Programação Imperativa e Funcional

Manipulação de Arquivos

Diego Bezerra

dfb2@cesar.school

C.E.S.A.R school

Objetivos

- Compreender a importância da manipulação de arquivos em C
- Utilizar funções da biblioteca `<stdio.h>` para ler, escrever e fechar arquivos
- Desenvolver programas que persistem dados e processam registros

Por que usar arquivos?

- Permitem armazenar grande quantidade de dados
- **Persistência de dados (armazenamento em disco)**
- Acesso concorrente aos dados (mais de um programa pode usar os dados ao mesmo tempo)
- Economia de memória RAM

Tipos de Arquivos

- A linguagem C trabalha com dois tipos de arquivos: de texto e binários
  - **Arquivo de texto**: armazena caracteres de 8 bits, como caractere ASCII
  - **Arquivo binário**: armazenam uma sequência de bits que está sujeita às conversões dos programas que geraram e são gravados na forma binária

Biblioteca e Conceitos

- A linguagem C possui uma série de funções para manipulação de arquivos, cujos protótipos estão definidos na biblioteca padrão **<stdio.h>**
  - No entanto, **suas funções se limitam a abrir/fechar e ler caracteres/bytes**
  - **É tarefa do programador criar a função que manipula os caracteres/bytes lidos**

Biblioteca e Conceitos

- Todas as funções de manipulação de arquivos trabalham com o conceito de **ponteiro de arquivo (FILE *)**
- Esse ponteiro representa a ligação com o arquivo físico

```
#include <stdio.h>

int main() {
     FILE *p;
     return 0;
}
```

Abrindo arquivos com fopen

- Para a abertura de um arquivo, podemos usar a função **fopen**
- **O primeiro parâmetro** determina qual arquivo deverá ser aberto
    - **Absoluto**: desde o diretório raíz: /home/user/dados.txt
    - **Relativo**: dados.txt
- **O segundo parâmetro** define o modo de abertura (o que pode ser feito)
- **Retorno**: ponteiro para o arquivo (ou **NULL** em caso de erro)

```
#include <stdio.h>

int main() {
     FILE *arquivo = fopen("dados.txt", "r");
     return 0;
}
```

Abrindo um arquivo

Modo	Arquivo	Função
“r”	Texto	Leitura. Arquivo deve existir.
“w”	Texto	Escrita. Cria arquivo se não houver. Apaga o anterior se ele existir.
“a”	Texto	Escrita. Os dados serão adicionados no fim do arquivo ("append").
“r+”	Texto	Leitura/Escrita. O arquivo deve existir e pode ser modificado.
“w+”	Texto	Leitura/Escrita. Cria arquivo se não houver. Apaga o anterior se ele existir.
“a+”	Texto	Leitura/Escrita. Os dados serão adicionados no fim do arquivo ("append").

Abrindo arquivos em C

- Um arquivo binário pode ser aberto para escrita utilizando o seguinte conjunto de comandos
- A condição **arquivo == NULL** verifica se o arquivo foi aberto com sucesso.

```
#include <stdio.h>

int main() {
     FILE *arquivo;
     arquivo = fopen("dados.txt", "r");
     if (arquivo == NULL) { printf("Erro na abertura do arquivo.\n)"; return 1; }
     // Comando para fechar o arquivo
     return 0;
}
```

Fechando um arquivo

- Sempre que terminamos de usar um arquivo que abrimos, devemos fechá-lo. Para isso, usa-se a função **fclose(FILE *fp)**
- O ponteiro **arquivo** passado à função **fclose** determina o arquivo a ser fechado. A função retorna zero no caso de sucesso
- **Importante**! Terminar um programa sem fechar um arquivo aberto pode provocar a perda de dados no arquivo ou corrompê-lo
- O sistema operacional pode limitar a quantidade de arquivos abertos ao mesmo tempo

Leitura e escrita de arquivos

- Uma vez aberto um arquivo, podemos ler e escrever nele
- Existem funções em C que permitem ler e escrever uma sequência de caracteres
  - **fprintf** - permite escrever dados formatados em um arquivo de texto. Se a escrita for realizada com sucesso, ela retorna a quantidade de bytes escritos. Caso contrário, retorna 0
  - **fscanf** - permite ler dados formatados de um arquivo de texto. Se a leitura for realizada com sucesso, ela retorna a quantidade de bytes lidos. Caso contrário, retorna 0. Ao ler o fim de um arquivo, ela retorna **EOF**

Leitura e escrita de arquivos

- **fscanf**(arquivo, “formatacao”, variaveis);
- **fprintf**(arquivo, “formatacao”, variaveis);

```
#include <stdio.h>

int main() {
     int i;
     FILE *arquivo;
     arquivo = fopen(“dados.txt”, “w”);
     for (i=0; i < 10; i++) {fprintf(arquivo, “%d\n”, i);}
     fclose(arquivo);
     arquivo = fopen(“dados.txt”, “r”);
     while(fscanf(arquivo, “%d”, &i) == 1) {printf(“%d\n”, i);}
     fclose(arquivo);
     return 0;
}
```

Leitura e escrita de arquivos

- Leitura e escrita de números e strings

```
#include <stdio.h>
#include <string.h>
int main() {
     int i;
     char nome[20] = “jogador”;
     FILE *arquivo;
     arquivo = fopen(“dados.txt”, “w”);
     for (i=0; i < 10; i++) {fprintf(arquivo, “%s %d\n”, nome, i);}
     fclose(arquivo);
     arquivo = fopen(“dados.txt”, “r”);
     while(fscanf(arquivo, “%19s %d”, nome, &i) == 2) {printf(“%s %d\n”, nome, i);}
     fclose(arquivo);
     return 0;
}
```

Agora é com vocês

- **Problema 1**. Escreva um programa em C que peça 3 notas de 3 alunos e salve esses dados em um arquivo chamado **notas.txt** com o nome e as notas separadas por espaço.
  - Ex.: Rafael 10.0 6.5 7.0
  - Leia o arquivo e exiba na tela o nome, as notas e a média simples
- **Problema 2**. Utilizando a biblioteca cli-lib e stdio.h, crie uma função para ler e salvar uma pontuação. O valor e nome devem ser salvos e exibidos na tela.

Referências

- Rangel Netto, J. L. M., Cerqueira, R. D. G., & Celes Filho, W. (2004). **Introdução a estrutura de dados: com técnicas de programação em C**.