MC 102 turma Z - 2sem2012
Linguagem C
Arquivos

Prof. Fernando Vanini

IC - Unicamp

Introdução

Nos exemplos mostrados até aqui, os dados utilizados são mantidos em variáveis, ou seja, na memória principal da máquina. Esses dados são perdidos quando o programa encerra sua execução.

Arquivos constituem a forma padrão de se manter os dados de forma que possam ser usados após o encerramento do programa. Os dados mantidos em arquivos podem ser usados por outros programas.

Um arquivo é uma sequência de dados mantida num meio permanente. O meio permanente mais comum é o disco magnético (ou HD, de ‘hard disk’).

Arquivos são criados e gerenciados pelo sistema operacional. Todo uso de arquivos feito por um programa é feito através de serviços oferecidos pelo sistema operacional. Esses serviços são disponibilizados ao programa através de funções de bibliotecas (<stdio.h>).

A função fopen()

- A função fopen() é usada para iniciar uma ‘sessão de uso’ de um arquivo. Ela retorna um valor do tipo FILE*, que será usado para identificar o arquivo nas operações de escrita ou leitura.
- A sessão de uso de um arquivo se encerra com a chamada à função fclose() que tem como parâmetro o identificador do arquivo.

Um exemplo: criação de um arquivo

```
int main(){
        int a = 10; b = 20; char * n = "alfabeto";
        FILE* arq = fopen("arquivo1.txt","w");
        fprintf(arq,"%d %d %s",a,b,n);
        fclose(arq);
    }
```

Nesse exemplo:

- A função fopen() é usada para iniciar a 'sessão de uso' do arquivo. Neste caso, o arquivo chamado 'arquivo1.txt'. O valor 'w' para o segundo parâmetro indica 'escrita' (ou seja, o arquivo será criado para escrita).
- A variável arq, do tipo FILE* mantém a identificação do arquivo.
- A função fprintf() é muito parecida com printf(), velha conhecida. A diferença é o parâmetro a mais, que identifica o arquivo no qual é feita a escrita.
- A 'sessão de uso' do arquivo se encerra com fclose().
- O conteúdo do arquivo será uma linha texto contendo o string escrito através de fprintf(). Isso pode ser verificado através do Notepad, no Windows (ou um equivalente , no caso de Linux).

Outro exemplo: leitura do arquivo

```
int main(){
      int x,y; char s[30];
       FILE* arq = fopen("arquivo1.txt","r");
       fscanf(arq,"%d %d %s", &x, &y, s);
       printf("x:%d y:%d s:%s",x,y,s);
       fclose(arq);
   }
```

Nesse exemplo:

- As funções fopen() e fclose() delimitam 'sessão de uso' do arquivo. Neste caso, o arquivo chamado 'arquivo1.txt' será usado para leitura, o que é indicado pelo valor 'r' para o segundo parâmetro de fopen().
- A função fcanf() é muito parecida com a já conhecida scanf(). A única diferença é o parâmetro a mais, que identifica o arquivo de onde é feita a leitura.
- A 'sessão de uso' do arquivo se encerra com fclose().
- As varíaveis x, y e s, cujo valor foi lido, podem ser usadas em seguida, como quaisquer outras variáveis do programa.

Outro exemplo (1)

Definição de um vetor de estruturas:

```
struct Aluno{
        int ra;
        char *nome;
        int curso;
};

struct Aluno turmaZ[] = {
         {1, "Huguinho_da_Silva", 32 },
         {2, "Zezinho",  32 },
         {3, "Luizinho", 32 },
         {4, "Donald",   32 },
         {5, "Mickey",   34 },
         {6, "Minnie",   34 },
         {7, "Margarida",34 }
};
```

Outro exemplo (2)

Função para escrita do vetor de estruturas:

```
void criaArquivo(char * nome, struct Aluno *turma, int n){
     FILE* arq = fopen(nome,"w");
     int i,j;
     for(i = 0; i < n; i++) {
       fprintf( arq,"%d %s %d \n",
                turma[i].ra,
                turma[i].
                nome,turma[i].curso
               );

     }
     fclose(arq);
}
```

Outro exemplo (3)

Nesse exemplo:

- A ‘sessão de uso’ do arquivo é delimitada por fopen() e fclose();
- Uma vez iniciada da sessão, várias linhas são escritas no arquivo, cada uma contendo os dados de um elemento do vetor de estruturas.
- Os valores escritos através de fprintf() são campos de uma estrutura, mas poderiam ser quaisquer variáveis do programa.

- Após o encerramento da sessão, o arquivo pode ser ‘aberto’ novamente, para leitura ou escrita. Isso pode ser feito no mesmo programa ou por outro programa.
- Após o ‘fechamento’ do arquivo, o mesmo poderá ser identificado a partir do seu nome (no caso, o nome do arquivo é passado como parâmetro para a função criaArquivo();

Outro exemplo (4)

Função para leitura do arquivo:

```
void leArquivo(char *nomearq){
     FILE *entrada = fopen(nomearq,"r");
     int ra, curso; char nome[20]; int nvars;
     do{
         nvars = fscanf( entrada,"%d %s %d ",
                              &ra,nome,&curso
                       );
         if(nvars > 0)
            printf(
                     "%d %d %s %d\n",
                     nvars,ra,nome,curso
                   );
     }while(nvars > 0);
     fclose(entrada);
}
```

Outro exemplo (5)

Nesse exemplo:

- Usa o valor de retorno da função fscanf(). Esse valor de retorno indica o número de variáveis efetivamente lidas (como o arquivo pode ter sido criado por outro programa, é necessário verificar se os valores que se deseja ler estão presentes no arquivo).
- Os valores para os dados dos alunos são lidos e impressos enquanto houver linhas no arquivo contendo os dados dos alunos.
- Os exemplos 3 e 4, propositadamente usam variáveis diferentes para ler e escrever os dados no arquivo. O único ponto comum aos dois é o conteúdo do arquivo, na forma de texto.

Arquivos formatados e não formatados

Os exemplos apresentados:

- Usam as funções fprintf() e fscanf() para escrever e ler os dados no arquivo. Essas funções são meras versões de printf() e scanf() , orientadas à escrita e leitura através da ‘saída padrão’ e ‘entrada padrão’.
- Os dados escritos através de fprintf() são convertidos para texto antes da escrita. Os valores lidos através de fscanf() são lidos como texto que é posteriormente convertido para o tipo de dado especificado no formato; em seguida esse valor é armazenado na variável.
- Ao se usar essas funções, os dados efetivamente escritos e lidos nos arquivos são sempre na forma de texto.
- Por essa razão, os arquivos manipulados através dessas funções são chamados de ‘arquivos formatados’ ou ‘arquivos texto’.
- A leitura e escrita sem formatação é possível através de funções específicas. Nesse caso, os arquivos manipulados serão chamados de ‘arquivos não formatados’ ou ‘arquivos binários’.

Arquivos não formatados (binários)

- Quando se usa arquivos no modo ‘não formatado’, os dados são mantidos no arquivo exatamente como ficariam na memória – nenhuma conversão é feita durante as operações de escrita ou leitura.
- O modo ‘não formatado’ é indicado na operação fopen().
- As operações de leitura e escrita em modo não formatado são feitas através de funções específicas, definidas em <stdio.h>

Observação:

- Arquivos formatados são conhecidos como ‘arquivos texto’.
- Arquivos não formatados são conhecidos como ‘arquivos binários’.

Um exemplo: criação e escrita

```
int main(){
    int i;
    FILE *arq = fopen("alunos.dat","wb");
    for(i = 0; i < NA; i++){
        fwrite(&turmaZ[i], sizeof(struct Aluno), 1, arq);
    }
    fclose(arq);
    system("PAUSE");
}
```

- O exemplo usa o mesmo vetor de estruturas usado anteriormente.
- O valor “wb” para o segundo parâmetro de fopen() indica escrita em ‘modo não formatado’ (binário).
- A função fwrite() é responsável pela escrita em modo não formatado. Parâmetros:
  - Apontador para a variável a ser escrita
  - Tamanho de cada variável a ser escrita
  - Número de variáveis
  - Descritor do arquivo

Um exemplo: leitura

```
int main(){
    int i;
    FILE *arq = fopen("alunos.dat","rb");
    if(arq == NULL) printf("arquivo não encontrado\n");
    else {
         for(i = 0; i < NA; i++){
            fread(&turma[i], sizeof(struct Aluno), 1, arq);
            imprimeAluno(turma[i]);
         }
         fclose(arq);
    }
    system("PAUSE");
}
```

- O valor “rb” para o segundo parâmetro de fopen() indica leitura em ‘modo não formatado”.
- A função fread() é responsável pela leitura ‘não formatada’. Parâmetros:
  - Apontador para a área de memória onde serão armazenados os valores lidos
  - Tamanho de cada variável a ser lida
  - Número de variáveis
  - Descritor do arquivo
- A função fread() retorna um valor inteiro que indica o número de variáveis efetivamente lidas (não usado no exemplo).