Unoeste

Algoritmos e Estruturas de Dados

Arquivos

Profª. Me. Carla Plantier Message

2022

Introdução

- Arquivos são coleções de bytes identificados por um nome único.
- Operações em discos são realizadas utilizando-se arquivos.

Arquivos

- Por que usar arquivos?
  - Permitem armazenar grande quantidade de informação;
  - Persistência dos dados (disco);
  - Acesso aos dados poder ser não sequencial;
  - Acesso concorrente aos dados (mais de um programa pode usar os dados ao mesmo tempo).

Arquivos do tipo texto e binário

- Uma maneira de classificar operações de acesso a arquivos é de acordo com a forma como estes são abertos: em modo texto ou em modo binário.

- Um arquivo aberto em modo texto é interpretado em C como sequências de caracteres agrupadas em linhas.

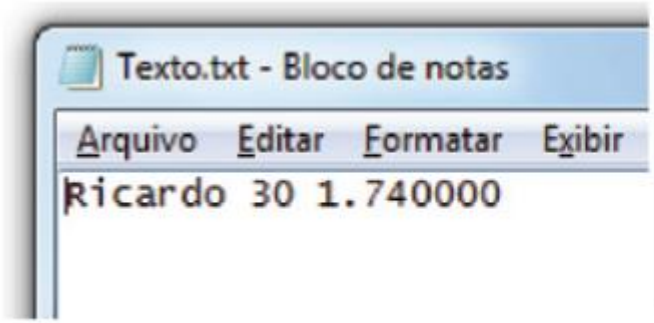
- As linhas são separadas por dois caracteres (13 decimal e 10 decimal), que são convertidos para um caractere único quando o arquivo é gravado (caractere de nova linha).

- Em um arquivo aberto em modo binário qualquer caractere é lido ou gravado sem alterações.

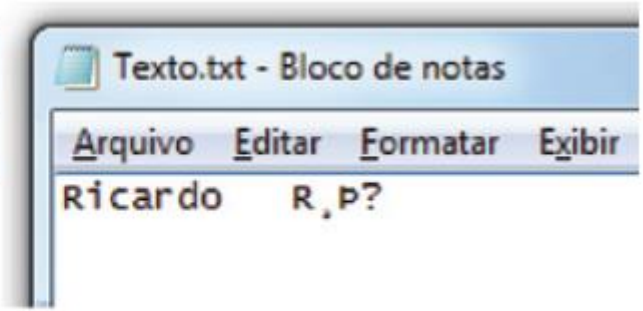
- Na forma de texto os números são armazenados como cadeias de caracteres e na forma binária estes são armazenados como estão na memória.

Os trechos abaixo possuem os mesmos dados

```
char nome[20] = "Ricardo";
int i = 30;
float a = 1.74;
```



Arquivo Texto



Arquivo Binário

Manipulando arquivo em C

- A linguagem C possui uma série de funções para manipulação de arquivos, cujos protótipos estão reunidos na biblioteca padrão de entrada e saída, **stdio.h**.

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

- A linguagem C não possui funções que automaticamente leiam todas as informações de um arquivo.
  - Suas funções se limitam a abrir/fechar e ler caracteres/bytes
  - É tarefa do programador criar a função que lerá um arquivo de uma maneira específica.

Utilização de ponteiros na declaração de arquivos

- Um ponteiro de arquivo é um apontador para suas informações (como nome, status e posição no disco).

- Para ler ou escrever em arquivos, um programa em C precisa utilizar ponteiros de arquivo, que devem ser declarados como variáveis ponteiros do tipo FILE.

- A palavra FILE refere-se a uma estrutura de arquivos definida na biblioteca stdio.h.
- Exemplo de declaração:
- FILE *fp;
- **fp** é o ponteiro para arquivos que nos permitirá manipular arquivos no C.

Função fopen()

- A função fopen( ) associa um arquivo a um buffer de memória. Em sua chamada devem ser utilizados dois parâmetros.
- O primeiro parâmetro refere-se ao nome do arquivo a ser aberto (que pode incluir uma especificação do caminho de pesquisa).
- O segundo parâmetro refere-se ao modo como o arquivo será aberto (para leitura ou escrita, em formato binário ou texto, etc).

Há três opções de abertura:

- “w” – para gravação
- “r” – para leitura
- “a” – para adição de dados
- Estas opções podem ser combinadas com dois modificadores:
- “b” – para abertura no modo binário
- “+” – para indicar que o arquivo já existe e deve ser atualizado.

Exemplo

- FILE *fp;
- fp = fopen(“teste.txt”,”wb”);
- A função fopen( ) retorna um ponteiro para a estrutura FILE, onde estão armazenadas informações sobre o arquivo.
- Caso ocorra um erro na abertura, este ponteiro será nulo.

- Um arquivo binário pode ser aberto para escrita utilizando o seguinte conjunto de comandos:
  - A condição **fp==NULL** testa se o arquivo foi aberto com sucesso. No caso de erro a função **fopen()** retorna um ponteiro nulo (**NULL**).

```
int main(){
    FILE *fp;
    fp = fopen("exemplo.bin","wb");
    if(fp == NULL)
        printf("Erro na abertura do arquivo.\n");

    fclose(fp);

    return 0;
}
```

Fechando um arquivo

- Após a gravação de um arquivo, é necessário fechá-lo.
- Fechar um arquivo faz com que qualquer caractere que tenha permanecido no buffer de memória seja gravado no disco.
- Além disso as áreas de comunicação (estrutura FILE e buffer) são liberadas.

Função fclose()

- A função fclose( ) simplesmente fecha o arquivo associado ao ponteiro FILE utilizado como argumento.
- Exemplo:
- fclose (fp);

Fim de arquivo

- A marca EOF (End of File) é um sinal enviado pelo sistema operacional ao programa em C para indicar o final de um arquivo.

- A marca de fim de arquivo pode ser diferente para diferentes sistemas operacionais.

- Assim, o valor de EOF pode ser qualquer. A biblioteca stdio.h define EOF com o valor correto para o sistema operacional sendo utilizado. Assim, no programa em C, a constante EOF deve ser utilizada para verificação de fim de arquivo.

Leitura e escrita de dados em arquivos

- Em linguagem C é possível ler e gravar arquivos de diferentes maneiras:

- Leitura e escrita de um caractere por vez - funções **getc( )** e **putc( )**;
- Leitura e escrita de strings – funções **fgets( )** e **fputs( )**;
- Leitura e escrita de dados formatados – funções **fscanf( )** e **fprintf( )**;
- Leitura e escrita de registros ou blocos – funções **fread( )** e **fwrite( )**;

Lendo um caractere de um arquivo

- A função getc( ) lê caracteres de um arquivo aberto no modo leitura. Utiliza como parâmetro o ponteiro para o arquivo declarado.

- A função getc( ) devolve EOF quando o fim do arquivo é alcançado.

Exemplo

```c
#include<stdio.h>
main()
{
FILE *fp;
int ch;
fp = fopen("arquivo.txt","r");
do
{
    ch = getc(fp);
    if (ch!=EOF)
    {
        printf("%c",ch); // apresenta na tela
    }
} while (ch != EOF);
  fclose(fp);
}
```

Escrevendo em um arquivo

- A função putc( ) escreve caracteres em um arquivo que foi previamente aberto no modo de escrita.
- Utiliza como parâmetros o caractere a ser escrito e o ponteiro para o arquivo declarado. Se a operação putc( ) for bem sucedida, devolve o caractere escrito. Caso contrário, devolve EOF.

Exemplo

```c
#include<stdio.h>
#include<stdlib.h>
main()
{
    FILE *fp;
    char ch;
    fp = fopen("arquivo.txt","w");
    do
    {
        printf ("Digite um caractere <espaco finalizar> ");
        fflush(stdin);
        scanf("%c",&ch);
        if (ch !=' ') // espaco finaliza a leitura
        {
            putc(ch,fp); // colocar o caractere no arquivo
        }
    }while (ch != ' ');
    fclose(fp);
}
```

Unoeste