33. Manipulação de arquivos

DIM0321

2015.1

Sumário

Definições

Definição (Arquivo)

- Entidade de armazenamento de informação,
- Sequência de bytes que codificam a informação, de acordo com o formato de representação
- Arquivos são armazenados pelo sistema na memória física do computador de acordo com um sistema de arquivo (*file system*).

Definição (Sistema de arquivo)

- Gera a organização dos dados num meio de armazenamento.
- Ele deve saber:
  - Criar um arquivo
  - Gravar um arquivo
  - Ler um arquivo
  - Posicionar-se dentro de um arquivo
  - Apagar um arquivo
  - Troca um arquivo

Princípios em C

Comunicação

A comunicação é feita através de canais de comunicação (abstração `FILE`).
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

Onde

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

Elementos

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

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 }
```

Modos

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

Observações

- A escrita com "w" apaga tudo que existia no arquivo. Se o arquivo não existir, ele é criado.
- A abertura em leitura ("r") precisa que o arquivo exista.

Dicas

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

A abertura pode falhar nos seguintes casos:

- Quando o arquivo requerido não existe
- Quando o arquivo requerido está bloqueado por outro programa
- Quando não temos permissão de acesso adequada (escrita, leitura) para o arquivo requerido

Exemplo

```c
#include <stdio.h>

int main (void)
{
     FILE *f;
     if ((f = fopen ("foo.txt", "r")) == NULL) {
          fprintf(stderr, "Erro na abertura");
     };
     return 0;
}
```

Lembretes

Observações

- As operações de leitura e escrita são similares às operações de leitura do teclado (`scanf`) e escrita na tela de console (`printf`)
- Em vez delas, usaremos `fscanf` e `fprintf`

Exemplo

```c
#include <stdio.h>

int main(void)
{
     FILE *f;
     if ((f = fopen("foo.txt", "r")) != NULL) {
          fscanf(f, "%i", &n);
          fprintf(f, "%i", n);
     };
     return 0;
}
```

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

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

Por que fechar ?

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

Os recursos utilizados devem ser liberados quando o acesso ao arquivo não for mais necessário. Arquivos podem ser fechados através da sub-rotina `fclose()`.

Função

- `int fclose (FILE *canal)`
- Recebe como parâmetro o endereço do canal de comunicação.
- Retorna 0 para operação bem sucedida, EOF caso contrário

Exemplo

```c
#include <stdio.h>

int main(void)
{
    int n, i, x, y;
    printf("Quantos pontos ?");
    scanf("%i", &n);
    FILE *arquivo = fopen("pontos.txt", "w");
    if (arquivo == NULL) {
        printf("Erro na abertura do arquivo");
    } else {
        fprintf(arquivo, "numero = %i\n", n);
        for (i = 0; i < n; i++) {
            printf("Digite um ponto %i:\n", i + 1);
            scanf("%i %i", &x, &y);
            fprintf(arquivo, "%i,%i\n", x, y);
        }
        fclose( arquivo );
    }
    return 0;
}
```

Exemplo

```c
#include <stdio.h>

int main(void)
{
    int n, i, x, y;
    FILE *arquivo = fopen("pontos.txt", "r");
    if (arquivo == NULL) {
        printf( "Erro na abertura do arquivo" );
    } else {
        fscanf(arquivo, "numero = %i\n", &n);
        printf("Existe %i pontos no arquivo\n", n);
        for (i = 0 ; i < n; i++) {
            fscanf(arquivo, "%i,%i\n", &x, &y);
            printf("ponto %i: %i,%i\n", i + 1, x, y);
        }
        fclose(arquivo);
    }
    return 0;
}
```

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

```c
#include <stdio.h>
int main(void)
{
    char *arquivo = "contas.bin";
    int valor = 65;
    FILE *canal = fopen(arquivo, "w");
    if (canal != NULL) {
        fputc(valor, canal);
        fclose(canal);
        canal = fopen(arquivo, "r");
        if (canal != NULL) {
            valor = fgetc(canal);
            fclose(canal);
        }
    }
    return 0;
}
```

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");` é nada mais 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
```

argc/argv

Parâmetros padrão

A função `main` tem 2 parâmetros sempre definidos

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

Exemplo

```c
#include <stdio.h>
int main(int argc, char *argv[])
{
      printf("argc = %d\n", argc);
      for (int i = 0; i < argc; i++)
            printf("argv[%d] = %s\n",
                  i,
                  argv[i]);
      return 0;
}
```

```
% ./a.out
```

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

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

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

Referências

Precisões

[KR88] 1, 7, 8

[Bac13] 12

André Backes, *Linguagem C completa e descomplicada*, Elsevier, 2013.

Brian W. Kernighan and Dennis M. Ritchie, *The (ANSI) C Programming Language*, 2nd ed., Prentice Hall Professional Technical Reference, 1988.

Perguntas ?

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