Arquivos em C

Prof. Marcelo Veiga Neves
marcelo.neves@pucrs.br

Arquivos em C

- Por que são necessários arquivos?
  - Quando um programa é terminado, os dados são perdidos - salvar dados em um arquivo irá preservar esses dados
  - Se um arquivo for usado para manter dados, pode-se facilmente acessar seu conteúdo, automatizando a entrada de dados
  - É possível mover dados de um computador a outro, sem modificações
- Tipos de arquivos
  - Arquivos texto
  - Arquivos binários

Arquivos em C

- Arquivos texto
  - Arquivos texto normalmente são identificados com a extensão *.txt*, *.csv*, *.json*, entre outros
  - Pode-se criar arquivos texto usando um editor de texto simples
  - Quando abre-se um desses arquivos, é possível interpretar o conteúdo diretamente e editá-lo
  - É necessário um esforço mínimo para mantê-los, são facilmente legíveis, porém ocupam maior espaço em disco[1]
- Arquivos binários
  - Arquivos binários são identificados com a extensão *.bin*, *.exe*, *.dat*, entre outros
  - Ao invés de armazenarem dados em texto puro, armazenam em valores binários (0's e 1's)
  - Podem armazenar uma densidade maior de dados, mas requerem um formato de dados específico (como registros)

[1] Apenas caracteres que podem ser impresssos são usados, o que reduz a densidade de informação representada por espaço em disco

Operações sobre arquivos

- Em C, pode-se realizar 5 principais operações sobre arquivos:
  - Criar um novo arquivo
  - Abrir um arquivo existente
  - Fechar um arquivo
  - Ler de um arquivo
  - Escrever em um arquivo
- Trabalhando com arquivos
  - Quando deseja-se trabalhar com arquivos em C, é necessário declarar um ponteiro de um tipo específico
  - Esse ponteiro é um descritor para o arquivo

```
FILE *fptr;
```

Abrindo um arquivo - para criação e edição

- A abertura de arquivos é realizada por meio da função *fopen()*
- Exemplos:

```
fptr = fopen("newdata.txt", "w");

fptr = fopen("olddata.bin", "rb");
```

- A sintaxe para abertura de arquivos considera o nome do arquivo e o modo de operação
  - Os modos de operação podem ser uma combinação de modo texto ou binário e leitura apenas ou escrita
  - "*r*" - apenas leitura (modo texto)
  - "*rb*" - apenas leitura (modo binário)
  - "*w*", "*wb*" - apenas escrita, (recria)
  - "*a*", "*ab*" - acrescenta dados ao final do arquivo (cria)
  - "*r* + ", "*rb* + " - leitura e escrita
  - "*w* + ", "*wb* + " - leitura e escrita (recria)
  - "*a* + ", "*ab* + " - leitura e escrita (cria)

Fechando, lendo e escrevendo em arquivos

- Um arquivo deve ser fechado após operações de leitura ou escrita
- O fechamento de arquivos é realizado por meio da função *fclose()*

```
fclose(fptr);
```

- Principais funções para leitura e escrita em arquivos
  - Arquivos texto: *fprintf()*, *fscanf()*, *fgetc()*, *fputc()* e *fgets()*
  - Arquivos binários: *fread()* e *fwrite()*
- As operações de escrita são "bufferizadas". Utiliza a função *fflush()* para forçar a escrita imediata do conteúdo do *buffer* no arquivo

```
fflush(fptr);
```

Escrevendo em um arquivo texto

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int num;
    FILE *fptr;

    fptr = fopen("test.txt", "w");
    if (fptr == NULL) {
        printf("Error creating file\n");
        return -1;
    }
    printf("Enter num: ");
    scanf("%d", &num);
    fprintf(fptr, "val1: %d\n", num);
    printf("Enter another num: ");
    scanf("%d", &num);
    fprintf(fptr, "val2: %d\n", num);
    fclose(fptr);

    return 0;
}
```

Lendo de um arquivo texto

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int num1, num2;
    FILE *fptr;

    if ((fptr = fopen("test.txt", "r")) == NULL){
        printf("Error opening file\n");
        return -1;
    }
    fscanf(fptr, "%*s %d %*s %d", &num1, &num2);
    printf("value of val1 = %d, val2 = %d\n", num1, num2);
    fclose(fptr);

    return 0;
}
```

Escrevendo e lendo de um arquivo binário

- Para escrever em um arquivo binário, é necessário utilizar a função *fwrite()*
- Essa função considera quatro argumentos:
  1. Endereço dos dados a serem escritos
  2. Tamanho dos dados a serem escritos
  3. Quantidade de itens
  4. Ponteiro para o arquivo (descritor)

```
fwrite(&data, data_size, items, fptr);
```

- Para ler de um arquivo binário, é necessário utilizar a função *fread()*
- Essa função considera quatro argumentos, de forma similar à função *fwrite()*

```
fread(&data, data_size, items, fptr);
```

Escrevendo em um arquivo binário

```c
#include <stdio.h>
#include <stdlib.h>

struct threeNum {
    int n1, n2, n3;
};

int main()
{
    int n;
    struct threeNum num;
    FILE *fptr;

    if ((fptr = fopen("test.bin", "wb")) == NULL) {
        printf("Error creating file\n");
        return -1;
    }
    for (n = 0; n < 5; n++) {
        num.n1 = n; num.n2 = 5 * n; num.n3 = 5 * n + 1;
        fwrite(&num, sizeof(struct threeNum), 1, fptr);
    }
    fclose(fptr);
    return 0;
}
```

Lendo de um arquivo binário

```c
#include <stdio.h>
#include <stdlib.h>

struct threeNum {
    int n1, n2, n3;
};

int main()
{
    int n;
    struct threeNum num;
    FILE *fptr;

    if ((fptr = fopen("test.bin", "rb")) == NULL) {
        printf("Error opening file\n");
        return -1;
    }
    for (n = 0; n < 5; n++) {
        fread(&num, sizeof(struct threeNum), 1, fptr);
        printf("n1: %d\tn2: %d\tn3: %d\n",
            num.n1, num.n2, num.n3);
    }
    fclose(fptr);
    return 0;
}
```

Acesso aleatório em arquivos binários

- Se um arquivo é composto por registros e for necessário acessar um registro de uma posição específica, será necessário iterar por todos os registros anteriores
- Isso irá desperdiçar um longo tempo de operação
- Uma forma mais eficiente de realizar acesso aleatório é utilizar a função *fseek()*
- A função *fseek()* considera três argumentos
  1. Ponteiro para o arquivo (descritor)
  2. Deslocamento (posição em bytes) no arquivo
  3. Referência para o deslocamento
     - SEEK_SET - início do arquivo
     - SEEK_END - fim do arquivo
     - SEEK_CUR - deslocamento relativo

```
fseek(fptr, offset, whence);
```

Lendo registros em ordem inversa de um arquivo binário

```c
#include <stdio.h>
#include <stdlib.h>

struct threeNum {
    int n1, n2, n3;
};

int main()
{
    int n; struct threeNum num; FILE *fptr;

    if ((fptr = fopen("test.bin", "rb")) == NULL) {
        printf("Error opening file\n");
        return -1;
    }
    fseek(fptr, -sizeof(struct threeNum), SEEK_END);
    for (n = 0; n < 5; n++) {
        fread(&num, sizeof(struct threeNum), 1, fptr);
        printf("n1: %d\tn2: %d\tn3: %d\n",
            num.n1, num.n2, num.n3);
        fseek(fptr, -2 * sizeof(struct threeNum), SEEK_CUR);
    }
    fclose(fptr);
    return 0;
}
```

Referências

- Este material foi construído pelo Prof. Sérgio Johann Filho