

33. Manipulação de arquivos

DIM0321

2015.1

Outline

- 1 Introdução
- 2 Leitura e escrita
- 3 Exemplos
- 4 Linha de comando e C

DIM0321

33. Manipulação de arquivos

2015.1 1 / 24

- 1 Introdução
- 2 Leitura e escrita
- 3 Exemplos
- 4 Linha de comando e C

DIM0321

33. Manipulação de arquivos

2015.1 3 / 24

DIM0321

33. Manipulação de arquivos

2015.1 2 / 24

Arquivo = entidade de armazenamento de informação, sequência de bytes que codificam a informação, de acordo com o formato de representação ()

DIM0321

33. Manipulação de arquivos

2015.1 4 / 24

Princípios

A comunicação é feita através de canais de comunicação.

Arquivo -> stream -> Processamento -> stream -> Arquivo

Processo

- Declarar uma variável para armazenar o canal de comunicação
- Abrir o arquivo (o resultado da chamada à sub-rotina é o canal de comunicação)
- Ler e/ou escrever no arquivo através do canal de comunicação
- Fechar o canal de comunicação

Biblioteca padrão

Tipos e sub-rotinas para acessar arquivos são declarados na biblioteca padrão `stdio.h`

- Canais de comunicação são registros do tipo `FILE`
- Todas as sub-rotinas de acesso à arquivos usam endereços (ponteiros) de registros `FILE`

DIM0321

33. Manipulação de arquivos

2015.1 4 / 24

Abrir um canal

Função principal

```
1 | FILE* fopen (char *nome, char *modo);
```

```
1 | #include <stdio.h>
2 |
3 | int main(void)
4 | {
5 |     FILE *f;
6 |     f = fopen("foo.txt", "r");
7 |     return 0;
8 | }
```

DIM0321

33. Manipulação de arquivos

2015.1 6 / 24

Elementos

Modos

"r"	read	
"w"	write	
"a"	append	escrita no fim do arquivo
"rb"	read binary	
"wb"	write binary	
"ab"	append binary	
"r+"	read and write	

Modos = "r", "w", "a" (adição no final), "rb", "wb", "ab" (versões binárias), "r+" (leitura e escrita)

Abertura de arquivo

Deve-se sempre testar se o canal de comunicação foi estabelecido

- Testar se o ponteiro retornado é `NULL` (valor nulo)
- Se for `NULL`, então houve erro na abertura

Causas de falha

DIM0321

33. Manipulação de arquivos

2015.1 7 / 24

Exemplo

```
1 | #include <stdio.h>
2 |
3 | int main (void)
4 | {
5 |     FILE *f;
6 |     if (f = fopen ("foo.txt", "r") == NULL) {
7 |         eprintf ("Erro na abertura");
8 |     };
9 |     return 0;
10| }
```

1 Introdução

2 Leitura e escrita

3 Exemplos

4 Linha de comando e C

DIM0321

33. Manipulação de arquivos

2015.1

8 / 24

** As operações de leitura e escrita são similares às operações de leitura do teclado (scanf) e escrita na tela de console (printf)

Exemplo

```
1 | #include <stdio.h>
2 |
3 | int main(void)
4 | {
5 |     FILE *f;
6 |     if ((f = fopen ("foo.txt", "r")) != NULL) {
7 |         fscanf (f, "%i", &n);
8 |         fprintf (f, "%i", n);
9 |     };
10|     return 0;
11| }
```

fscanf e fprintf

fscanf

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

O retorno da sub-rotina fscanf corresponde ao:

- número de valores que foram lidos e atribuídos caso a leitura tenha sido bem sucedida
- valor da constante EOF (End Of File), caso contrário

fprintf

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

O retorno da sub-rotina fprintf corresponde ao:

- número de caracteres (bytes) escritos caso a escrita seja bem sucedida
- um valor negativo, caso contrário

DIM0321

33. Manipulação de arquivos

2015.1

10 / 24

DIM0321

33. Manipulação de arquivos

2015.1

10 / 24

Buffer

Funcionamento da escrita

- O canal de comunicação é geralmente “buferizado”
- O comando `fprintf` não escreve diretamente no arquivo, mas numa zona intermediária chamada de buffer
- Quando o buffer enche, ou o canal é fechado, ou é enviada uma nova linha, o conteúdo é descarregado no arquivo.
- É feito porque E/S são caras.

Observação

- A função `printf` pode ser vista como uma versão especializada de `fprintf` para o canal `stdout`.
- Esse canal é também buferizado.

Fechar o arquivo

A abertura de um arquivo demanda alocação de recursos:

- Memória usada para representar o canal de comunicação
- “Quota” que o sistema operacional atribui ao programa ou usuário

Portanto os recursos utilizados devem ser liberados quando o acesso ao arquivo não for mais necessário. Arquivos podem ser fechados através da sub-rotina `fclose()`. Recebe como parâmetro o endereço do canal de comunicação. Retorna 0 para operação bem sucedida, EOF caso contrário

Função

- `int fclose (FILE *canal)`

1 Introdução

2 Leitura e escrita

3 Exemplos

4 Linha de comando e C

Exemplo

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     int n, i, x, y;
6     printf("Quantos pontos ?");
7     scanf("%i", &n);
8     FILE *arquivo = fopen("pontos.txt", "w");
9     if (arquivo == NULL) {
10         printf("Erro na abertura do arquivo");
11     } else {
12         fprintf(arquivo, "numero = %i\n", n);
13         for (i = 0; i < n; i++) {
14             printf("Digite um ponto %i:\n", i + 1);
15             scanf("%i %i", &x, &y);
16             fprintf(arquivo, "%i,%i\n", x, y);
17         }
18         fclose( arquivo );
19     }
20     return 0;
21 }
```

Exemplo

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     int n, i, x, y;
6     FILE *arquivo = fopen("pontos.txt", "r");
7     if (arquivo == NULL) {
8         printf( "Erro na abertura do arquivo" );
9     } else {
10         fscanf(arquivo, "numero = %i\n", &n);
11         printf("Existe %i pontos no arquivo\n", n);
12         for (i = 0 ; i < n; i++) {
13             fscanf(arquivo, "%i,%i\n", &x, &y);
14             printf("ponto %i: %i,%i\n", i + 1, x, y);
15         }
16         fclose(arquivo);
17     }
18     return 0;
19 }
```

Leitura / escrita por caractere

Funções adicionais

- Existem outras sub-rotinas de leitura e de escrita
- Leitura e escrita de caractere por caractere

```
1 int fputc(int caractere, FILE *canal);
2 int fgetc(FILE *canal );
```

Observações

- Ao invés de texto, o conteúdo do arquivo é uma codificação binária de um unsigned char.
- O primeiro parâmetro de fputc é convertido em unsigned char.
- fgetc retorna o próximo byte no arquivo ou EOF caso seja o fim.

Exemplo

```
1 #include <stdio.h>
2 int main(void)
3 {
4     char *arquivo = "contas.bin";
5     int valor = 65;
6     FILE *canal = fopen(arquivo, "w");
7     if (canal != NULL) {
8         fputc(valor, canal);
9         fclose(canal);
10        canal = fopen(arquivo, "r");
11        if (canal != NULL) {
12            valor = fgetc(canal);
13            fclose(canal);
14        }
15    }
16    return 0;
17 }
```

Canais de comunicação padrão

Canais

Os programas em linguagem C também apresentam três canais de comunicação sempre abertos

- Canal de conexão com o teclado = Entrada padrão stdin
- Canais de conexão com a tela de console: Saída padrão stdout, Saída de erros padrão stderr

Obs.: printf ("oi"); nada mais é que fprintf (stdout, "oi");

Uso dos canais no shell

- Canais de comunicação podem ser redirecionados usando comandos do terminal
- O canal stdin passa a apontar para o arquivo entrada.txt

```
1| % nome_programa < entrada.txt
```
- O canal stdout passa a apontar para o arquivo saida.txt

```
1| % nome_programa > saida.txt
```
- O canal stderr passa a apontar para o arquivo erros.txt

```
1| nome_programa 2> erros.txt
```

Exemplo

```
1| nome_programa < entrada.txt > saida.txt 2> erros.txt
```

1 Introdução

2 Leitura e escrita

3 Exemplos

4 Linha de comando e C

DIM0321

33. Manipulação de arquivos

2015.1 19 / 24

argc/argv

Parâmetros padrão

A função main tem 2 parâmetros sempre definidos

- `int argc` : o número de elementos na linha de comando
 - $argc \geq 1$
- `char *argv[]` : um arranjo dos elementos no formato de strings.
 - `argv[0]` contém o nome do programa

Exemplo

```
1| #include <stdio.h>
2| int main(int argc, char *argv[])
3| {
4|     printf("argc = %d\n", argc);
5|     for (int i = 0; i < argc; i++)
6|         printf("argv[%d] = %s\n",
7|             i,
8|             argv[i]);
9|     return 0;
10| }
```

```
1| % ./a.out
```

```
1| argc = 1
2| argv[0] = ./a.out
```

```
1| % ./a.out 1 a 1.2
```

```
1| argc = 4
2| argv[0] = ./a.out
3| argv[1] = 1
4| argv[2] = a
5| argv[3] = 1.2
```

DIM0321

33. Manipulação de arquivos

2015.1 21 / 24

DIM0321

33. Manipulação de arquivos

2015.1 22 / 24

Referências

Precisões

[?] 1, 7, 8

[?] 12

Perguntas ?



<http://dimap.ufrn.br/~richard/dim0321>