Programação Estruturada/Programação II: Manipulação de Arquivos em C

Introdução

A manipulação de arquivos é essencial em programação, pois permite armazenar e recuperar informações de forma persistente. Em C, utilizamos a biblioteca `<stdio.h>` para lidar com arquivos, possibilitando operações como leitura, escrita e fechamento de arquivos.

Para manipular arquivos em C, utilizamos uma variável especial do tipo FILE*. Formalmente, FILE* é um ponteiro para uma estrutura do tipo FILE, definida na biblioteca padrão <stdio.h>. Essa estrutura armazena informações sobre o arquivo aberto, como sua posição atual, modo de acesso e o estado do fluxo de dados. Podemos pensar no FILE* como um "controle remoto" que permite ao programa realizar operações de leitura, escrita e navegação dentro do arquivo, sem precisar conhecer os detalhes de como o sistema operacional gerencia os arquivos internamente.

O que é NULL?

Sempre que tentamos abrir um arquivo, o sistema pode falhar por diversos motivos, como o arquivo não existir ou o programa não ter permissão para acessá-lo. Quando isso acontece, a função `fopen()` retorna um valor especial chamado `NULL`, que indica que a abertura do arquivo falhou.

Abrindo um Arquivo

Para manipular um arquivo, utilizamos a função `fopen()`, que retorna um "controle remoto" do arquivo. Exemplo de abertura de um arquivo para escrita:

```c
#include <stdio.h>

int
main()
{
    FILE* arquivo;
    arquivo = fopen("dados.txt", "w");

    if (arquivo == NULL)
    {
        printf("Erro ao abrir o arquivo!\n");
        return (1);
    }

    printf("Arquivo aberto com sucesso!\n");
    fclose(arquivo);
    return (0);
}
```

Modos de Abertura

A função `fopen()` recebe dois argumentos: o nome do arquivo e o modo de abertura. Alguns modos comuns são:

- `"r"` – Abre para leitura (arquivo deve existir).
- `"w"` – Abre para escrita (cria um novo arquivo ou apaga o conteúdo existente).
- `"a"` – Abre para anexar dados (mantém o conteúdo e adiciona novos dados ao final).
- `"r+"` – Abre para leitura e escrita.
- `"w+"` – Abre para leitura e escrita, apagando o conteúdo existente.
- `"a+"` – Abre para leitura e escrita, mantendo o conteúdo e permitindo adicionar novos dados.

Escrevendo em um Arquivo

Para escrever em um arquivo, utilizamos a função `fprintf()`:

```c
#include <stdio.h>

int
main()
{
    FILE* arquivo = fopen("dados.txt", "w");

    if (arquivo == NULL)
    {
        printf("Erro ao abrir o arquivo!\n");
        return (1);
    }

    fprintf(arquivo, "Este e um exemplo de escrita em arquivo.\n");

    fclose(arquivo);
    return (0);
}
```

Lendo um Arquivo

Podemos utilizar **fgets()** para ler uma linha de texto ou **fscanf()** para ler dados formatados, assim como usamos `scanf()` no teclado.

Usando fgets()

```c
#include

int
main()
{
    FILE* arquivo = fopen("dados.txt", "r");
    char linha[100];
```

```
    if (arquivo == NULL)
    {
        printf("Erro ao abrir o arquivo!\n");
        return (1);
    }

    while (fgets(linha, sizeof(linha), arquivo) != NULL)
    {
        printf("%s", linha);
    }

    fclose(arquivo);
    return (0);
}
```

Usando fscanf()

A função **fscanf()** funciona de maneira parecida com o `scanf()`, mas lê os dados de um arquivo ao invés de ler da entrada padrão. É muito útil quando o arquivo tem dados organizados em um formato previsível.

Por exemplo, imagine que você gravou dados em um arquivo assim:

```
Alice 20
Bruno 35
```

Podemos ler esses dados com `fscanf()`:

```
#include <stdio.h>

int
main()
{
    FILE* arquivo = fopen("dados.txt", "r");
    char nome[50];
    int idade;

    if (arquivo == NULL)
    {
        printf("Erro ao abrir o arquivo!\n");
        return (1);
    }

    // Lê até o final do arquivo
    while (fscanf(arquivo, "%s %d", nome, &idade) == 2)
    {
        printf("Nome: %s | Idade: %d\n", nome, idade);
    }
```

```
    fclose(arquivo);
    return (0);
}
```

No exemplo acima, cada vez que chamamos `fscanf()`, ele tenta ler uma string (até o primeiro espaço) e um inteiro, e armazena nos endereços das variáveis. A função retorna o número de itens lidos com sucesso.

Fechando um Arquivo

Sempre que terminarmos de usar um arquivo, devemos fechá-lo com `fclose()`, para evitar perdas de dados e liberar recursos do sistema.

```
fclose(arquivo);
```

Exercícios de Fixação

1. Crie um programa em C que solicite ao usuário seu nome e idade e salve essas informações em um arquivo chamado dados.txt. Cada execução do programa deve sobrescrever os dados anteriores.

Requisitos:

- Usar a função fopen() no modo de escrita ("w").
- Utilizar fprintf() para escrever os dados no arquivo.
- Fechar o arquivo com fclose().

2. Crie um programa em C que abra o arquivo dados.txt, criado no exercício anterior, e exiba seu conteúdo na tela.

Requisitos:

- Abrir o arquivo no modo de leitura ("r").
- Usar fgets() para ler os dados linha por linha.
- Exibir o conteúdo na tela com printf().
- Tratar o erro caso o arquivo não exista.

3. Escreva um programa para criar o arquivo "multiplos7.txt" e gravar em seu conteúdo os números múltiplos de sete de 1 até 94365, um em cada linha.

4. Crie um arquivo chamado "entrada.txt" com o conteúdo abaixo:

```
pao
joao
marciano
meridiano
colossal
fibrose
merito
```

```
brito
nao
```

Faça um programa em C que leia o conteúdo do arquivo linha por linha usando o fscanf(). A cada linha lida, conte e imprima a quantidade de letras da string.