Arquivos - Parte 2

Professor: Guilherme Corredato Guerino

Disciplina: Algoritmos e Técnicas de Programação

Ciência da Computação
Universidade Estadual do Paraná

Outras operações

- Além das **operações básicas**, outras **operações** também podem ser realizadas **com arquivos**.

- **Entre essas outras operações estão:**
  - Capturar erros no uso de arquivos
  - Voltar o cursor ao início do arquivo

Outras operações

- Além das **operações básicas**, outras **operações** também podem ser realizadas **com arquivos**.

- **Entre essas outras operações estão:**
  - Apagar um arquivo
  - Renomear um arquivo

Capturar Erros

- Para **capturar erros** durante uma **operação com arquivos**, utiliza-se a função **ferror()**.

- **Sintaxe:**

```
int ferror (FILE* ponteiro_arquivo);
```

Observações

- A função ferror() **retorna um número inteiro** e deve ser chamada **logo depois que outra função que manipula arquivo for chamada**.

- Se o **número retornado** pela função for **diferente de zero**, significa que **ocorreu um erro** durante a **última operação realizada com o arquivo**. Se o número **for zero**, **não ocorreu erro**.

Observações

- **ferror()** se torna muito útil quando queremos verificar se cada acesso a um arquivo teve sucesso, de modo que consigamos garantir a integridade dos nossos dados.
- Na maioria dos casos, se um arquivo pode ser aberto, ele pode ser lido ou gravado. Porém, existem situações em que isto não ocorre.
  - Por exemplo, pode acabar o espaço em disco enquanto gravamos, ou o disco pode estar com problemas e não conseguimos ler, etc.

```c
#include <stdio.h>
#include <stdlib.h>
int main()
{
    FILE *pf;
    char string[100];
    if((pf = fopen("arquivo.txt","w")) ==NULL)
    {
        printf("\nNao consigo abrir o arquivo ! ");
        exit(1);
    }
    do
    {
        printf("\nDigite uma nova string. Para terminar, digite <enter>: ");
        gets(string);
        fputs(string, pf);
        putc('\n', pf);
        if(ferror(pf))
        {
            perror("Erro na gravacao");
            fclose(pf);
            exit(1);
        }
    } while (strlen(string) > 0);
    fclose(pf);
}
```

Voltar o Cursor ao Início

- **Cursor** é um **ponteiro** que **indica a partir de qual posição**, dentro de um **arquivo**, uma determinada **operação** será executada.

- Quando o **arquivo é aberto**, o **cursor aponta para a posição zero**, onde está o **primeiro byte do arquivo**.

Voltar o Cursor ao Início

- Caso seja realizada uma **leitura** por exemplo, o **cursor se movimentará de acordo com a quantidade de bytes que forem lidos**.

- O cursor pode ser **reposicionado ao início do arquivo**, por meio da função **rewind()**.

Voltar o Cursor ao Início

- **Sintaxe:**

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

```c
#include <stdio.h>

int main() {
    FILE* arquivo = fopen("arquivo.txt", "w+");
    if(arquivo == NULL) {
        fprintf(stderr, "Erro ao tentar abrir arquivo.txt.");
        return 1;
    }

    int a = getc(arquivo);
    rewind(arquivo);
    int b = getc(arquivo);
    fclose(arquivo);

    printf("%c == %c\n", a, b); // a == b, o mesmo caractere foi lido

    return 0;
}
```

Apagar um Arquivo

- Para apagar um arquivo, utiliza-se a função **remove()**.

- **Sintaxe:**

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

Onde:
nome_arquivo: indica o **nome físico** do arquivo que será removido. **O caminho do arquivo pode ser incluído**.

Apagar um Arquivo

- Se o valor retornado pela função remove() for **zero**, significa que a **remoção foi realizada com êxito**.

- Caso contrário, **será retornado qualquer outro valor**.

- **Observação:** é importante **fechar o arquivo** antes de **apagá-lo**.

```c
#include <stdio.h>

int main() {
    char* caminho = "C:\\arquivo.txt";

    if(remove(caminho) == 0) {
        printf("Arquivo removido com sucesso!");
    } else {
        printf("Não foi possível remover o arquivo %s.", caminho);
    }

    return 0;
}
```

Renomear um Arquivo

- Para trocar o nome de um arquivo, utiliza-se a função **rename()**.

```
int rename (char* nome_atual, char* nome_novo);
```

Onde:
nome_atual: nome físico do arquivo. O caminho pode ser incluído.
nome_novo: novo nome físico do arquivo. O caminho pode ser incluído.

```
#include <stdio.h>

int main() {
    // caminhos no formato do sistema operacional Windows
    char* caminho = "C:\\arquivo.txt";
    char* caminho_novo = "C:\\arq.txt";

    if(rename(caminho, caminho_novo) == 0) {
        printf("Arquivo %s renomeado para %s com sucesso!\n", caminho,
caminho_novo);
    } else {
        printf("Não foi possível renomear o arquivo %s.", caminho);
    }

    return 0;
}
```