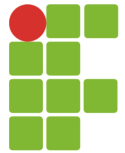
Arquivos Binários

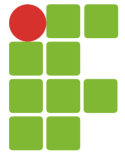
Programação de Computadores 1

Prof. Daniel Saad Nogueira Nunes

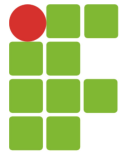
IFB – Instituto Federal de Brasília,
Campus Taguatinga



Sumário

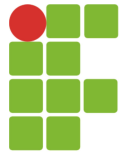


Sumário



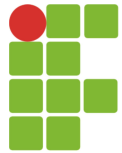
Introdução

- Em arquivos textos, os valores são expressos através de caracteres. Por exemplo, o inteiro 12345 é expresso através de 5 caracteres, necessitando portanto de 5 bytes.
- Em arquivos binários, não estamos preocupados em representar os valores de uma forma legível para qualquer pessoa. Como um inteiro normalmente ocupa 4 bytes, podemos armazenar qualquer inteiro utilizando um padrão de 4 bytes.



Introdução

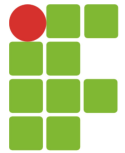
- O único problema é que precisamos saber que ali existe um inteiro a ser lido. Aplicações que lidam com arquivos binários precisam conhecer a estrutura do arquivo.
- Assim, arquivos binários fornecem um mecanismo uniforme para **reduzir o tamanho** do arquivo e **armazenar estruturas complexas** mantendo um **acesso simples**, visto que, se conhecermos o padrão dos bytes a serem lidos, conseguimos decodificar a estrutura.
- Ideal quando se quer maximizar o **desempenho** em troca de legibilidade.



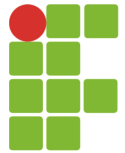
Introdução

Exemplos de arquivos binários

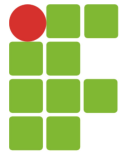
- Imagens, como arquivos `.bmp`, `.jpg` ou `.png`.
- Textos digitais em formato `.pdf`.
- Executáveis.
- Arquivos comprimidos `.zip`, `.7z` ou `.tar.gz`.



Sumário

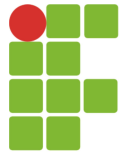


Sumário



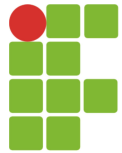
Abertura

- Assim como em arquivos texto, arquivos binários podem ser abertos com o `fopen` e fechados com o `fclose`.
- Os arquivos continuam sendo identificados pelo seu caminho, mas o modo de abertura muda. Agora temos que colocar o sufixo **b** para indicar que queremos abrir um arquivo binário.



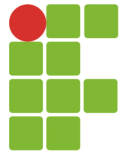
Modos de abertura

Modo	Permissão	Indicador de posição
rb	leitura	início do arquivo
rb+ ou r+b	leitura e atualização	início do arquivo
wb	escrita	início do arquivo
wb+ ou w+b	escrita e atualização	início do arquivo
ab	escrita	final do arquivo
ab+ ou a+b	escrita e atualização	final do arquivo



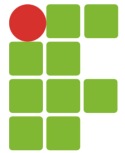
Modos de abertura

- `rb`: somente leitura. O arquivo precisa existir, caso contrário `fopen` retornará **NULL**.
- `wb`: somente escrita. Se o arquivo não existir, ele é criado. Se o arquivo existir, ele é completamente sobrescrito.
- `rb+`: leitura e atualização. O arquivo precisa existir, caso contrário `fopen` retornará **NULL**. Também é possível realizar operações de escrita no arquivo.
- `wb+`: escrita e atualização. Se o arquivo não existir, ele é criado. Se o arquivo existir, ele é completamente sobrescrito. Permite operações de leitura.

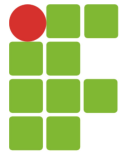


Modos de abertura

- `ab`: somente escrita. Se o arquivo existir, ele não é sobrescrito. Se ele não existir, ele é criado. Qualquer operação de escrita é feita a partir do **final** do arquivo.
- `ab+`: escrita e atualização. Se o arquivo existir, ele não é sobrescrito. Se ele não existir, ele é criado. Qualquer operação de escrita é feita a partir do **final** do arquivo.

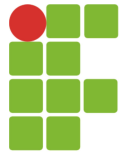


Sumário



Leitura

- Diferentemente de um arquivo texto, não utilizamos mecanismos de leitura formatada como o `fscanf`.
- Devemos utilizar um mecanismo que lê um padrão de bytes e armazena o valor em uma variável.
- O `fread` é o mecanismo padrão para leitura de arquivos binários.

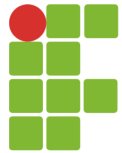


Leitura: fread

fread

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

- `ptr`: o ponteiro para a área de memória em que se quer armazenar os bytes lidos.
- `size`: o tamanho de cada elemento em bytes.
- `count`: a quantidade de elementos a serem lidos.
- `fp`: o ponteiro para o arquivo.
- Retorno: o número de bytes lidos ou 0 caso fim de arquivo ou falha na leitura.

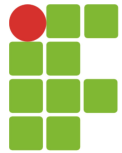


Exemplo: fread

- ```
  // lê um inteiro para a variável x
  int x;
  fread(&x, sizeof(int), 1, fp);
  ```

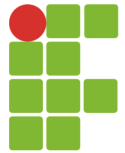
- ```
  // lê um vetor de inteiros 'v' de tamanho 100
  int v[100];
  fread(v, sizeof(int), 100, fp);
  ```

- ```
  // lê um vetor de pessoas 'p' de tamanho 100
  pessoa p[100];
  fread(p, sizeof(pessoa), 100, fp);
  ```
  ```



Exemplo: fread

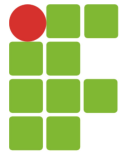
```c
// lê um arquivo binário de inteiros até o final
while (fread(&x, sizeof(int), 1, fp) != 0) {
    //...
}
```



Exemplo: fread

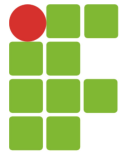
Problema

Leia um arquivo binário em que o número de inteiros armazenados, n está descrito no início do arquivo e em seguida há a presença de n inteiros.



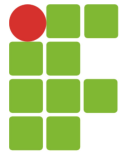
Exemplo: fread

- O caminho do arquivo binário será capturado através da linha de comando.
- Como não sabemos a quantidade exata de inteiros armazenada, primeiramente leremos o valor n e em seguida utilizaremos alocação dinâmica de memória para criar o vetor do tamanho que precisamos.



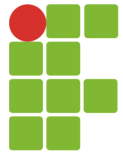
Exemplo: fread

```
void testa_parametros(int argc) {
    if (argc != 2) {
        printf("Uso: ./executavel <arquivo>");
        exit(0);
    }
}
```



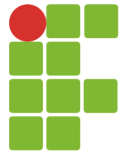
Exemplo: fread

```
void testa_abertura(FILE *fp, const char *path, const char *mode) {
    if (fp == NULL) {
        printf("Falha em abrir o arquivo %s com o modo %s\n", path, mode);
        exit(0);
    }
}
```



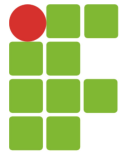
Exemplo: fread

```c
int le_tamanho_vetor(FILE *fp) {
    int n;
    fread(&n, sizeof(int), 1, fp);
    return n;
}
```



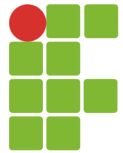
Exemplo: fread

```
int *le_vetor(FILE *fp, int n) {
    int *v = malloc(sizeof(int) * n);
    fread(v, sizeof(int), n, fp);
    return v;
}
```



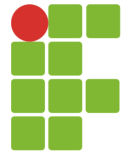
Exemplo: fread

```
void imprime_vetor(int *v, int n) {
    for (int i = 0; i < n; i++) {
        printf("%d ", v[i]);
    }
    printf("\n");
}
```



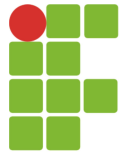
Exemplo: fread

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

void testa_parametros(int argc) {
    if (argc != 2) {
        printf("Uso: ./executavel <arquivo>");
        exit(0);
    }
}

void testa_abertura(FILE *fp, const char *path, const char *mode) {
    if (fp == NULL) {
        printf("Falha em abrir o arquivo %s com o modo %s\n", path, mode);
        exit(0);
    }
}

int le_tamanho_vetor(FILE *fp) {
    int n;
    fread(&n, sizeof(int), 1, fp);
    return n;
}
```

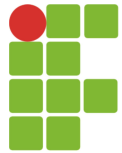


Exemplo: fread

```
23
24  int *le_vetor(FILE *fp, int n) {
25      int *v = malloc(sizeof(int) * n);
26      fread(v, sizeof(int), n, fp);
27      return v;
28  }
29
30  void imprime_vetor(int *v, int n) {
31      for (int i = 0; i < n; i++) {
32          printf("%d ", v[i]);
33      }
34      printf("\n");
35  }
36
37  int main(int argc, char *argv[]) {
38      testa_parametros(argc);
39      FILE *fp = fopen(argv[1], "rb");
40      testa_abertura(fp, argv[1], "rb");
41      int n = le_tamanho_vetor(fp);
42      int *v = le_vetor(fp, n);
43      imprime_vetor(v, n);
44      free(v);
45      return 0;
46  }
```

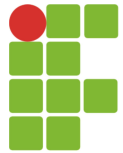


Sumário



Escrita

- Analogamente, utilizaremos a função `fwrite` para armazenar os bytes desejados em um arquivo binário.
- Ela se parece muito com a função `fread` do ponto de vista sintático, recebendo os mesmos argumentos.

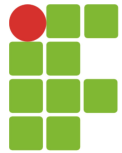


Escrita: fwrite

fwrite

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

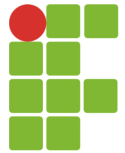
- `ptr`: o ponteiro para a área de memória que contém os bytes a serem escritos.
- `size`: o tamanho de cada elemento em bytes.
- `count`: a quantidade de elementos a serem lidos.
- `fp`: o ponteiro para o arquivo.
- Retorno: o número de bytes escritos ou 0 em caso de falha.



Exemplo: fwrite

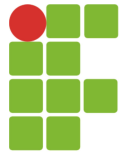
Problema

Escrever um programa que escreve um vetor de tamanho 5 com os valores $1, 2, 3, 4, 5$ em um arquivo binário.



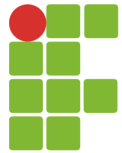
Exemplo: fwrite

- O caminho do arquivo binário de saída será capturado através da linha de comando.



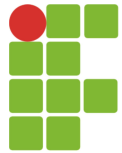
Exemplo: fwrite

```
void testa_parametros(int argc) {
    if (argc != 2) {
        printf("Uso: ./executavel <arquivo>");
        exit(0);
    }
}
```



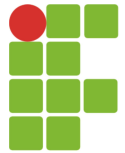
Exemplo: fwrite

```
void testa_abertura(FILE *fp, const char *path, const char *mode) {
    if (fp == NULL) {
        printf("Falha em abrir o arquivo %s com o modo %s\n", path, mode);
        exit(0);
    }
}
```



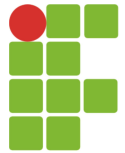
Exemplo: fwrite

```
void escreve_vetor(FILE *fp, int *v, int n) {
    fwrite(&n, sizeof(int), 1, fp);
    fwrite(v, sizeof(int), n, fp);
}
```



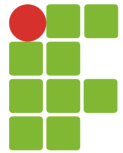
Exemplo: fwrite

```c
int main(int argc, char *argv[]) {
    testa_parametros(argc);
    FILE *fp = fopen(argv[1], "wb");
    testa_abertura(fp, argv[1], "wb");
    int v[] = {1, 2, 3, 4, 5};
    escreve_vetor(fp, v, 5);
    return 0;
}
```



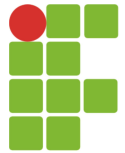
Exemplo: fwrite

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

void testa_parametros(int argc) {
    if (argc != 2) {
        printf("Uso: ./executavel <arquivo>");
        exit(0);
    }
}

void testa_abertura(FILE *fp, const char *path, const char *mode) {
    if (fp == NULL) {
        printf("Falha em abrir o arquivo %s com o modo %s\n", path, mode);
        exit(0);
    }
}

void escreve_vetor(FILE *fp, int *v, int n) {
    fwrite(&n, sizeof(int), 1, fp);
    fwrite(v, sizeof(int), n, fp);
}

```

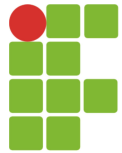


Exemplo: fwrite

```
23  int main(int argc, char *argv[]) {
24      testa_parametros(argc);
25      FILE *fp = fopen(argv[1], "wb");
26      testa_abertura(fp, argv[1], "wb");
27      int v[] = {1, 2, 3, 4, 5};
28      escreve_vetor(fp, v, 5);
29      return 0;
30  }
```

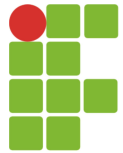


Sumário



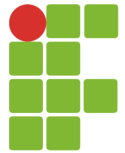
Acesso aleatório

- Com arquivos binários é bem simples buscar um dado em específico.
- Tome a aplicação anterior, em que tínhamos um vários inteiros escritos no arquivo.
- Se quiséssemos acessar o i-ésimo inteiro, bastaria avançar o indicador de posição até o byte `sizeof(int)*(n+1);`
- Lembre-se que o primeiro inteiro corresponde ao número de inteiros que estão gravados em seguida!
- Para posicionar o indicador de posição no lugar correto, faríamos: `fseek(fp,sizeof(int)*(n+1),SEEK_SET);`.
- Em seguida, faríamos um `fread` para ler o inteiro desejado.

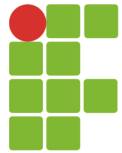


Acesso aleatório

```
18  int obtem_inteiro(FILE *fp, int id) {
19      int x;
20      fseek(fp, sizeof(int) * (id + 1), SEEK_SET);
21      fread(&x, sizeof(int), 1, fp);
22      return x;
23  }
```



Sumário



Exemplo: registro de pessoas

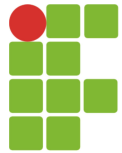
Problema

Suponha um arquivo binário contendo os seguintes dados de pessoas:

- Nome (31 caracteres).
- CPF (15 caracteres).
- Idade (inteiro).

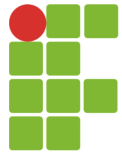
Criar funções que:

- Imprime as informações pessoais de todas as pessoas do arquivo.
- Altera a idade de uma pessoa no arquivo dado um CPF.



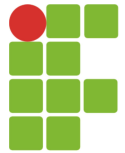
Exemplo: registro de pessoas

```
typedef struct pessoa {
    char nome[31];
    char cpf[15];
    int idade;
} pessoa;
```



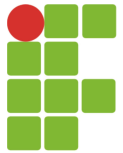
Exemplo: registro de pessoas

```
void imprime_pessoa(const pessoa *p) {
    printf("Nome: %s\n", p->nome);
    printf("CPF: %s\n", p->cpf);
    printf("Idade: %d\n\n", p->idade);
}
```



Exemplo: registro de pessoas

```
void imprime_arquivo(FILE *fp) {
    pessoa p;
    while (!fread(&p, sizeof(pessoa), 1, fp)) {
        imprime_pessoa(&p);
    }
}
```



Exemplo: registro de pessoas

```
void altera_idade(FILE *fp, const char *cpf, int nova_idade) {
    pessoa p;
    fseek(fp, 0, SEEK_SET);
    while (!fread(&p, sizeof(pessoa), 1, fp)) {
        if (strcmp(p.cpf, cpf) == 0) {
            p.idade = nova_idade;
            fseek(fp, -sizeof(pessoa), SEEK_CUR);
            fwrite(&p, sizeof(pessoa), 1, fp);
            break;
        }
    }
}
```