Funções para trabalhar com arquivos

Algoritmos e Estruturas de Dados

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.

fseek( )

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. Após a chamada a fseek, este ponteiro será movimentado para a posição desejada.
O **segundo** parâmetro é chamado offset e consiste no número de bytes, a partir da posição especificada pelo **terceiro** parâmetro, de deslocamento do ponteiro. A função devolve 0, quando bem sucedida.
Exemplo:

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

Outro exemplo: int fseek (FILE *fp,long numbytes,int origem);

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:

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

Tendo-se definido a partir de onde irá se contar, numbytes determina quantos bytes de deslocamento serão dados na posição atual.

rewind()

Outra opção de movimentação pelo arquivo é simplesmente retornar para o seu início.
Para tanto, usa-se a função **rewind**:

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

ftell()

Retorna a posição do ponteiro de um arquivo binário em relação ao seu começo.

Esta função aceita um único argumento, que é o ponteiro para a estrutura FILE do seu arquivo e retorna um valor do tipo long, que representa o número de bytes do começo do arquivo até a posição atual.

Sintaxe: long ftell(pont tipo FILE);

Apagando um arquivo

Além de permitir manipular arquivos, a linguagem C também permite apagá-lo do disco. Isso pode ser feito utilizando a função **remove()**:

```
int remove(char *nome_do_arquivo);
```

Diferente das funções vistas até aqui, esta função recebe o **caminho e nome** do arquivo a ser excluído, e não um ponteiro para FILE.
Como retorno temos um valor inteiro, o qual será igual a 0 se o arquivo for excluído com sucesso.

Exemplo

```c
int main(){
    int status;
    status = remove("ArqGrav.txt");
    if(status != 0){
        printf("Erro na remocao do arquivo.\n");
        system("pause");
        exit(1);
    }else
        printf("Arquivo removido com sucesso.\n");

    return 0;
}
```

Verificando final de Arquivo

Além da constante EOF, temos a função **feof().**

Sintaxe: int feof (nomebuf)

A função feof é utilizada para a verificação do final do arquivo.

Essa função retorna um valor inteiro caso o ponteiro chegou no final do arquivo, caso contrário o retorno é zero.

Essa função pode ser utilizada **quando a leitura é feita caractere a caractere ou de qualquer outra forma.**

Onde: nomebuf é o nome da variável que está sendo utilizada como ponteiro para o buffer do arquivo onde será feita a verificação de final de arquivo.

Renomeando um arquivo

A função **rename()** altera o nome do arquivo.

int rename(char nomeantigo, char nomenovo)

A função retorna 0 se deu certo, ou outro int se não der certo.