Aula 08 - Arquivos

Leonardo Anjoletto Ferreira

Assuntos

- leitura e escrita
- opções e retornos das funções
- outras funções

Do jeito mais fácil...

- `printf` vira `fprintf`
- `scanf` vira `fscanf`
- `fopen` para abrir um arquivo
- `fclose` para fechar um arquivo
- `feof` para saber se chegou ao final do arquivo

Exemplo de escrita em arquivo

```c
#include <stdio.h>

int main(){
    // abre o arquivo para escrita
    FILE *f = fopen("arquivo.txt", "w");

    fprintf(f, "test\n"); // escreve uma linha
    fprintf(f, "%d", 42); // escreve uma variável

    fclose(f); // fecha o arquivo
}
```

Exemplo de leitura de arquivo

```c
#include <stdio.h>
int main(){
    // abre o arquivo para leitura
    FILE *f = fopen("arquivo.txt", "r");
    char c[255]; // para armazenar o que for lido

    fscanf(f, "%s", &c); // le a primeira linha
    printf("%s\n", c);   // exibe a linha
    fscanf(f, "%s", &c); // le a próxima linha
    printf("%s\n", c);  // exibe o que foi lido
    fclose(f);          // fecha o arquivo
}
```

Exemplo de leitura sem saber tamanho do arquivo

```c
#include <stdio.h>

int main(){
    FILE *f = fopen("arquivo.txt", "r");
    char linha[255];
    int cont = 0;

    // enquanto não chegar ao final do arquivo
    while (!feof(f)){
        fscanf(f, "%s", linha);
        printf("%d: %s\n", cont++, linha);
    }
}
```

Opções de abertura de arquivo

opção	resultado
`w`	escrita
`r`	leitura
`a`	adição (appending)
`b`	binário

Opções de abertura de arquivo

A diferença entre um arquivo de texto e um binário é na forma de escrita dos dados: enquanto o binário escreve cada byte em disco sem realizar nenhuma conversão, o arquivo de texto converte cada caracter.

Portanto, também temos que tomar cuidado com os caracteres de quebra de linha (`\n`) e retorno de carro (`\r`) nos arquivos de texto.

Opções de abertura de arquivo

Para abrir um arquivo como leitura e escrita, adicionamos um `+`:

opção	resultado
`w+`	aponta para o começo do arquivo
`r+`	sobrescreve o arquivo
`a+`	aponta para o final do arquivo

Retornos das funções

Como sabemos, é comum que funções em C retornem inteiros que indentificam o resultado da chamada da função.

Retorno de `fopen`

- Retorna um ponteiro para o arquivo segundo as opções escolhidas
- Se o arquivo não for encontrado, retorna `NULL`

```c
#include <stdio.h>

int main(){
    FILE *f = fopen("erro", "r");

    if (f == NULL)
        printf("arquivo nao encontrado");
    else
        printf("arquivo aberto para leitura");
}
```

Retorno das outras funções

- `fclose` :
  - 0 se conseguiu fechar o arquivo
  - Diferente de zero caso contrário
- `fprinft` : a quantidade de caracteres escritos
- `fscanf` : a quantidade de caracteres lidos
- `feof` :

  - 0 se não chegou ao final do arquivo
  - 1 se aponta para o final do arquivo

Outras funções de aquivos

- `int remove(const char *fileName)` : apaga um arquivo
- `int ferror(FILE *fp)` : verifica se a operação anterior deu erro
- `int rewind(FILE *fp)` : retorna o ponteiro para o início do arquivo

Outras funções para leitura/escrita

- `int fputc(int ch, FILE *fp)` : escreve o caracter `ch` em `fp`
- `int fputs(const char *str, FILE *fp)` : escreve a string `str` em `fp`
- `int fgetc(FILE *fp)` : le um caracter de `fp`
- `int fgets(const char *str, int x, FILE *fp)` : le `x` caracteres da string `str` do arquivo `fp`

Outras funções para leitura/escrita

- `int fread(void *p, int size, int n, FILE *fp)` : le os `n` dados de tamanho `size` do arquivo `fp` e armazena em `p`
- `int fwrite(const void *p, int size, int n, FILE *fp)` : escreve `n` dados do tamanho `size` da variável `p` em `fp`

Arquivos e streams

- Arquivo é armazenado em um dispositivo físico
- Stream é uma abstração usada em linguagens de programação para representar arquivos, permitindo que a implementação não dependa de detalhes do dispositivo, agregando o conceito de buffer.

Ou seja, abrimos streams no programa e geramos o arquivo quando fazemos a operação de fechar o arquivo.

Arquivos e streams

No ANSI C, por padrão temos 3 streams:

- `stdin` : entrada padrão, geralmente é o teclado
- `stdout` : saída padrão, normalmente sendo usado o terminal (console)
- `stderr` : saída de erro padrão, normalmente também é usado o terminal ou um arquivo

Apesar do padrão, podemos redirecionar as saídas e entrada.