utilizadas quando deseja-se gravar ou ler quantidades maiores de caracteres.
  - São funções apropriadas para acesso de gravação e leitura de matrizes de bytes e estruturas.
  - Essas funções trabalham corretamente apenas no modo binário.

Leitura e gravação de blocos de bytes

```c
/* CopiaArq.C */
/* Mostra o uso de fread() e fwrite() */
#include <stdio.h> /* define FILE */
#include <stdlib.h>

void CopiaArquivo(const char* Origem, const char *Destino);

int main()
{
    char Origem[] = "CopiaArq.C";
    char Destino[]= "CopiaArq.cpy";

    CopiaArquivo(Origem, Destino);
    system("type CopiaArq.cpy");
    system("pause");
      return 0;
}
```

Leitura e gravação de blocos de bytes

```
19  void CopiaArquivo(const char* Origem, const char *Destino)
20  {
21      unsigned char buffer[1024];
22      int BytesLidos;
23
24      FILE *fOrigem, *fDest; /* ponteiro para arquivo */
25
26      if((fOrigem = fopen(Origem,"rb"))==NULL) return;
27      if((fDest = fopen(Destino,"wb"))== NULL) return;
28      do
29      {
30          BytesLidos= fread(buffer,sizeof(char),sizeof(buffer), fOrigem);
31          fwrite(buffer,sizeof(char), BytesLidos, fDest);
32      } while(BytesLidos);
33
34      fclose(fOrigem);
35      fclose(fDest);
36  }
```

Leitura e gravação de blocos de bytes

- **Função *fread()***

```
unsigned fread(void *buffer,
               int tamanho_da_unidade,
               int numero_de_unidades,
           FILE *fp);
```

- No exemplo anterior queremos ler 1024 bytes por vez:

```
BytesLidos = fread(buffer, sizeof(char), sizeof(buffer), fOrigem);
```

  - A função retorna o número de unidades efetivamente lidas. Esse número pode ser menor que 1024 quando o fim do arquivo for encontrado ou ocorrer algum erro.

Leitura e gravação de blocos de bytes

- **Função *fwrite()***

```
unsigned fwrite(void *buffer,
                int tamanho_da_unidade,
                int numero_de_unidades,
                FILE *fp);
```

- No exemplo anterior queremos gravar 1024 bytes a cada vez:

```
fwrite(buffer, sizeof(char), BytesLidos, fDest);
```

  - A função retorna o número de itens gravados. Esse valor será igual a BytesLidos, a menos que ocorra algum erro.

stdin, stdout e stderr

- São os três primeiros elementos da matriz de estruturas FILE;
- São constantes que podem ser usadas para acessar qualquer um dos três arquivos “standard” predefinidos pelo sistema operacional (leitura, gravação e saída de erro).
- Cada uma delas pode ser tratada como um ponteiro para uma estrutura FILE.

stdin, stdout e stderr

- **Por padrão, a leitura é do teclado, a gravação e a saída de erro são do vídeo.**

Ponteiro	Stream
stdin	Entrada padrão (teclado)
stdout	Saída padrão (vídeo)
stderr	Saída de erro (vídeo)

  - Esses ponteiros são constantes, logo não podem ter seus valores alterados, mas podem ser usados como argumentos de funções.
  - Algumas funções, como ***getchar*** e ***putchar***, usam ***stdin*** e ***stdout*** automaticamente.

stdin, stdout e stderr

- **Você pode utilizar os ponteiros *FILE* definidos em *stdio.h* para acessar os arquivos de entrada e saída padrão.**
- **Exemplos:**
  - fgets(string, 80, stdin)
    - Lê um cadeia de caracteres do teclado.
  - fputs(string, stdout)
    - Imprimirá string no vídeo.

Gravando estruturas em disco

- **Para ler ou gravar estruturas utilizaremos as funções *fread()* e *fwrite()***

```
 7   typedef struct Livro
 8   {
 9       char Titulo[50];
10       char Autor[50];
11       short NumReg;
12       double Preco;
13   }Livro;
```

Gravando estruturas em disco

```
15  Livro GetLivro()
16  {
17      Livro livro;
18      printf("\n\tDigite o título: ");
19      gets(livro.Titulo);
20      printf("\tDigite o nome do autor: ");
21      gets(livro.Autor);
22      printf("\tDigite o número do registro: ");
23      scanf("%hd",&livro.NumReg);
24      printf("\tDigite o preço: ");
25      scanf("%lf", &livro.Preco);
26      rewind(stdin);
27      return livro;
28  }
29
30  void PrintLivro(Livro livro)
31  {
32      printf("\n\tTítulo: %s\n", livro.Titulo);
33      printf("\tAutor : %s\n", livro.Autor);
34      printf("\tNo.Reg: %hd\n", livro.NumReg);
35      printf("\tPreço : %.2lf\n", livro.Preco );
36  }
```

Gravando estruturas em disco

```
38  int main()
39  {
40      Livro livro;
41      char resp;
42      FILE *fptr; /* ponteiro para arquivo */
43      /* Abre arquivo para leitura e gravação. Adiciona dados no final*/
44      if((fptr = fopen("Livros.Dat","ab+"))==NULL)
45          exit(1);
46      do
47      {
48          livro = GetLivro();
49          if(fwrite(&livro,sizeof(Livro),1,fptr)!=1)
50              break;
51          printf("Mais um livro (s/n)? ");
52          resp = getche();
53      }while(resp != 'n' && resp != 'N');
54
55      fflush(fptr); /* Esvazia o conteúdo do buffer de saída */
56      fseek(fptr,0,0); /*Coloca o ponteiro no início do arquivo */
57
58      puts("\n\nLISTA DE LIVROS DO ARQUIVO");
59      puts("==========================");
60
61      while(fread(&livro, sizeof(Livro),1,fptr)==1)
62          PrintLivro(livro);
63      fclose(fptr);
64      system("pause");
65      return 0;
66  }
```

Gravando estruturas em disco

- **Função *rewind()***
  - Reposiciona o indicador de posição de arquivo no início do arquivo especificado pelo argumento.
  - Foi utilizada para limpar alguma sobra de bytes no teclado antes da próxima leitura.
  - Protótipo:

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

Gravando estruturas em disco

- **Função *fflush()***
  - Grava o conteúdo de qualquer dado existente no *buffer* para o arquivo associado ao ponteiro que recebe como argumento.
  - Retorna 0 (zero) para indicar sucesso, caso contrário, retorna ***EOF***.
  - Protótipo:

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

Gravando estruturas em disco

- **Ponteiros para arquivos:**
  - Aponta para um byte particular chamado ***posição atual***.
  - Sempre que alguma coisa é gravada ou lida no arquivo, o ponteiro é movido para o fim dessa coisa e a próxima leitura ou gravação começará neste ponto.
  - Quando um arquivo é aberto, o ponteiro é fixado em seu primeiro byte.
  - Se um arquivo é aberto usando a opção ***a (append)***, o seu ponteiro será posicionado no fim do arquivo.

Gravando estruturas em disco

- **Função *fseek()***
  - Permite movimentar a posição corrente de leitura e gravação do arquivo para uma posição escolhida.
  - Exemplo: *fseek(fptr, 0, 0);*
    - Fixa o ponteiro no primeiro byte do arquivo e o descolamento é de 0 bytes a partir do início.

Gravando estruturas em disco

- **Função *fseek()***
  - Argumentos:
    1. Ponteiro para a estrutura FILE;
    2. Deslocamento: consiste no número de bytes que desejamos deslocar a partir da posição especificada pelo terceiro argumento.
    3. Posição: Determinam de onde o deslocamento começará. Existem três possibilidades:

Constante	Posição	Significado
SEEK_SET	0	Início do arquivo
SEEK_CUR	1	Posição atual
SEEK_END	2	Fim do arquivo

Gravando estruturas em disco

- **Função *ftell()***
  - Retorna a posição corrente do ponteiro de um arquivo (em bytes), sempre a partir do início do arquivo.
  - Aceita um único argumento: o ponteiro ***FILE*** do arquivo.
  - Retorna um valor do tipo ***long***, que representa o número de bytes do começo do arquivo até a posição atual.
  - Observação:
    - Pode não retornar o número exato de bytes se for usada com arquivos no modo texto.

Gravando estruturas em disco

- **Função *ferror()***
  - Retorna "verdadeiro" se tiver ocorrido um erro durante a última operação no arquivo ou, caso contrário, retorna zero (falso).

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

Gravando estruturas em disco

- **Função *perror()***
  - Usada para indicar em que parte do programa ocorreu o erro.
  - A mensagem é impressa seguida de vírgula e da mensagem de erro do sistema.

```
void perror(const char *string);
```

Referências

- MIZRAHI, V. V. Treinamento em Linguagem C. 2ª Edição. São Paulo: Person Prentice Hall, 2008.