Linguagem de Programação I

- Arquivos -

Prof. Ulysses Santos Sousa

ulyssessousa@ifma.edu.br

Aula 10

Roteiro

- **Introdução**
- **Leitura e gravação em alto nível**
  - Funções fopen(), fclose() e exit()
  - Funções fputc() e fgetc()
  - Funções fputs() e fgets()
  - Funções fprintf() e fscanf()

Introdução

- **Arquivos**
  - A palavra **arquivo** é usada para indicar um "fluxo de dados" (stream).
  - Todo lugar que tem capacidade de receber bytes da memória do computador ou transferi-los para ela recebe o nome de arquivo.
  - Exemplos: arquivos em disco, teclado, vídeo, impressora, portas de comunicação etc.

Logo, o conceito de arquivos é ampliado no sentido de considerar arquivos não somente os que existem em discos.

Introdução

- **Em C, podemos trabalhar com arquivos por meio de acesso em:**
  - Alto nível
    - Conjunto de funções de acesso *bufferizadas*
  - Baixo nível
    - Conjunto de funções não *bufferizadas*

Leitura e gravação em alto nível

- **Tipo FILE**
  - Quando trabalhamos com arquivos, nosso programa e o sistema operacional devem participar conjuntamente de certas informações sobre o arquivo.
  - Em C, as informações necessárias são guardadas em uma estrutura do tipo ***FILE***, definida no arquivo ***stdio.h.***

Os membros da estrutura FILE possuem informações do arquivo como: seu tamanho atual, localização de seus buffers de dados etc, com os quais o programador não deve ser preocupar, mas voltar seu interesse para a abertura de um arquivo e de como trabalhar com ele.

Leitura e gravação em alto nível

- **Função *fopen()***
  - Executa duas tarefas:
    1. Cria e preenche uma estrutura FILE com a informações necessárias para o programa e o SO.
    2. Retorna um ponteiro do tipo FILE que aponta para a localização na memória dessa estrutura criada.

Leitura e gravação em alto nível

- **Função *fopen()***

```
FILE fopen(const char *nome_arquivo, const char *modo_de_abertura)
```

- **Onde:**
  - O primeiro parâmetro é o nome do arquivo;
  - O segundo parâmetro é o modo de acesso, por exemplo: “r” (leitura), “w” escrita.

Leitura e gravação em alto nível

Modos de abertura de arquivo: MODO TEXTO	
“r”	Abre um arquivo para leitura. Se o arquivo não existir retorna NULL.
“w”	Cria um arquivo para gravação. Se o arquivo já existir, elimina seu conteúdo e recomeça a gravação a partir do seu início.
“a”	Abre um arquivo para gravação, a partir de seu final. Se o arquivo não existir, ele será criado.
“r+”	Abre o arquivo para atualização, ou seja, tanto para leitura como para gravação. Se o arquivo não existir, a operação irá falhar e fopen() retornará NULL.
“w+”	Cria um arquivo para atualização, ou seja, tanto para leitura como para gravação. Se o arquivo já existir, seu conteúdo será destruído.
“a+”	Abre um arquivo para atualização, gravando novos dados a partir do final do arquivo. Se o arquivo não existir, ele será criado.

Leitura e gravação em alto nível

Modos de abertura de arquivo: BINÁRIO	
“rb”	Abre um arquivo para leitura. Se o arquivo não existir, a operação irá falhar e **fopen()** retornará NULL.
“wb”	Cria um arquivo para gravação. Se o arquivo já existir, elimina seu conteúdo e recomeça a gravação a partir de seu início.
“ab”	Abre um arquivo para gravação, a partir do seu final. Se o arquivo não existir, será criado.
“rb+”	Abre um arquivo para atualização, ou seja, tanto para leitura como para gravação. Se o arquivo não existir, o operação irá falhar e **fopen()** retornará NULL.
“wb+”	Cria um arquivo para atualização, ou seja, tanto para leitura como para gravação. Se o arquivo já existir, seu conteúdo será destruído.
“ab+”	Abre um arquivo para atualização, gravando novos dados a partir do final do arquivo. Se o arquivo não existir, ele será criado.

Leitura e gravação em alto nível

- **Formas de leitura e gravação:**
  - fputc() e fgetc()
    - Gravar e ler um caractere por vez.
  - fputs() e fgets()
    - Gravar e ler linha a linha.
  - fprintf() e fscanf()
    - Gravar e ler dados formatados.
  - fwrite() e fread()
    - Gravar e ler blocos de bytes.

Leitura e gravação de um caractere por vez

- **fputc()**
  - Grava um caractere por vez no arquivo.
  - Recebe dois argumentos:
    - O caractere a ser gravado.
    - O ponteiro para estrutura FILE do arquivo.
  - Retorna o caractere gravado ou EOF se acontecer algum erro.

```
int fputc(int caractere, FILE *ponteiro_arquivo);
```

Leitura e gravação de um caractere por vez

- **Função *fclose()***
  - Fecha o arquivo associado ao ponteiro FILE enviado como argumento.

Por que fechar arquivos?

1. O programa força a gravação do buffer no arquivo.
2. Para liberar as áreas de comunicação usadas, para que estejam disponíveis a outros arquivos. Essas áreas incluem a estrutura FILE e o *buffer*.

Leitura e gravação de um caractere por vez

- **Função *exit()***
  - Também fecha arquivos, mas fecha todos os arquivos que estiverem abertos.
  - Encerra o programa e devolve o controle ao Sistema Operacional.

Leitura e gravação de um caractere por vez

- **Exemplo - fputc()**

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

int main()
{
    FILE *fptr; //ponteiro para arquivo
    char ch;
    fptr = fopen("arqtext.txt", "w");
    while((ch=getche()) != '\r')
        fputc(ch, fptr);
    fclose(fptr);
    return 0;
}
```

Leitura e gravação de um caractere por vez

- **Exemplo - fgetc()**

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

int main(void)
{
    FILE *fptr;
    short int ch;
    fptr = fopen("arqtext.txt", "r");
    while((ch=fgetc(fptr)) != EOF)
        printf("%c", ch);
    fclose(fptr);
    return 0;
}
```

Leitura e gravação em alto nível

- **Função *feof()***
  - Retorna *verdadeiro* se o final do arquivo tiver atingido; caso contrário, retorna *falso*:

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

int main(void)
{
    FILE *fptr;
    short int ch;
    fptr = fopen("arqtext.txt", "r");
    while(!feof(fptr)){
        ch = fgetc(fptr);
        printf("%c", ch);
    }
    fclose(fptr);
    return 0;
}
```

```
int feof(FILE *p);
```

Leitura e gravação em alto nível

- **Constante *EOF***
  - Não é um caractere que pertence ao arquivo e sim uma constante, do tipo ***short int.***
  - É definida no arquivo ***stdio.h*** com o valor 0xFFF ou -1.
  - O arquivo ***stdio.h*** define ***EOF*** com o valor correto para o seu sistema operacional, logo devemos utilizar esta constante para testar o fim do arquivo.

Leitura e gravação em alto nível

- **Cuidados ao abrir o arquivo:**
  - Verificar se o arquivo foi aberto com sucesso antes de ler ou escrever nele.

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

int main(void)
{
    FILE *fptr;
    short int ch;
    if ((fptr = fopen("arqtex.txt", "r")) == NULL)
    {
        puts("Nao foi possivel abrir o arquivo.");
        exit(1);
    }
    while(!feof(fptr)){
        ch = fgetc(fptr);
        printf("%c", ch);
    }
    fclose(fptr);
    return 0;
}
```

Leitura e gravação linha a linha

- **Função *fputs()***
  - Recebe dois argumentos:
    - Ponteiro char, para a cadeia de caracteres a ser gravada;
    - Ponteiro para a estrutura FILE do arquivo.
  - Retorna um número positivo ou EOF se acontecer algum erro.
  - Não adiciona automaticamente o caractere de nova linha no fim dela.

```
int fputs(const char *string, FILE *ponteiro_arquivo)
```

Leitura e gravação linha a linha

```c
/* OFileStr.C */
/* Grava strings no arquivo */
#include <stdio.h> /* define FILE */
#include <stdlib.h>

int main(void)
{
    FILE *fptr;  /* ponteiro para arquivo */

    fptr = fopen("TesteSTR.txt","w");/*Cria p/gravar em modo texto */

    fputs("Um grande antídoto contra o egoísmo\n",fptr);
    fputs("é a generosidade... Dê, mesmo que\n", fptr);
    fputs("isso requeira de você um esforço\n", fptr);
    fputs("consciente. Pelo fato de partilhar\n",fptr);
    fputs("tudo o que possui, seu egoísmo se\n",fptr);
    fputs("abrandará.\n",fptr);
    fclose(fptr);
    system("TYPE TesteSTR.txt");
    system("PAUSE");
    return 0;
}
```

Leitura e gravação linha a linha

- **Função *fgets()***
  - Recebe três argumentos:
    - O ponteiro char para a cadeia de caracteres em que os dados lidos do arquivo serão colocados.
    - O número máximo de caracteres a serem lidos.
    - O ponteiro para a estrutura FILE do arquivo.
  - Retorna um ponteiro para a cadeia de caracteres ou NULL se encontrar alguma erro ou fim do arquivo.

```
char *fgets(char string, int limite, FILE *ponteiro_arquivo);
```

Leitura e gravação linha a linha

```c
/* IFileStr.C */
/* Lê linha a linha do arquivo */
#include <stdio.h> /* define FILE */
#include <stdlib.h>

int main(void)
{
    FILE *fptr;  /* ponteiro para arquivo */
    char str[81];

    /*Abre p/leitura em modo texto*/
    if( (fptr = fopen(("TesteSTR.txt","r")) == NULL)
    {
        puts("Não foi possível abrir o arquivo");
        exit(1);
    }

    while(fgets(str,80,fptr) != NULL) /*Lê uma linha de texto */
        printf("%s",str);

    fclose(fptr);
    system("pause");
    return 0;
}
```

Leitura e gravação de dados formatados

- **Função *fprintf*()**
  - É similar ao ***printf()***, exceto pelo fato de que o ponteiro para ***FILE*** é tomado como argumento.
  - Todas as possibilidades de formatação de ***printf()*** operam com ***fprintf()***.

```
int fprintf(FILE *ponteiro_arquivo, const char *formato, [argumentos]);
```

Leitura e gravação de dados formatados

```c
/* OFileFormat.C */
/* Grava dados formatados num arquivo */
#include <stdio.h> /* define FILE */
#include <stdlib.h>
#include <string.h>
#define TRUE 1

int main(void)
{
    FILE *fptr; /* ponteiro para arquivo */
    char titulo[30];
    int regnum;
    double preco;
    fptr = fopen("Livros.txt","w");
    while (TRUE)
    {
        printf("\nDigite titulo, registro e preco do livro: ");
        scanf("%s %d %lf",titulo,&regnum,&preco);
        if(strlen(titulo) <= 1) break;
        fprintf(fptr,"%s %d %.2lf\n",titulo,regnum,preco);
    }

    fclose(fptr);
    system("pause");
    return 0;
}
```

Leitura e gravação de dados formatados

- **Função *fscanf()***
  - Similar à função ***scanf()***, exceto pelo fato de que, como em ***fprintf()***, um ponteiro para **file** deverá ser incluído como primeiro argumento.

```
int fscanf(FILE *ponteiro_arquivo, const char *formato, [argumento]);
```

Leitura e gravação de dados formatados

```c
/* IFileFormat.C */
/* Lê dados formatados do arquivo */
#include <stdio.h> /* define FILE */
#include <stdlib.h>

int main(void)
{
    FILE *fptr; /* ponteiro para arquivo */
    char titulo[30];
    int regnum;
    double preco;
    fptr = fopen("Livros.txt","r");

    while ( fscanf(fptr,"%s %d %lf", titulo, &regnum, &preco) != EOF)
        printf("%s %d %.2lf\n", titulo, regnum, preco);

    fclose(fptr);
    system("pause");
      return 0;
}
```

Referências

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