# 1001350: Escrita e leitura de dados em arquivos binários

Jander Moreira*

## Sumário



# 1 Apresentação

Este estudo corresponde a uma prática de programação em C com arquivos. O problema abordado é simples: armazenar 10 valores inteiros em um arquivo (seção 2.2) e recuperação posterior desses valores (seção 2.3)

## 1.1 Códigos fonte

Todos os programas citados neste texto têm seu código fonte disponibilizado.

# 2 Transferência de dados

Esta seção mostra, usando programas como exemplos, o armazenamento e recuperação de dados em arquivos.

## 2.1 A representação de valores `int`

Antes de começar, é preciso lembrar que os valores têm representação específica em memória.

Aqui se relembra o que é um inteiro em C. Em particular, será usado do tipo `int`, visto que há muitos "inteiros" diferentes disponíveis.

*Jander Moreira – Universidade Federal de São Carlos – Departamento de Computação – *Rodovia Washington Luis, km 235 – 13565-905 - São Carlos/SP – Brasil* – jander@dc.ufscar.br

O programa `info_int.c` exemplifica, para vários valores inteiros (aleatórios), a sua representação byte a byte. A execução apresenta cada byte em hexadecimal[1] (00 a FF) e em decimal (de 0 a 255).

O objetivo deste programa é apenas mostrar que um determinado valor decimal inteiro é representado, internamente na memória, por um conjunto específico de bytes. A interpretação dos bytes não é relevante no contexto deste texto.

Execute-o para ver os resultados.

O entendimento do código fonte é irrelevante. Há também programas para outros tipos de dados.

O entendimento de que existe uma representação específica para um tipo de dados é essencial!

## 2.2 Criando um arquivo de inteiros

Este exemplo é para criar um arquivo que contém inteiros. O princípio é criar um arquivo novo (em todas as execuções!) e gravar 10 valores inteiros lidos do teclado. Escritos os valores, o arquivo é fechado.

O arquivo é criado com o comando `fopen`, passando o nome do arquivo (variável `nome_arquivo`) e usando o modo `"wb"` (Código 1). Nesse modo, se o arquivo não existe ele é criado e, se existir, é truncado para tamanho zero. O *arquivo lógico* é representado pela variável `arquivo`, que é um ponteiro para `FILE`.

**Código 1**

```
FILE *arquivo = fopen(nome_arquivo, "wb");
```

A verificação de erro que se sucede é indicada para o caso de não conseguir criar o arquivo (por exemplo, se não tiver autorização para criar um arquivo em um dado diretório). Veja o Código 2.

**Código 2**

```
if(arquivo == NULL){
    perror(nome_arquivo);
    condicao_termino = 1; // termina com erro
}
else ...
```

Em caso de sucesso na criação do arquivo, um `for` é usado para ler 10 valores (quantidade arbitrária). Depois de cada leitura, seu valor é copiado para o arquivo. Essa repetição está no Código 3.

**Código 3**

```
for(int i = 0; i < 10; i++){
    // leitura
    int valor;
    printf("valor: ");
    scanf("%d", &valor);

    // gravacao
    fwrite(&valor, sizeof valor, 1, arquivo);
}
```

No Código 3, a leitura é feita tradicionalmente do teclado por meio da função `scanf`.

Aqui é importante lembrar que a função `scanf` interpreta os caracteres digitados pelo usuário (um número decimal, `"%d"`) e o converte para sua representação binária, que é armazenada no

[1] Base 16.

espaço reservado à variável `valor`. (O programa da seção 2.1 pode ser usado para ver quais bytes cada valor usa.)

A Tabela 1 mostra os bytes usados para representar 10 valores inteiros escolhidos para exemplo.

Tabela 1: Dez valores inteiros e suas representações em bytes. Esses bytes foram obtidos pela execução do programa da seção 2.1 em um sistema Linux de 64bits.

| **Valor** | **Bytes** (em hexadecimal) |
|---|---|
| 125 | `7D 00 00 00` |
| 328 | `48 01 00 00` |
| -1000 | `18 FC FF FF` |
| 0 | `00 00 00 00` |
| 346887 | `07 4B 05 00` |
| -1 | `FF FF FF FF` |
| 7 | `07 00 00 00` |
| -2000000 | `80 7B E1 FF` |
| 8388608 | `00 00 80 00` |
| 1073741968 | `90 00 00 40` |

A transferência dos bytes da variável `valor` para o arquivo é feita pela função `fwrite` já mostrada no Código 3 e posta em destaque no Código 4.

**Código 4**

```
fwrite(&valor, sizeof valor, 1, arquivo);
```

Essa instrução indica a cópia dos quatro bytes que compõe a variável `valor` para o arquivo aberto do descritor `arquivo`.

Em mais detalhes, função do `fwrite` é copiar os dados da memória principal para o arquivo. Para isso, ela utiliza como fonte de dados os bytes indicados pelo endereço `&valor` e transfere a quantidade de bytes igual a `sizeof valor`. Neste caso, são quatro bytes que é o tamanho de um `int`. O número de ocorrências, ou seja, a quantidade de inteiros que é copiada é apenas um; isso porque a variável valor é apenas um único `int`. Finalmente, o destino dos bytes é `arquivo`, ou seja, o fluxo de dados correspondente ao *arquivo físico* aberto pelo `fopen`.

A função `fwrite` é executada 10 vezes pelo programa, copiando o conteúdo de `valor` para o arquivo a cada chamada. Assim, são copiados 40 bytes para o arquivo no total.

Depois da repetição há a necessidade de encerrar o acesso ao aquivo antes do programa terminar. A função `fclose`, indicada no Código 5, é usada para esse fim. O arquivo fechado é, naturalmente, o mantido na variável `arquivo`.

**Código 5**

```
fclose(arquivo);
```

O programa completo está no Código 10 apresentado na íntegra na seção 3.

### Exemplo de execução

Uma execução do programa de criação de um arquivo com 10 inteiros foi feita e, como entrada, foram usados os valores da Tabela 1.

Como resultado, foi criado um arquivo `int.dados` com tamanho total de 40 bytes. A Figura 1 mostra o conteúdo do arquivo.

Figura 1: Conteúdo do arquivo `int.dados` criado a partir da execução do Código 10. Nas colunas centrais são apresentados os bytes que compõem o arquivo, em hexadecimal.

```
7d 00 00 00 48 01 00 00  18 fc ff ff 00 00 00 00
07 4b 05 00 ff ff ff ff  07 00 00 00 80 7b e1 ff
00 00 80 00 90 00 00 40
```

Uma análise cuidadosa permite verificar que os bytes gravados no arquivo correspondem aos apresentados na Tabela 1. Ou seja, cada quatro bytes do arquivo representam um dos valores inteiros digitados como entrada para o programa.

#### Conclusão

A conclusão desta seção é que os bytes que formam cada inteiro digitado como entrada para o programa `cria_inteiro.c` foram copiados para o arquivo `int.dados`.

## 2.3 Leitura dos dados do arquivo de inteiros

Salvo na ocorrência de uma pane com o dispositivo no qual foi gravado o arquivo `int.dados`, ou então na situação dele ser deliberadamente apagado, os dados dos 10 inteiros estão armazenados de forma permanente.

Agora será apresentado o código que pode ser usado para recuperar os valores lidos.

Para se ter acesso ao *arquivo físico* chamado `int.dados`, é preciso abri-lo. A função `fopen` é usada para esse fim (Código 6). O modo de abertura é `"rb"`: o `"r"` indica que o arquivo será aberto para leitura, o que, naturalmente, preserva todo seu conteúdo. A especificação do `"b"` (binário) é importante no sistema operacional Windows.

**Código 6**

```
FILE *arquivo = fopen(nome_arquivo, "rb");
if(arquivo == NULL){
    perror(nome_arquivo);
    condicao_termino = 1; // termina com erro
}
else ...
```

A verificação da condição de abertura é feita pela comparação com `NULL`, que indica a falha no acesso. Problemas de acesso incluem, por exemplo, a inexistência do arquivo ou a falta de permissão de leitura.

A leitura do arquivo é feita pela função `fread`, no formato do Código 7.

**Código 7**

```
int valor;
fread(&valor, sizeof valor, 1, arquivo);
```

Os parâmetros do `fread` são idênticos aos do `fwrite`. Apenas o sentido da transferência dos dados é invertido.

O comando do Código 7 usa o endereço da variável `valor` como destino (`&valor`, portanto) e faz a cópia de exatos `sizeof valor` bytes. A quantidade de elemetos copiados é um, visto que o destino é um único `int`. No parâmetro final, `arquivo` indica que a origem dos bytes é o arquivo aberto.

Figura 2: Resultado da execução do programa do Código 11 com o arquivo criado com os dados da Tabela 1.

```
valor: 125
valor: 328
valor: -1000
valor: 0
valor: 346887
valor: -1
valor: 7
valor: -2000000
valor: 8388608
valor: 1073741968
```

Dentro do contexto da leitura dos 10 valores inteiros gravados, no Código 8 está a repetição completa.

**Código 8**

```
for(int i = 0; i < 10; i++){
    // leitura
    int valor;
    fread(&valor, sizeof valor, 1, arquivo);

    // escrita na tela
    printf("valor: %d\n", valor);
}
```

Depois das leituras, o arquivo é fechado com o `fclose` (Código 9).

**Código 9**

```
fclose(arquivo);
```

O código completo está na seção 3, no Código 11.

### Exemplo de execução

A execução do programa de leitura recupera, de quatro em quatro bytes, os dados que estão no arquivo.

Como exemplo, na primeira leitura, os quatro primeiro bytes (`7D 00 00 00`) são transferidos para a variável `valor`. Internamente, o programa interpreta essa combinação de bits como sendo o valor 125, que é o que o comando `printf` põe na tela logo em seguida à leitura. O mesmo ocorre para os demais bytes, escrevendo os inteiros armazenados corretamente.

### Conclusão

A leitura a partir de um arquivo é a transferência de dados do arquivo para a memória principal. Uma vez que os bytes vão para uma variável com um tipo específico, a interpretação do valor é a mesma de quando houve a gravação.

# 3 Códigos fonte

O código para criar o arquivo de inteiros que foi descrito na seção 2.2 está no Código 10.

**Código 10**

```
/*
  Criacao de um arquivo com valores inteiros
  em sua representacao binaria
  Input: uma sequencia de valores inteiros
    (teclado)
  Output: as representacoes binarias dos
    valores digitados em um arquivo com
    nome int.dados

  Jander, 2020

  Principio:
    - Criar um arquivo vazio
    - Fazer uma repeticao para ler 10
        valores inteiros e escrever
        cada um deles no arquivo
    - Encerrar o acesso ao arquivo
*/
#include <stdio.h>
#include <stdlib.h>
#include <error.h>

int main(void){
    // condicao de termino (0 = sem erros)
    int condicao_termino = 0;

    // nome do arquivo
    char *nome_arquivo = "int.dados";

    // criacao do arquivo
    // "wb" = cria ou zera arquivo; formato binario
    // "b" = necessario no Windows
    FILE *arquivo = fopen(nome_arquivo, "wb");
    if(arquivo == NULL){
        perror(nome_arquivo);
        condicao_termino = 1; // termina com erro
    }
    else{
        // repeticao de leitura do teclado
        // e gravacao no arquivo
        for(int i = 0; i < 10; i++){
            // leitura
            int valor;
            printf("valor: ");
            scanf("%d", &valor);

            // gravacao
            fwrite(&valor, sizeof valor, 1, arquivo);
        }

        // encerramento do acesso
        fclose(arquivo);

        condicao_termino = 0; // termina sucesso
    }
```

▷

```
    return condicao_termino;
}
```

Arquivo: codigo/cria_inteiro.c.

No Código 11 está a listagem completa do programa da seção 2.3, que faz a leitura dos 10 inteiros armazenados no arquivo.

**Código 11**

```
/*
  Recuperacao dos dados inteiros armazenados em
  um arquivo com representacao binaria
  Input: os bytes formando representacao de valores
    inteiros a partir do arquivo int.dados
  Output: a apresentacao dos valores recuperados
    em decimal (tela)

  Jander, 2020

  Principio:
    - Abrir um arquivo com pelo menos 10
        inteiros armazenados
    - Fazer uma repeticao para ler 10
        valores inteiros do arquivo
        e escrever cada um deles na tela
    - Encerrar o acesso ao arquivo
*/
#include <stdio.h>
#include <stdlib.h>
#include <error.h>

int main(void){
    // condicao de termino (0 = sem erros)
    int condicao_termino = 0;

    // nome do arquivo
    char *nome_arquivo = "int.dados";

    // abertura do arquivo
    // "rb" = abre o arquivo; formato binario
    // "b" = necessario no Windows
    FILE *arquivo = fopen(nome_arquivo, "rb");
    if(arquivo == NULL){
        perror(nome_arquivo);
        condicao_termino = 1; // termina com erro
    }
    else{
        // repeticao de leitura do arquivo
        // e apresentacao na tela
        for(int i = 0; i < 10; i++){
            // leitura
            int valor;
            fread(&valor, sizeof valor, 1, arquivo);

            // escrita na tela
            printf("valor: %d\n", valor);
        }
```

▷

```
        // encerramento do acesso
        fclose(arquivo);

        condicao_termino = 0; // termina sucesso
    }

    return condicao_termino;
}
```

Arquivo: codigo/recupera_inteiro.c.