

Introdução à Programação

Arquivos Binários

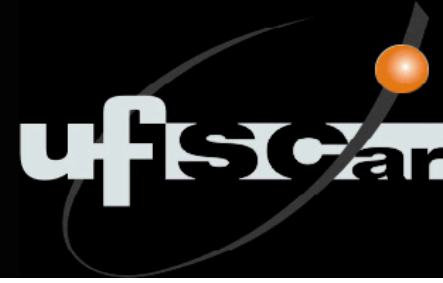
Prof. Tiago A. Almeida

`talmeida@ufscar.br`

Departamento de Computação
Universidade Federal de São Carlos

Introdução à Programação

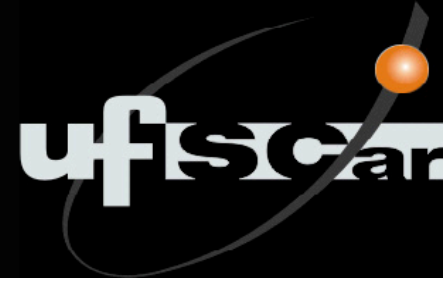
Motivação



- Variáveis `int` ou `float` têm tamanho fixo na memória. Por exemplo, um `int` ocupa 4 bytes
- Representação em texto precisa de um número variável de dígitos (10, 5.673, 100.340), logo de um tamanho variável
- Armazenar dados em arquivos de forma análoga a utilizada em memória permite:
 - Reduzir o tamanho do arquivo
 - Realizar busca não sequencial

Introdução à Programação

Abertura

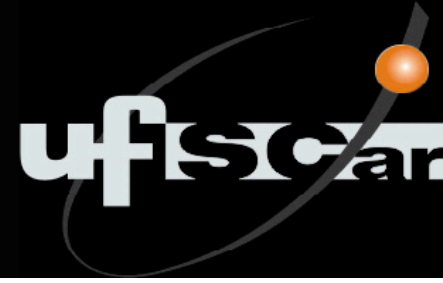


- `FILE* fopen(char *nome, char *modo);`
 - Abre o arquivo **nome** com o **modo** dado
 - **nome**: string com nome / caminho do arquivo
 - **modo**: string que indica se o arquivo será aberto para leitura, escrita, ambos, ou variações
 - Modo `"rb"` – apenas leitura (início). O arquivo deve existir.
 - Modo `"wb"` – apenas escrita (início). Cria um novo arquivo.
 - Modo `"ab"` – apenas escrita, mantém os dados originais (final). Cria arquivo caso não exista.
 - Modo `"r+b"` – leitura e escrita (início). O arquivo deve existir.
 - Modo `"w+b"` – leitura e escrita (início). Cria um novo arquivo.
 - Modo `"a+b"` – leitura (início) e escrita (final). Cria arquivo, caso não exista.

- <http://www.cplusplus.com/reference/cstdio/fopen/>

Introdução à Programação

Leitura e escrita de dados



- As funções `fread` e `fwrite` permitem a leitura e escrita de blocos de dados
- A ideia é semelhante a alocação de memória dinâmica
- Devemos determinar o número de elementos a serem lidos ou gravados e o tamanho de cada um

```
size_t fread(void *ptr, size_t size,  
             size_t nmemb, FILE *stream);  
  
size_t fwrite(const void *ptr, size_t size,  
              size_t nmemb, FILE *stream);
```

- As funções necessárias para manipulação estão na `stdio.h`

Introdução à Programação

Leitura de dados



```
int fread (void *dados, int tam_elem, int num_elem, FILE *f);
```

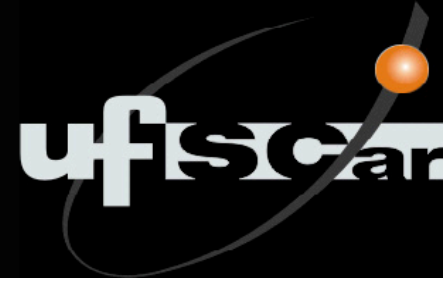
- Lê do arquivo `f`, `num_elem` elementos de `tam_elem` bytes cada e armazena na variável `dados`
- Retorna o número de elementos lidos com sucesso

```
/* Le um inteiro do arquivo 'myfile.bin' */  
  
FILE *f;  
int val;  
  
f = fopen ( "myfile.bin" , "rb" );  
fread(&val, sizeof(int), 1, f);  
fclose (f);
```

- <http://www.cplusplus.com/reference/cstdio/fread/>

Introdução à Programação

Escrita de dados



```
int fwrite (void *dados, int tam_elem, int num_elem, FILE *f);
```

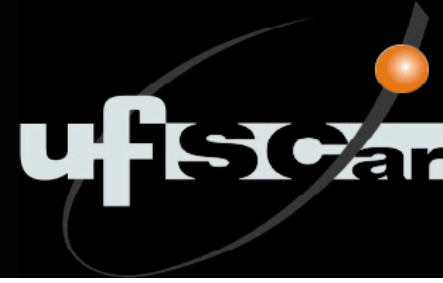
- Escreve no arquivo `f`, `num_elem` elementos de `tam_elem` bytes cada, armazenados na variável `dados`
- Retorna o número de elementos gravados com sucesso

```
/* Escreve conteudo de buffer no arquivo 'myfile.bin */  
  
FILE * pFile;  
char buffer[] = { 'x' , 'y' , 'z' };  
  
pFile = fopen ( "myfile.bin" , "wb" );  
fwrite (buffer , sizeof(char) , sizeof(buffer) , pFile );  
fclose (pFile);
```

- <http://www.cplusplus.com/reference/cstdio/fwrite/>

Introdução à Programação

Exemplos



- Ver exemplos:

- `fwrite-fread.c`
- `copiar.c`

Introdução à Programação

Vetores



```
/* Lê vetor de arq.texto e escreve em arq. binario */

FILE *fr, *fw;
int v1[100], tam;
int i;

fr = fopen ("v-in.txt", "r");

fscanf(fr, "%d", &tam); // le tamanho do vetor
for (i = 0; i < tam; i++)
    fscanf(fr, "%d", &v1[i]);

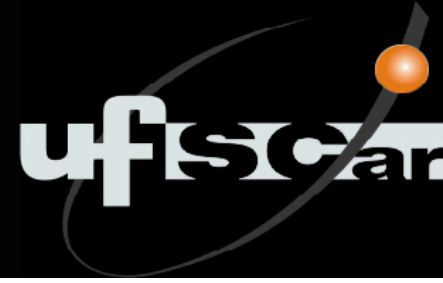
fclose(fr);

fw = fopen ("v.bin", "wb"); // cria o arquivo binario
fwrite(&tam, sizeof(int), 1, fw); // escreve o tamanho do vetor
fwrite(v1, sizeof(int), tam, fw); // escreve o vetor

fclose(fw);
```


Introdução à Programação

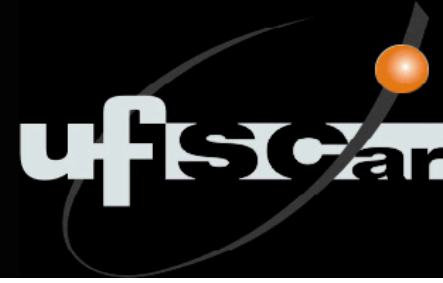
Acesso direto



- Podemos fazer o acesso não sequencial usando a função `fseek`
- Esta função altera a posição de leitura/escrita no arquivo
- O deslocamento pode ser relativo ao:
 - Início do arquivo
 - Ponto atual
 - Final do arquivo

Introdução à Programação

Acesso direto

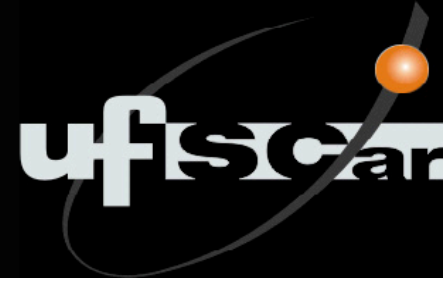


```
int fseek(FILE *f, long distancia, int origem);
```

- Move o cursor do arquivo `f` para a posição `distancia` relativa a alguma origem. A origem deve ser uma das três constantes:
 - `SEEK_SET` – início do arquivo
 - `SEEK_CUR` – posição atual
 - `SEEK_END` – fim do arquivo
- `fseek` retorna 0 em caso de sucesso
- <http://www.cplusplus.com/reference/cstdio/fseek/>

Introdução à Programação

Acesso direto



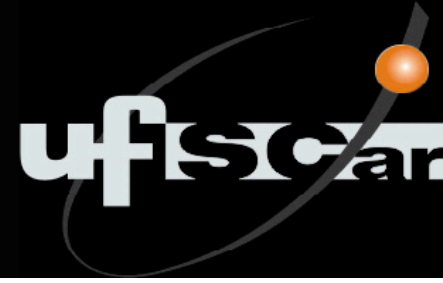
```
int fseek(FILE *f, long distancia, int origem);
```

```
FILE *pFile;  
  
pFile = fopen ( "example.txt" , "w" );  
  
fputs ( "This is an apple." , pFile );  
  
fseek ( pFile , 9 , SEEK_SET );  
  
fputs ( " sam" , pFile );  
  
fclose ( pFile );
```

- Ver exemplo `fseek.c`

Introdução à Programação

Registros



- Um arquivo pode armazenar registros (como um banco de dados)
- Isso pode ser feito de forma bem fácil se lembrarmos que um registro, como qualquer variável em C, tem um tamanho fixo
- O acesso a cada registro pode ser direto, usando a função `fseek`
- A leitura ou escrita do registro pode ser feita usando as funções `fread` e `fwrite`
- Ver o exemplo em `registro.c`