Aula: Arquivos

Introdução a Programação

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

Departamento de Computação
Universidade Federal de Ouro Preto

Aula: Arquivos

1. Arquivos de texto
2. Biblioteca `<stdio.h>`
3. Exemplos e exercícios
4. Arquivos binários
5. Biblioteca `<stdio.h>` e arquivos binários
6. Navegando em arquivos: `fseek` e `ftell`
7. Exercícios

Aula: Arquivos

Arquivos

- Podem armazenar grande quantidade de **informação**.
- Mantém dados de forma **persistente** (gravado em disco).
- Acesso aos dados pode ser **não sequencial**.
- Acesso à informação pode ser **concorrente**.

Arquivos

Arquivo texto

- Armazena caracteres seguindo uma codificação (utf-8, por exemplo).
- Exemplo:

```
1 Este é um arquivo de texto, composto por caracteres...
2  - abc
3  - def...
```

Arquivo binário

- Sequência de bits sujeita às convenções do programa que o gerou.
- Exemplos: arquivos executáveis, compactados, de registros, etc.

Aula: Arquivos

Biblioteca <stdio.h>

C fornece o tipo FILE para representar um arquivo.
Na prática, usamos um ponteiro do tipo FILE.

Exemplo de declaração:

```
1  // arquivo para leitura
2  FILE *entrada;
3
4  // arquivo para gravação
5  FILE *saida;
```

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 do arquivo
   - `"r"` (read): **leitura**
   - `"w"` (write): **gravação** (sobrescreve o arquivo, se existir)
   - `"r+"` (read/update): **leitura** e **gravação** (arquivo tem que existir)
   - `"w+"` (write/read): **leitura** e **gravação**
   - `"a+"` (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.txt para leitura
FILE *arquivo = fopen("file.txt", "r");
...
fclose(arquivo);
```

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

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

Biblioteca <stdio.h>

Para impressão (gravar no arquivo), podemos utilizar a função `fprintf`, cujo funcionamento é muito parecido com a função `printf`.

```
1  int fprintf(FILE *stream, const char *format, ... );
```

Exemplo:

```
1  FILE *arquivo = fopen("texto.txt", "w");
2
3  // escrevendo texto e um número inteiro no arquivo
4  int n = 10;
5  fprintf(arquivo, "O valor de n = %d\n", n);
6
7  fclose(arquivo);
```

Biblioteca <stdio.h>

Exemplo completo de uso de fprintf:

```c
#include <stdio.h>

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

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

    // gravando um inteiro e uma palavra no arquivo
    fprintf(arquivo, "%s - %d\n", palavra, inteiro);

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

Biblioteca <stdio.h>

Para leitura, podemos utilizar a função `fscanf`, cujo funcionamento é muito parecido com a função `scanf`.

```
1  int fscanf(FILE *stream, const char *format, ... );
```

- A função retona o número de argumentos preenchidos ou `EOF` se o fim do arquivo for atingido.

Exemplo de uso:

```
1  FILE *arquivo = fopen("file.txt", "r");
2
3  // lendo um inteiro e um caractere separados por um espaço
4  int inteiro;
5  char caractere;
6  fscanf(arquivo, "%d %c", &inteiro, &caractere);
7
8  fclose(arquivo);
```

Biblioteca <stdio.h>

Exemplo completo de uso de `fscanf` para ler um vetor:

```c
#include <stdio.h>

int main()
{
    // declaração e carregamento do arquivo
    FILE *arquivo = fopen("file.txt", "r");

    // lendo o tamanho do vetor
    int n; // no máximo 100
    fscanf(arquivo, "%d", &n);

    // criando e lendo o vetor
    int vetor[100];
    for (int i = 0; i < n; i++)
        fscanf(arquivo, "%d", &vetor[i]);

    ...

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

Outra funções

A biblioteca <stdio.h> fornece outras funções úteis para **ler** dados de um arquivo texto:

```
// lê uma linha, incluindo o '\n' de um arquivo (lembram dela?)
char *fgets (char *str, int num, FILE *stream);

// lê um caractere e retorna (sim, retorna como um inteiro)
int fgetc(FILE *stream);

// retorna 0 se a posição atual não for o fim do arquivo
// e um valor diferente de 0 caso contrário
int feof(FILE *stream);
```

Outra funções

A biblioteca <stdio.h> também fornece outras funções úteis para **gravar** dados em um arquivo texto:

```c
// escreve uma string no arquivo
// a função retorna EOF em caso de erro
int fputs(const char *str, FILE *stream);

// escreve um caractere no arquivo (sim, como um inteiro);
// a função retorna EOF em caso de erro
int fputc(int character, FILE *stream);

// retorna 0 se a posição atual não for o fim do arquivo
// e um valor diferente de 0 caso contrário
int feof(FILE *stream);

// atualiza o arquivo (grava todo o conteúdo que ainda não foi
// gravado); retorna 0 em caso de sucesso e EOF caso contrário
int fflush(FILE *stream);
```

Aula: Arquivos

Exemplo 1

Crie um programa que conta o número de espaços em branco em um arquivo passado como argumento.

```c
int main(int argc, char **argv)
{
    int nEspacos = 0;
    char c;
    FILE *arquivo;

    arquivo = fopen(argv[1], "r"); // argv[1] é o primeiro argumento
    while (!feof(arquivo)) {
        c = fgetc(arquivo);
        if (c == ' ') nEspacos++;
    }
    fclose(arquivo);

    printf("O arquivo possui %d espaços.\n", nEspacos);
    return 0;
}
```

Exemplo 2

Crie um programa que copia um arquivo texto em outro arquivo texto removendo espaços (nomes dos arquivos são passados por argumento).

```c
int main(int argc, char **argv)
{
    char c;
    FILE *entrada, *saida;

    entrada = fopen(argv[1], "r"); // primeiro argumento
    saida = fopen(argv[2], "w");   // segundo argumento
    while (!feof(entrada)) {
        c = fgetc(entrada);
        if (c != ' ' && c != EOF)
            fputc(c, saida);
    }
    fclose(entrada);
    fclose(saida);

    return 0;
}
```

Exemplo 2 (alternativa)

Crie um programa que copia um arquivo texto em outro arquivo texto removendo espaços (nomes dos arquivos são passados por argumento).

```
1  int main(int argc, char **argv)
2  {
3      char c;
4      FILE *entrada, *saida;
5
6      entrada = fopen(argv[1], "r"); // primeiro argumento
7      saida = fopen(argv[2], "w");   // segundo argumento
8      while (fscanf(entrada, "%c", &c) != EOF) {
9          if (c != ' ')
10             fprintf(saida, "%c", c);
11     }
12     fclose(entrada);
13     fclose(saida);
14
15     return 0;
16 }
```

Exemplo 3

Crie uma função que lê uma matriz $n \times m$ de inteiros de um arquivo texto e a imprime na saída. Assuma que $n \leq 100$ e $m \leq 100$.
O arquivo tem a seguinte informação:

- Os dois primeiro números indicam as dimensões da matriz (n e m).
- Em seguida, a matriz é incluída no arquivo.
- Exemplo:

```
1  5 4
2  10 9 4 3
3  2 8 3 0
4  2 3 1 9
5  28 3 6 4
6  9 1 4 5
```

Exemplo 3

```
1  /* Esta função lê os dados de uma matriz; note que as dimensões da
2   * matriz são armazenadas nas variáveis n e m passadas por referência
3   */
4  void leMatriz(int matriz[100][100], char arquivo[], int &n, int &m)
5  {
6      FILE *entrada = fopen(arquivo, "r");
7
8      fscanf(entrada, "%d", n);
9      fscanf(entrada, "%d", m);
10
11     for (int i = 0; i < *n; i++) {
12         for (int j = 0; j < *m; j++)
13             fscanf(entrada, "%d", &matriz[i][j]);
14     }
15
16     fclose(entrada);
17 }
```

Exemplo 4

Crie uma função que escreve uma matriz de inteiros em um arquivo.

- Os dois primeiro números no arquivo indicam as dimensões da matriz.
- Em seguida, a matriz é incluída no arquivo.
- Exemplo:

```
5 4
10 9 4 3
2 8 3 0
2 3 1 9
28 3 6 4
9 1 4 5
```

- A função deve ter a seguinte assinatura:

```
void escreveMatriz(int matriz[][100], char arquivo[], int n, int m);
```

Exemplo 4

```c
/* Esta função escreve uma matriz de inteiros em um arquivo.
 */
void escreveMatriz(int matriz[][100], char arquivo[], int n, int m)
{
    FILE *saida = fopen(arquivo, "w");
    fprintf(saida, "%d %d\n", n, m);

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (j > 0)
                fprintf(arquivo, " ");
            fprintf(arquivo, "%d ", matriz[i][j]);
        }
        fprintf(arquivo, "\n");
    }

    fclose(saida);
}
```

Exercícios

Exercício

Elabore um programa que lê um arquivo de texto de, no máximo, 100 linhas (e 100 colunas) e cria um arquivo com as linhas em ordem inversa.

Dica: utilize um vetor de strings (`char[100][100]`) para armazenar as linhas, e use a função `fgets` para ler uma linha completa do arquivo.

Aula: Arquivos

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.txt", "w");
2
3  fprintf(arquivo, "%d: %d\n", aluno.matricula, aluno.nota);
4
5  fclose(arquivo);
```

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

```
1  FILE *arquivo = fopen("arquivo.dat", "wb");
2
3  fwrite(&aluno.matricula, sizeof(int), 1, arquivo);
4  fwrite(&aluno.nota, sizeof(int), 1, arquivo);
5
6  fclose(arquivo);
```

Aula: Arquivos

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);
```

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 retorna 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;
}
```

Aula: Arquivos

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: Arquivos

Exemplo

Escreva um programa que lê n inteiros ($n \leq 100$) 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

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

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

int main()
{
    int n, v[100];
    FILE *txt, *bin;

    // lendo vetor da entrada
    scanf("%d", &n);
    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);

    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.

Perguntas?