

AULAS 26 - ARQUIVOS BINÁRIOS*

Deľano Beder

*Baseado no livro: Linguagem C Completa e Descomplicada, de André Backes

MODOS DE ABERTURA DE UM ARQUIVO BINÁRIO

Modo	Arquivo	Função
"rb"	Binário	Leitura. Arquivo deve existir.
"wb"	Binário	Escrita. Cria arquivo se não houver. Apaga o conteúdo anterior se ele existir.
"ab"	Binário	Escrita. Os dados serão adicionados no fim do arquivo ("append").
"r+b"	Binário	Leitura/Escrita. O arquivo deve existir e pode ser modificado.
"w+b"	Binário	Leitura/Escrita. Cria arquivo se não houver. Apaga o conteúdo anterior se ele existir.
"a+b"	Binário	Leitura/Escrita. Os dados serão adicionados no fim do arquivo ("append").

ESCRITA DE BLOCO DE DADOS

- A função **fwrite** é responsável pela escrita de um bloco de dados da memória para um arquivo
- Seu protótipo é:

```
unsigned fwrite(void *buffer, int numero_de_bytes,  
               int count, FILE *fp);
```

ESCRITA DE BLOCO DE DADOS

```
unsigned fwrite(void *buffer, int numero_de_bytes,  
               int count, FILE *fp);
```

- A função **fwrite** recebe 4 argumentos
 - **buffer:** ponteiro para a região de memória na qual estão os dados que serão copiados;
 - **numero_de_bytes:** tamanho de cada posição de memória a ser copiada;
 - **count:** total de unidades de memória que devem ser copiadas;
 - **fp:** ponteiro associado ao arquivo onde os dados serão escritos.

ESCRITA DE BLOCO DE DADOS

- Note que temos dois valores numéricos
 - `numero_de_bytes`
 - `count`
- Isto significa que o número total de bytes escritos é:
 - `numero_de_bytes * count`
- Retorna o número de unidades efetivamente escritas.
 - Este número pode ser menor que `count` quando ocorrer algum erro.

ESCRITA DE BLOCO DE DADOS

- Exemplo
da
função
fwrite

```
int main(){
    FILE * arq;
    int v[]={1,2,3,4,5};
    char str[]="ola mundo!";
    float x1=1.5;
    arq = fopen ("teste.bin","wb");
    //escreve o conteúdo do vetor v no arquivo
    fwrite(v,sizeof(int),5,arq);
    //escreve o conteúdo da string str no arquivo
    fwrite(str,sizeof(char),strlen(str),arq);
    //escreve o conteúdo do float x1 no arquivo
    fwrite(&x1,sizeof(float),1,arq);
    //escreve o conteúdo das duas primeiras posições de v em arq
    fwrite(v,sizeof(int),2,arq);
    fclose(arq);
    return 0;
}
```

LEITURA DE BLOCO DE DADOS

- A função **fread** lê e transfere um bloco de dados de um arquivo para a memória
- Seu protótipo é:

```
unsigned fread(void *buffer, int numero_de_bytes,  
               int count, FILE *fp);
```

LEITURA DE BLOCO DE DADOS

- A função **fread** funciona como a função **fwrite**, porém lendo dados do arquivo.
- Retorna o número de itens escritos. Este valor será igual a **count** a menos que ocorra algum erro.

```
unsigned fread(void *buffer, int numero_de_bytes,  
               int count, FILE *fp);
```


LEITURA DE BLOCO DE DADOS

- Exemplo da
função **fread**

```
int main(){
    FILE * arq;
    int i, v1[5],v2[2];
    char str[11];
    float x;
    arq = fopen ("teste.bin","rb");
    fread(v1,sizeof(int),5,arq);
    fread(str,sizeof(char),10,arq);
    str[10]='\0';
    fread(&x,sizeof(float),1,arq);
    fread(v2,sizeof(int),2,arq);
    for (i=0;i<5;i++){
        printf("%d ", v1[i]);
    }
    printf("\n%s\n",str);
    printf("%.2f\n",x);
    for (i=0;i<2;i++){
        printf("%d ", v2[i]);
    }
    fclose(arq);
    return 0;
}
```

ESCRITA/LEITURA DE BLOCO DE DADOS

- Quando o arquivo for aberto para dados binários, **fwrite** e **fread** podem manipular **qualquer tipo de dado**.
 - int
 - float
 - double
 - array
 - struct
 - etc.

EXERCÍCIO

Defina uma função que leia um vetor de inteiros de um arquivo texto. O primeiro inteiro lido é o tamanho do vetor. O tamanho do vetor deve ser alocado dinamicamente. A função deve retornar o vetor lido (ponteiro para int).

int *le_vetor_texto(int *n)

Defina uma função que escreva um vetor (e seu tamanho) em um arquivo de saída binário. O primeiro inteiro escrito é o tamanho do vetor.

void escreve_vetor_binario(int *v, int n)

EXERCÍCIO

Defina uma função que leia um vetor de inteiros de um arquivo binário. O primeiro inteiro lido é o tamanho do vetor. O tamanho do vetor deve ser alocado dinamicamente. A função deve retornar o vetor lido.

int *le_vetor_binario(int *n)

Defina uma função que escreva um vetor de inteiros (e seu tamanho) em um arquivo texto. O primeiro inteiro escrito é o tamanho do vetor.

void escreve_vetor_texto(int *v, int n)

FUNÇÃO FTELL()

- Retorna o valor atual do indicador de posição de um arquivo

```
long int ftell(FILE *stream)
```

- Retorna:
 - Em caso de sucesso, é retornado o valor atual do indicador de posição.
 - Em caso de erro, é retornado -1.

FUNÇÃO FTELL()

```
int main() {
    int i, n;
    double valor;
    FILE *f = fopen_e_teste("doubles.dat", "wb");
    printf("N: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        printf("(%do) ", (i + 1));
        scanf("%lf", &valor);
        fwrite(&valor, sizeof(double), 1, f);
    }
    fclose(f);

    return 0;
}
```

N: 5
(1o) 10.23
(2o) 34.78
(3o) 45.50
(4o) 67.80
(5o) 78.90

```
(01) [00 - 08] : 10.23
(02) [08 - 16] : 34.78
(03) [16 - 24] : 45.50
(04) [24 - 32] : 67.80
(05) [32 - 40] : 78.90
```

```
int main() {
    double valor;
    long int ini = 0;
    long int fim;
    FILE *f = fopen_e_teste("doubles.dat", "rb");

    while (fread(&valor, sizeof(valor), 1, f) > 0) {
        fim = ftell(f);
        printf("(%02ld) ", fim/sizeof(double));
        printf("[%02ld - %02ld] : ", ini, fim);
        printf("%.2lf\n", valor);
        ini = fim;
    }

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

FUNÇÃO FTELL() / FEOF()

```
int main() {
    int i, n;
    double valor;
    FILE *f = fopen_e_teste("doubles.dat", "wb");
    printf("N: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        printf("(%do) ", (i + 1));
        scanf("%lf", &valor);
        fwrite(&valor, sizeof(double), 1, f);
    }
    fclose(f);

    return 0;
}
```

N: 5
(1o) 10.23
(2o) 34.78
(3o) 45.50
(4o) 67.80
(5o) 78.90

```
(01) [00 - 08] : 10.23
(02) [08 - 16] : 34.78
(03) [16 - 24] : 45.50
(04) [24 - 32] : 67.80
(05) [32 - 40] : 78.90
```

```
int main() {
    double valor;
    long int ini = 0;
    long int fim;
    FILE *f = fopen_e_teste("doubles.dat", "rb");
    fread(&valor, sizeof(valor), 1, f);
    while (!feof(f)) {
        fim = ftell(f);
        printf("(%02ld) ", fim/sizeof(double));
        printf("[%02ld - %02ld] : ", ini, fim);
        printf("%.2lf\n", valor);
        ini = fim;
        fread(&valor, sizeof(valor), 1, f);
    }
    fclose(f);
    return 0;
}
```

MOVENDO-SE PELO ARQUIVO

- De modo geral, o acesso a um arquivo é sequencial. Porém, é possível fazer buscas e acessos randômicos em arquivos.
- Para isso, existe a função **fseek**:

```
int fseek(FILE *fp, long numbytes, int origem);
```

- Esta função move a posição corrente de leitura ou escrita no arquivo em tantos bytes, a partir de um ponto especificado.

MOVENDO-SE PELO ARQUIVO

- A função **fseek** recebe 3 parâmetros
 - **fp**: o ponteiro para o arquivo;
 - **numbytes**: é o total de bytes a partir de **origem** a ser pulado;
 - **origem**: determina a partir de onde os **numbytes** de movimentação serão contados.
- A função devolve o valor 0 quando bem sucedida

```
int fseek(FILE *fp, long numbytes, int origem);
```

MOVENDO-SE PELO ARQUIVO

- Os valores possíveis para **origem** são definidos por macros em **stdio.h** e são:

Nome	Valor	Significado
SEEK_SET	0	Início do arquivo
SEEK_CUR	1	Ponto corrente do arquivo
SEEK_END	2	Fim do arquivo

- Portanto, para mover **numbytes** a partir
 - do início do arquivo, **origem** deve ser SEEK_SET
 - da posição atual, **origem** deve ser SEEK_CUR
 - do final do arquivo, **origem** deve ser SEEK_END
- numbytes** pode ser negativo quando usado com SEEK_CUR e SEEK_END

MOVENDO-SE PELO ARQUIVO

- Exemplo da função **fseek**

```
struct cadastro{ char nome[20], rua[20]; int idade;};
int main(){
    FILE *f = fopen("arquivo.txt", "wb");

    struct cadastro c, cad[4] = {"Ricardo", "Rua 1", 31,
                                   "Carlos", "Rua 2", 28,
                                   "Ana", "Rua 3", 45,
                                   "Bianca", "Rua 4", 32};

    fwrite(cad, sizeof(struct cadastro), 4, f);
    fclose(f);

    f = fopen("arquivo.txt", "rb");
    fseek(f, 2*sizeof(struct cadastro), SEEK _ SET);
    fread(&c, sizeof(struct cadastro), 1, f);
    printf("%s\n%s\n%d\n", c.nome, c.rua, c.idade);
    fclose(f);

    return 0;
}
```

MOVENDO-SE PELO ARQUIVO

- Outra opção de movimentação pelo arquivo é simplesmente retornar para o seu início.
- Para tanto, usa-se a função **rewind**:

```
void rewind(FILE *fp);
```

APAGANDO UM ARQUIVO

- Além de permitir manipular arquivos, a linguagem C também permite apagá-lo do disco. Isso pode ser feito utilizando a função **remove**:

```
int remove(char *nome_do_arquivo);
```

- Diferente das funções vistas até aqui, esta função recebe o **caminho e nome** do arquivo a ser excluído, e não um ponteiro para FILE.
- Como retorno temos um valor inteiro, o qual será igual a 0 se o arquivo for excluído com sucesso.

APAGANDO UM ARQUIVO

- Exemplo da função **remove**

```
int main() {  
    int status;  
    status = remove("ArqGrav.txt");  
    if(status != 0) {  
        printf("Erro na remocao do arquivo.\n");  
        system("pause");  
        exit(1);  
    } else  
        printf("Arquivo removido com sucesso.\n");  
  
    return 0;  
}
```