Aula 30: Arquivos binários

Introdução a Programação

Túlio Toffolo & Puca Huachi
http://www.toffolo.com.br

BCC201 – 2019/2
Departamento de Computação – UFOP

Aulas anteriores

- Estruturas heterogêneas
- Arquivos de texto
- Alocação dinâmica (Partes 1, 2 e 3)

Aula de hoje

1. Arquivos binários
2. Usando a biblioteca `<stdio.h>`
3. Navegando em arquivos: `fseek` e `ftell`
4. Exercícios
5. Próxima aula

Aula de hoje

Arquivos binários

- Sequência de bits sujeita às convenções do programa que o gerou.
- Muitos úteis para salvar informação de forma compacta.
- Permitem, por exemplo, armazenar registros (como `structs`) em arquivos.
- Exemplos de arquivos binários:
  - arquivos executáveis,
  - arquivos compactados,
  - arquivos de registros, etc.

Arquivos de texto vs arquivos binários

Há diferentes formas de salvar dados em arquivos:

Arquivo de texto:

```
1  FILE *arquivo = fopen("arquivo.dat", "w");
2  fprintf(arquivo, "%d: %d\n", aluno.matricula, aluno.nota);
3  fclose(arquivo);
```

Arquivo binário (a informação é armazenada diretamente em ***bytes***):

```
1  FILE *arquivo = fopen("arquivo.dat", "wb");
2  fwrite(&aluno.matricula, sizeof(int), 1, arquivo);
3  fwrite(&aluno.nota, sizeof(int), 1, arquivo);
4  fclose(arquivo);
```

Aula de hoje

Biblioteca <stdio.h>

A função `fopen` é usada para abrir um arquivo e tem o seguinte protótipo:

```
1  FILE * fopen(const char *filename, const char *mode);
```

Note que a função tem 2 parâmetros:

1. `filename`: nome do arquivo a ser aberto
2. `mode`: modo de abertura (note que o `b` indica "modo" binário)
   - `"rb"` (read): **leitura**
   - `"wb"` (write): **gravação** (sobrescreve o arquivo, se existir)
   - `"rb+"` (read/update): **leitura** e **gravação** (arquivo tem que existir)
   - `"wb+"` (write/read): **leitura** e **gravação**
   - `"ab+"` (append/update): **acrescenta** dados no arquivo

Biblioteca <stdio.h>

Após abrir um arquivo, **temos que fechá-lo** com a função `fclose`.

```
1  int fclose(FILE *stream);
```

A função retorna 0 em caso de sucesso e `EOF` (-1) caso contrário.

Biblioteca <stdio.h>

Exemplos de uso de fopen e fclose:

```c
// abrindo arquivo file.dat para leitura
FILE *arquivo = fopen("file.dat", "rb");
...
fclose(arquivo);
```

```c
// abrindo arquivo file.dat para gravação
FILE *arquivo = fopen("file.dat", "wb");
...
fclose(arquivo);
```

```c
FILE *arquivo;
...
// abrindo arquivo file.dat no modo "append"
arquivo = fopen("file.dat", "ab+"); // ab+ ou a+b
...
fclose(arquivo);
```

Biblioteca <stdio.h>

Para gravar *bytes* no arquivo, usamos a função `fwrite`.

```
1  size_t fwrite(const void *ptr, size_t size, size_t count, FILE *file);
```

- A função retona o número de elementos gravados com sucesso.

Exemplo:

```
1  FILE *arquivo = fopen("texto.dat", "wb");
2
3  // escrevendo sizeof(int) bytes no arquivo
4  int n = 10;
5  fwrite(&n, sizeof(int), 1, arquivo);
6
7  fclose(arquivo);
```

Biblioteca <stdio.h>

Exemplo completo de uso de `fwrite`:

```c
#include <stdio.h>

int main()
{
    int inteiro = 10;
    char palavra[10] = "Palavra";

    // declaração e carregamento do arquivo
    FILE *arquivo = fopen("file.dat", "wb");

    // escrevendo sizeof(int) * 1 bytes no arquivo
    fwrite(&inteiro, sizeof(int), 1, arquivo);
    // escrevendo sizeof(char) * 10 bytes no arquivo
    fwrite(palavra, sizeof(char), 10, arquivo);

    // fechando (e salvando) o arquivo
    fclose(arquivo);
    return 0;
}
```

Exemplo (2) completo de uso de `fwrite` (para gravar um vetor):

```c
#include <stdio.h>

void gravaVetor(int n, int *vetor, char path[])
{
    // declaração e carregamento do arquivo
    FILE *arquivo = fopen(path, "wb");

    // escrevendo o tamanho do vetor
    fwrite(&n, sizeof(int), 1, arquivo);

    // alocando e lendo o vetor (ou seja, sizeof(int) * n bytes)
    fwrite(vetor, sizeof(int), n, arquivo);

    // fechando o arquivo
    fclose(arquivo);
}

int main()
{
    int n = 10;
    int vetor[] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
    gravaVetor(n, vetor, "vetor.dat");
    return 0;
}
```

Biblioteca <stdio.h>

Para ler *bytes* do arquivo, usamos a função `fread`:

```
1  size_t fread(void *ptr, size_t size, size_t count, FILE *file);
```

- A função retona o número de elementos lidos.

Exemplo de uso:

```
1  FILE *arquivo = fopen("file.dat", "rb");
2
3  // lendo um inteiro e um caractere
4  int inteiro;
5  char caractere;
6  fread(&inteiro, sizeof(int), 1, arquivo);
7  fread(&caractere, sizeof(char), 1, arquivo);
8
9  fclose(arquivo);
```

Biblioteca <stdio.h>

Exemplo completo de uso de `fread`:

```c
#include <stdio.h>

int main()
{
    int inteiro;
    char palavra[10];

    // declaração e carregamento do arquivo
    FILE *arquivo = fopen("file.dat", "rb");

    // lendo sizeof(int) * 1 bytes no arquivo
    fread(&inteiro, sizeof(int), 1, arquivo);
    // lendo sizeof(char) * 10 bytes no arquivo
    fread(palavra, sizeof(char), 10, arquivo);

    // imprimindo dados lidos:
    printf("%d - %s\n", inteiro, palavra);

    // fechando o arquivo
    fclose(arquivo);

    return 0;
}
```

Exemplo (2) completo de uso de `fread` (para ler um vetor):

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

int main()
{
    // declaração e carregamento do arquivo
    FILE *arquivo = fopen("vetor.dat", "rb");

    // lendo o tamanho do vetor
    int n;
    fread(&n, sizeof(int), 1, arquivo);

    // alocando e lendo o vetor (ou seja, sizeof(int) * n bytes)
    int *vetor = malloc(n * sizeof(int));
    fread(vetor, sizeof(int), n, arquivo);

    // imprimindo dados lidos:
    for (int i = 0; i < n; i++)
        printf("%d ", vetor[i]);

    // liberando memória e fechando o arquivo
    free(vetor);
    fclose(arquivo);
    return 0;
}
```

Aula de hoje

A função `fseek`

A função `fseek` reposiciona o indicador de **posição** em um arquivo.

```
1  int fseek(FILE *file, long int offset, int whence);
```

- A função retorna 0 em caso de sucesso e outro valor caso contrário.

Note que a função tem 3 parâmetros:

1. `file`: ponteiro para o arquivo considerado;
2. `offset`: quantidade de *bytes* de deslocamento (podemos utilizar números negativos);
3. `whence`: indica de onde o deslocamento é feito:
   - `SEEK_SET`: **início** do arquivo;
   - `SEEK_CUR`: posição **atual** no arquivo;
   - `SEEK_END`: **final** do arquivo.

A função `ftell`

A função `ftell` retorna a **posição** atual em um arquivo (em bytes):

```
1  long int ftell(FILE *file);
```

- A função retorna a **posição** atual no arquivo (em *bytes*).

Exemplo de uso:

```c
#include <stdio.h>

int main(int argc, char *argv[]) {
    FILE* arquivo = fopen(argv[1], "rb"); // b indica modo binário

    // move para o indicador de posição para o final do arquivo
    fseek(arquivo, 0, SEEK_END);

    // captura a posição atual (em bytes)
    long int tamanho = ftell(arquivo);

    printf("O arquivo %s tem %ld bytes\n", argv[1], tamanho);

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

`fseek` e `ftell` também podem ser utilizados em arquivos de texto:

```c
#include <stdio.h>

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

    fprintf(arquivo, "Imprimindo um texto no arquivo arquivo.txt\n");

    // movendo indicador de posição em -5 bytes
    fseek(arquivo, -5, SEEK_CUR);
    fprintf(arquivo, ".TXT");

    // movendo indicador de posição para 14 bytes a partir do início
    fseek(arquivo, 14, SEEK_SET);
    fprintf(arquivo, "TEXTO");

    // movendo indicador de posição para o final do arquivo
    fseek(arquivo, 0, SEEK_END);
    printf("Tamanho do arquivo: %ld bytes\n", ftell(arquivo));

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

Aula de hoje

Exemplo

Escreva um programa que lê n inteiros da entrada e a escreve:

- No arquivo texto `"vetor.txt"`
- No arquivo binário `"vetor.dat"`

Em seguida, compare o conteúdo (e tamanho) dos arquivos.

Exemplo de entrada: número de inteiros seguido pelos valores inteiros

```
1 8
2 1000000 2000000 3000000 4000000 5000000 6000000 7000000 8000000
```

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

int main()
{
    int n, *v;
    FILE *txt, *bin;

    // lendo vetor da entrada
    scanf("%d", &n);
    v = malloc(n * sizeof(int));
    for (int i = 0; i < n; i++)
        scanf("%d", &v[i]);

    // escrevendo vetor em arquivo texto
    txt = fopen("vetor.txt", "w");
    fprintf(txt, "%d\n", n);
    for (int i = 0; i < n; i++)
        fprintf(txt, "%d ", v[i]);   // escrevendo vetor (elemento por elemento)
    fclose(txt);

    // escrevendo vetor em arquivo binário
    bin = fopen("vetor.dat", "wb");
    fwrite(&n, sizeof(int), 1, bin);
    fwrite(v, sizeof(int), n, bin);  // escrevendo bloco de memória do vetor
    fclose(bin);

    free(v);
    return 0;
}
```

Exercícios

Exercício 1

Crie uma estrutura `Aluno` contendo matrícula (`int`), frequência (`float`) e nota (`float`). Em seguida, crie um programa que lê os dados de n alunos e escreve estes dados em um arquivo **binário**. Exemplo de entrada:

```
1 Digite o nro de alunos: 3
2
3 Digite a matricula, frequencia e nota de cada aluno:
4 0312 100.0 10.0
5 0313 100.0 9.5
6 0314 74.0  6.0
```

Exercício 2 (Opcional)

Crie um programa que lê as n estruturas gravadas (no exercício anterior) de um arquivo binário.

Aula de hoje

Próxima aula

- Dúvidas sobre o TP?
- Exercícios e revisão para prova

Perguntas?