Linguagem de Programação C

Arquivos Binários

Cristiano Lehrer

http://www.ybadoo.com.br/

Motivação

- Variáveis `int` e `float` tem tamanho fixo em memória:
  - `int` – 2 *bytes*.
  - `float` – 4 *bytes*.
- A representação em texto dos números precisa de uma quantidade variável de caracteres:
  - `1, 123, 23.34, -54.678, 5000.78`
- Armazenando os números nos arquivos conforme eles são armazenados em memória permite:
  - Reduzir o tamanho do arquivo.
  - Permitir uma busca não sequencial pelas informações.

Modos de Abertura (1/2)

Modo de Abertura	Breve Descrição	Permite Leitura	Permite Escrita	Se o Arquivo não Existe	Se o Arquivo Existe	Posição Inicial
r	Leitura	Sim	Não	NULL	OK	Início
w	Escrita	Não	Sim	Cria	Recria	Início
a	Acrescentar	Não	Sim	Cria	OK	Fim
r+	Ler/Escrever	Sim	Sim	Cria	Permite Alterar Dados	Início
w+	Ler/Escrever	Sim	Sim	Cria	Recria	Início
a+	Ler/Escrever	Sim	Sim	Cria	Permite Acrescentar Dados	Fim

Caso exista algum erro de abertura, a função `fopen` retornar o valor `NULL`.

Modos de Abertura (2/2)

- Modo Texto
  - Por padrão, todos os arquivos são abertos em modo texto.
  - Um caractere de nova linha: `"\n"`
- Modo Binário
  - Para que o arquivo seja aberto em modo binário, o flag `"b"` deverá ser adicionado ao flag de abertura do arquivo, como em `"rb"`, `"wb"`, `"ab"`, `"a+b"` e assim por diante;
  - Dois caractere de nova linha: `CR+LF`

Fazer uma Cópia do Arquivo (1/2)

```
#include <stdio.h>
int main(int argc, char *argv[])
{
  FILE *fin, *fout;
  char ch;
  if(argc != 3){
    printf("Sintaxe: \n\n%s Origem Destino\n\n", argv[0]);
    return 0;
  }
  if((fin = fopen(argv[1], "rb")) == NULL){
    printf("Impossível abrir o arquivo %s\n", argv[1]);
    return 0;
  }
```

Fazer uma Cópia do Arquivo (2/2)

```
  if((fout = fopen(argv[2], "wb")) == NULL){
    printf("Impossível criar o arquivo %s\n", argv[2]);
    return 0;
  }

  ch = fgetc(fin);
  while(!feof(fin)){
    fputc(ch, fout);
    ch = fgetc(fin);
  }

  fclose(fin);
  fclose(fout);
  return 0;
}
```

Escrita (1/2)

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

- A função `fwrite` escreve `nmemb` elementos de dados, cada um com `size` *bytes* de comprimento, para o fluxo apontado por `stream`, obtendo-os da localização apontada por `ptr`.
- A função retorna o número de itens que foram gravados com sucesso.
- Caso ocorra um erro o valor de retorno é menor do que `nmemb` ou zero.

Escrita (2/2)

```c
#include <stdio.h>
int main (){
  int inum = 10;
  float fnum = 2.5;
  double pi = 3.141516;
  char c = 'Z';

  FILE *pa;
  char *nome = "teste.dat";

  if((pa = fopen(nome, "w+b")) == NULL){
    printf("\nNao foi possivel abrir o arquivo para escrita.\n");
    return 0;
  }

  fwrite(&inum, sizeof(int), 1, pa);
  fwrite(&fnum, sizeof(float), 1, pa);
  fwrite(&pi, sizeof(double), 1, pa);
  fwrite(&c, sizeof(char), 1, pa);

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

Leitura (1/2)

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

- A função `fread` lê `nmemb` objetos, cada um com `size` *bytes* de comprimento do fluxo apontado por `stream` e os coloca na localização apontada por `prt`.
- Caso ocorra um erro ou o fim do arquivo seja atingido, o valor de retorno é menor do que nmemb ou zero.
- Esta função não distingue entre um fim de arquivo e erro, portanto é aconselhável o uso de `feof()` ou `ferror()` para determinar o que ocorreu.

Leitura (2/2)

```c
#include <stdio.h>
int main (){
  int inum;
  float fnum;
  double pi;
  char c;
  FILE *pa;
  char *nome = "teste.dat";

  if((pa = fopen(nome, "r+b")) == NULL){
    printf("\nNao foi possivel abrir o arquivo para escrita.\n");
    return 0;
  }

  fread(&inum, sizeof(int), 1, pa);
  fread(&fnum, sizeof(float), 1, pa);
  fread(&pi, sizeof(double), 1, pa);
  fread(&c, sizeof(char), 1, pa);
  printf("%d, %f, %lf, %c\n", inum, fnum, pi, c);

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

Fazer uma Cópia do Arquivo (1/2)

```
#include <stdio.h>
int main(int argc, char *argv[])
{
  FILE *fin, *fout;
  char ch;
  if(argc != 3){
    printf("Sintaxe: \n\n%s Origem Destino\n\n", argv[0]);
    return 0;
  }
  if((fin = fopen(argv[1], "rb")) == NULL){
    printf("Impossível abrir o arquivo %s\n", argv[1]);
    return 0;
  }
```

Fazer uma Cópia do Arquivo (2/2)

```
  if((fout = fopen(argv[2], "wb")) == NULL){
    printf("Impossível criar o arquivo %s\n", argv[2]);
    return 0;
  }

  while((fread(&ch, sizeof(char), 1, fin)) == 1)
  {
    fwrite(&ch, sizeof(char), 1, fout);
  }

  fclose(fin);
  fclose(fout);
  return 0;
}
```

Entrada e Saída Formatadas

```
int fscanf(FILE *arq, const char *format, ...)
int fprintf(FILE *arq, const char *format, ...)
```

- A função `fscanf` retorna EOF se tiver detectado *End-of-File* ou retorna o número de parâmetros que conseguiu ler com sucesso;
- No exemplo anterior, considerando que a cópia fosse de arquivos textos e que `ch` seja declarado como um `char`:

```
while(fscanf(fin, "%c", &ch) != EOF)
{
  fprintf(fout, "%c", ch);
}
```

Arquivos Padrões

Arquivo	Descrição
stdin	Representa a entrada-padrão e está normalmente associada ao teclado.
stdout	Representa a saída-padrão e está normalmente associada a tela.
stderr	Representa o *standard error* (local para onde devem ser enviadas as mensagens de erro de um programa). É semelhante ao **stdout** e permite enviar as mensagens de erro para um local diferente da saída de um programa.
stdaux	Representa o denominado *aux device* e está definido como a porta principal de comunicações (COM1 num PC).
stdprn	Representa o *standard printer* (impressora padrão).

Os cinco arquivos padrões são automaticamente abertos sempre que um programa começa a executar, não necessitando ser abertos pelo programador.

Comando equivalentes

```
printf("Um Dois Três");          fprintf(stdout, "Um Dois Três");
puts("Olá");                     fputs("Olá\n", stdout);
scanf("%d %c", &a, &b);          fscanf(stdin, "%d %c", &a, &b);
```