AP2C2 – Arquivos

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

Arquivos *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.

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*. A seguir, um exemplo de declaração:

```
FILE *fpt;
```

Abrindo um arquivo

Quando se solicita ao sistema operacional a abertura de um arquivo, a informação retornada é um ponteiro para a estrutura FILE.

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 *fpt
fpt = 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.

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. Porém, a biblioteca *stdio.h* define EOF com o valor correto para o sistema operacional utilizado.

Fechando um arquivo

Após a gravação de um arquivo, é necessário fechá-lo, fazendo 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, como exemplo a seguir:

```
fclose (fpt);
```

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.

Escrevendo um caractere 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.

Lendo uma string de um arquivo

A função *fgets( )* lê uma string especificada até que um caractere de nova linha seja lido ou que n-1 caracteres tenham sido lidos. Se um caractere de nova linha é lido, este fará parte da string. A função retorna um ponteiro para a string (ou um ponteiro nulo se ocorrer um erro).

A função *fgets( )* utiliza 3 argumentos. O primeiro é um ponteiro para o buffer onde será armazenada a linha lida. O segundo é um número inteiro que indica o limite máximo de caracteres a serem lidos. Este número deve ser pelo menos 1 maior que o número de caracteres lidos, pois um caractere NULL ('\0') é acrescentado na próxima posição livre. O terceiro argumento é um ponteiro para a estrutura FILE do arquivo que está sendo lido. A função termina a leitura após ler um caractere de nova linha ('\n') ou um caractere de fim de arquivo (EOF).

Escrevendo uma string em um arquivo

A função *fputs( )* efetua a gravação de uma string de caracteres em um arquivo. Utiliza como parâmetros a string a ser gravada e o ponteiro para o arquivo onde a string será armazenada.

Lendo e escrevendo dados formatados

As funções *fprintf( )* e *fscanf( )* são utilizadas para ler e armazenar dados formatados de diferentes maneiras (inteiros, reais, caracteres).

Lendo e escrevendo registros em arquivos

As funções *fread( )* e *fwrite( )* são utilizadas para ler e armazenar dados de matrizes, estruturas, ou mesmo matrizes de estruturas através de uma única instrução.

A função *fwrite( )* utiliza 4 parâmetros:

- o primeiro é um ponteiro para a localização de memória do dado a ser gravado;

- o segundo é um número inteiro que indica o tamanho do tipo de dado a ser gravado, podendo utilizar a função *sizeof( )*, que retorna o tamanho de um dado;

- o terceiro parâmetro é um número inteiro que informa quantos itens do mesmo tipo serão gravados;

- o quarto parâmetro é um ponteiro para a estrutura FILE do arquivo que se quer gravar.

Acesso Aleatório

A leitura e gravação de dados é realizada de forma sequencial, ou seja, toma-se um grupo de itens (caracteres, strings ou estruturas mais complexas) que são armazenados em sequência.

Na leitura, começa-se no início do arquivo, que é percorrido até chegar-se ao final. No entanto, é possível também acessar arquivos de forma aleatória, ou seja, acessar um determinado item sem necessidade de percorrer todo o arquivo.

A função *fseek*( ) faz com que o ponteiro de arquivo aponte para uma posição específica dentro do arquivo. Esta função utiliza 3 parâmetros:

- o primeiro é o ponteiro para a estrutura FILE do arquivo;

- o segundo parâmetro é chamado *offset* e consiste no número de bytes, a partir da posição especificada pelo terceiro parâmetro;

- o terceiro parâmetro é o deslocamento do ponteiro.

A seguir, um exemplo:

```
fseek (fptr,offset,0);
```

Após a chamada a *fseek()*, este ponteiro será movimentado para a posição desejada.

Exercícios

1) Desenvolva um programa que leia um arquivo de texto e conte o número de caracteres em seu conteúdo.
2) Desenvolva um programa que leia nomes pelo teclado e a cada nome lido grave-o em um arquivo.
3) Desenvolva um programa que leia os nomes do arquivo criado no exercício anterior e mostre na tela o número de caracteres de cada nome.
4) Desenvolva um programa que leia 2 notas bimestrais de alunos e armazene os dados de cada aluno em linhas de um arquivo. O programa deve conter também uma função que leia o arquivo e calcule e mostre na tela a média das notas de cada aluno.