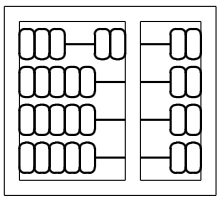

MC102 – ALGORITMOS E PROGRAMAÇÃO DE COMPUTADORES
Instituto de Computação — UNICAMP
1º Semestre de 2005

Aula 22 - Arquivos

1 Objetivos

Apresentar e exemplificar as funções de manipulação de arquivos existentes na biblioteca padrão `stdio.h`.

2 Motivação

Fazer programas que manipulem estruturas de dados em arquivos, usando registros, por exemplo..

3 Porque usar arquivos

A vantagem de se utilizar arquivos é que o acesso a ele pode ser feito em qualquer posição, desde que se saiba a posição em que se encontra o dado. Por isso é muito comum utilizar arquivos para escrever registros de tamanho fixo. Assim, para ler o quinto registro, sabemos que a posição que ele se encontra no arquivo é a $5 * sizeof(registro)$.

Outra vantagem é poder armazenar uma quantidade muito grande de dados, até maior que a memória disponível, já que normalmente somente uma pequena parte do arquivo é carregada na memória por vez.

Os arquivos são armazenados em dispositivos secundários de memória, por exemplo disco rígido, CD-ROM ou disquete. Isso traz mais uma vantagem porque estes dispositivos são não voláteis, ou seja, ao contrário da memória principal, eles retêm a informação mesmo depois do reboot da máquina. Cada arquivo possui um nome e algumas informações adicionais como data de criação e modificação. Porém, a desvantagem dos dispositivos secundários é de ter o acesso bem mais lento que a memória.

Os arquivos podem ser gravados de duas maneiras: como texto ou como binário. O arquivo texto armazena caracteres que podem ser exibidos na tela do terminal ou até modificados com um editor de textos puro, já o arquivo binário não pode ser exibido corretamente, ele deve ser consultado somente pelo aplicativo que o criou (ou outro compatível). O formato binário armazena dados como sequências de bits, e é normalmente ocupa menos espaço do que o arquivo texto.

4 Operações com arquivos

Assim como outras operações de entrada e saída, devemos incluir a biblioteca `stdio.h` para trabalhar com arquivos. Os arquivos são manipulados com variáveis do tipo apontador para arquivo, que são declaradas da seguinte forma:

```
#include <stdio.h>

int main()
{
  FILE *arq;
}
```

Operações de leitura e escrita para arquivos são muito similares às já conhecidas para leitura do teclado e escrita na tela. A diferença é que seu nome começa com a letra `f` e o primeiro argumento dessas novas funções passa a ser a variável que aponta para um arquivo. Por exemplo:

```
/* definindo variáveis */
char nome[] = "Fernando Henrique Cardoso";
int num;

/* escrever um nome para tela */
printf("O nome é %s.", nome);

/* escrever um nome para um arquivo controlado pela variável arq */
fprintf(arq, "O nome é %s.", nome);

/* lendo um número do teclado */
scanf("%d", &num);

/* lendo um número do arquivo */
fscanf(arq, "%d", &num);
```

Antes de ler e gravar arquivos, temos que indicar em qual arquivo queremos operar e se queremos ler ou gravar dados (ou ambos). Isso é feito com uma operação de abertura de arquivo. De modo similar, após a manipulação do arquivo, deveremos realizar uma operação para fechar o arquivo e com isso armazená-lo da memória principal de volta para o disco.

```
#include <stdio.h>

int main()
{
  FILE *arq;

  /* abrir arquivo:
     fopen (nome_arquivo, modo_de_operação);
     o modo de operação pode ser leitura ("r"), escrita ("w") ou ambos ("r+")
     também existem outros modos, como escrita no fim do arquivo ("a")
  */
  arq = fopen ("teste.txt", "w");

  /* escrever um texto no arquivo */
  fprintf(arq, "Testando a fprintf.");

  /* fechar arquivo:
     fclose (variavel_controle_arquivo);
  */
  fclose(arq);
}
```

5 Gravando e lendo vetores de arquivos

Vamos usar as funções básicas de arquivos para criar dois programas: um que cria um arquivo com o vetor de números lido pelo teclado e outro que abre um arquivos com números e os imprime na tela.

Primeiro criamos o programa que grava um vetor em arquivo.

```
#include <stdio.h>

#define MAX 1000
```

```c
int main()
{
  int i, n, vet[MAX];
  char nome_arquivo[] = "vetor1.txt";
  FILE *arq;

  /* ler um vetor do teclado */
  printf("Digite o numero de elementos do vetor: ");
  scanf("%d", &n);
  printf("Digite os elementos do vetor: ");
  for (i = 0; i < n; i++) scanf("%d", &vet[i]);

  /* abrir um arquivo para escrita em modo texto */
  arq = fopen (nome_arquivo, "w");

  /* escrever no arquivo o tamanho do vetor e os elementos */
  /* obs: não esquecer de gravar espaços entre os números */
  fprintf(arq, "%d ", n);
  for (i = 0; i < n; i++) fprintf(arq, "%d ", vet[i]);

  /* fechar o arquivo */
  fclose (arq);
}
```

Note que o primeiro número gravado no arquivo é o tamanho do vetor, e não um número que está dentro do vetor. Os números seguintes são os dados do vetor em sequência. Agora vamos criar o segundo programa, que lê um arquivo gerado seguindo a mesma regra e remonta o vetor em memória.

```c
#include <stdio.h>
#define MAX 1000

int main()
{
  int i, n, vet[MAX];
  char nome_arquivo[] = "vetor1.txt";
  FILE *arq;

  /* abrir um arquivo binário para leitura */
  arq = fopen (nome_arquivo, "r");

  /* ler o tamanho e os dados do vetor do arquivo */
  fscanf (arq, "%d", &n);
  for (i = 0; i < n; i++) fscanf (arq, "%d", &vet[i]);

  /* fechar o arquivo */
  fclose (arq);

  /* mostrar o vetor na tela */
  printf("Foi lido um vetor de tamanho %d com os elementos: ", n);
  for (i = 0; i < n; i++) printf("%d ", vet[i]);
}
```

6 Exercícios

1) Faça um programa que leia do teclado um número inteiro e um nome de arquivo. Grave o número lido no arquivo e confira, com um editor de texto, se o arquivo foi realmente gravado.

2) Faça agora um programa que leia um nome de arquivo pelo teclado e mostre na tela o primeiro número inteiro que encontrar dentro do arquivo.

3) Mude os seus exercícios de vetores e matrizes para que eles gravem e leiam arquivos.

7 Referências

Estas aulas foram baseadas nas notas de aula do **prof. Alexandre Falcão**

(http://www.dcc.unicamp.br/ afalcao/mc102/notas-aula.pdf)

na apostila do **prof. Flávio Keidi Miyazawa** e no material de aula do **prof. André Shiguemoto**.