ESTRUTURA DE DADOS

ARQUIVOS
E
MANIPULAÇÃO DE STRINGS

Profa. Fabrícia Damando Santos
fabriciadamando@gmail.com

Manipulação de arquivos

- Para tratar de arquivos a linguagem C fornece um nível de abstração entre o programador e o dispositivo que estiver sendo usado. Esta abstração é chamada fila de bytes e o dispositivo normalmente é o arquivo.
- Existe um sistema bufferizado de acesso ao arquivo, onde um ponteiro de arquivo define vários aspectos do arquivo, como nome, status e posição corrente, além de ter a fila associada a ele.
- Arquivos é uma boa opção para quem não possui o acesso a uma base de dados ou, simplesmente, para quem deseja armazenar informações básicas.
- Outra utilização para arquivos de baixo nível é dar "carga"em tabelas do BD, ou seja, através de um arquivo texto, registros serão incluídos em uma tabela.
- Para trabalharmos com arquivos devemos:
  - Abrir o arquivo em modo de leitura e/ou gravação;
  - Iniciar o processo de leitura e/ou gravação;
  - Fechar o arquivo.

Arquivos

- Um arquivo pode ser visto de duas maneiras,:
  - "modo texto", como um texto composto de uma seqüência de caracteres,
  - "modo binário", como uma seqüência de bytes (números binários).
- Podemos optar por salvar (e recuperar) informações em disco usando um dos dois modos, texto ou binário.
- Uma vantagem do *arquivo texto é que pode ser lido por uma pessoa e editado com editores de textos convencionais*.
- Em contrapartida, com o uso de um arquivo binário é possível salvar (e recuperar) grandes quantidades de informação de forma bastante eficiente.
- O sistema operacional pode tratar arquivos "texto" de maneira diferente da utilizada para tratar arquivos "binários".
- Em casos especiais, pode ser interessante tratar arquivos de um tipo como se fossem do outro, tomando os cuidados apropriados.

O que podemos fazer no C

- abertura de arquivos: o sistema operacional encontra o arquivo com o nome dado e prepara o buffer na memória.
- leitura do arquivo: o sistema operacional recupera o trecho solicitado do arquivo. Como o buffer contém parte da informação do arquivo, parte ou toda a informação solicitada pode vir do buffer.
- escrita no arquivo: o sistema operacional acrescenta ou altera o conteúdo do arquivo. A alteração no conteúdo do arquivo é feita inicialmente no buffer para depois ser transferida para o disco.
- fechamento de arquivo: toda a informação constante do buffer é atualizada no disco e a área do buffer utilizada na memória é liberada.

Arquivos

- Arquivos podem assumir formato ASCII ou binário

- Ponteiro para Arquivos (definido em stdio.h):

```
FILE *fid;
```

onde:

fid = ponteiro para arquivo

Principais funções

Nome	fopen
Descrição	Abre um arquivo. A função retornará false se ocorrer algum erro.
Sintaxe	fid=fopen(string nome_arquivo, string modo_de_abertura) Onde: nome_arquivo = nome do arquivo modo_de_abertura = define a forma como o arquivo será aberto conforme os parâmetros abaixo: •r - somente leitura a partir do início do arquivo •r+ - leitura e gravação a partir do início do arquivo •w – somente gravação a partir do início do arquivo, onde todo o conteúdo do arquivo será apagado. Se o arquivo não existir, a função tentará criá-lo. •w+ - como o anterior, efetuando leitura e gravação(cria novo arquivo ou sobrepõe se já existir) •a – somente leitura a partir do fim de um arquivo. •a+ - leitura e gravação a partir do fim de um arquivo •b: modo binário •t: modo texto

Nome	fclose
Descrição	Fecha um arquivo aberto. Retorna true se obtiver sucesso ou false em caso de erro.
Sintaxe	fclose(int fp) Ex: fid = fopen(“teste.txt”, “w+”); fclose(fid);

Nome	fwrite
Descrição	Escreve um número específico de bytes de uma string em um arquivo
Sintaxe	fwrite(dados, tamanho, quantidade, ptr_arq); Onde: dados = endereço de memória de onde dados serão lidos/gravados tamanho = número de bytes de cada item lido/gravado quantidade = número de itens lidos/gravados ptr_arq = ponteiro para arquivo

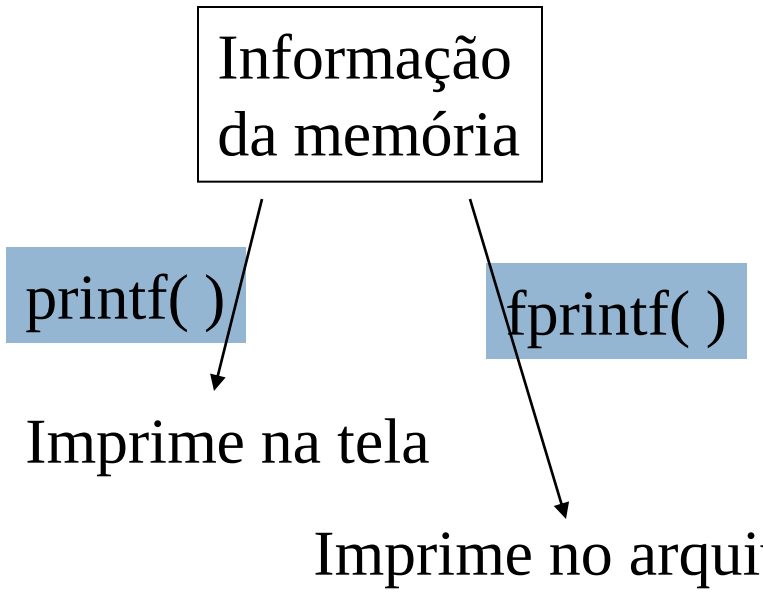
Nome	fread
Descrição	Usada para ler o arquivo
Sintaxe	fread(dados, tamanho, quantidade, ptr_arq); Onde: dados = endereço de memória de onde dados serão lidos/gravados tamanho = número de bytes de cada item lido/gravado quantidade = número de itens lidos/gravados ptr_arq = ponteiro para arquivo

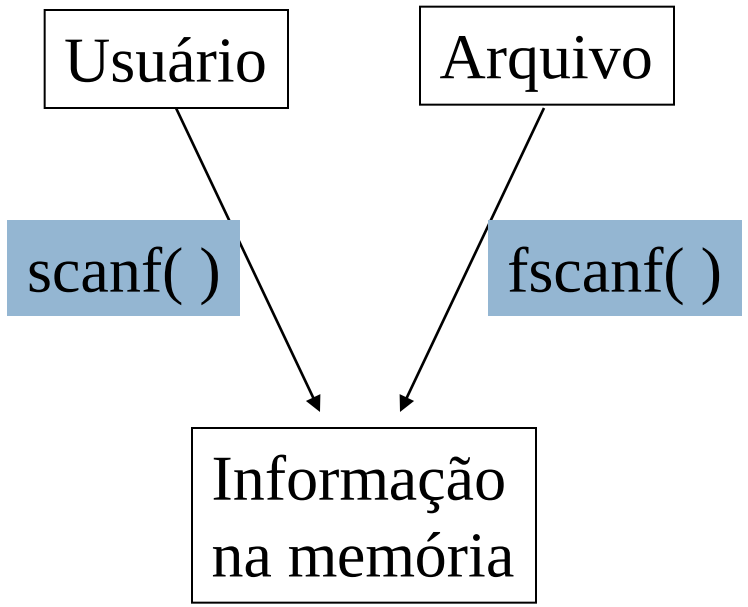
Nome	feof
Descrição	Retorna true caso o ponteiro de arquivo esteja no fim do arquivo ou ocorra um erro, caso contrário, retorna false.

Sintaxe	fid = fopen(nome_arq,"rt");
	while (!feof(fid)){
	...
	}

Leitura e escrita em modo texto

- A função utilizada para escrever no arquivo é fprintf( )
- A função utilizada para ler do arquivo é fscanf( )
- Cuidado para não fazer confusão com printf( ) e scanf( ) !





Exemplos - escrita e leitura modo texto


```
...
int num = 10; char string[10];
FILE *fid;

fid = fopen("teste.bin", "w+t");
fprintf(fid, "Num: %d" , num);
fclose(fid);

fid = fopen("teste.bin", "rt");
fscanf(fid, "%s", string); // lê a string "Num:"
fscanf(fid, "%d",&num); // lê um inteiro
fclose(fid);
...
```

Exemplos - escrita e leitura modo binário


```
...
int num = 10; char string[10];
FILE *fid;

fid = fopen("teste.bin","w+b");
fwrite("Num: ", sizeof(char), 10, fid); // escreve texto - fwrite(dados,
                                         //tamanho, quantidade, ptr_arq);
fwrite(&num, sizeof(int), 1, fid); // escreve valor de num
fclose(fid);


fid = fopen("teste.bin","rb");
fread(string, sizeof(char), 10, fid); // lê string
fread(&num, sizeof(int), 1, fid); // lê inteiro
fclose(fid);
...
```

Exemplo

Escreva um programa para armazenar em um arquivo texto nomes completos e idades que vão sendo inseridos, até que o nome “fim” seja inserido.

Depois faça um código para leitura do arquivo.

Código para gravar em arquivo

```
#include <stdio.h>
#include <string.h> //strcmp()
int main() {
    char nome_arq[20], nome[20], temp, i;
    int idade;
    FILE *fid;
    printf("\nEntre com o nome do arquivo: ");
    scanf("%s",nome_arq);  fflush(stdin);
    fid = fopen(nome_arq,"wt");
    if (fid == NULL) {
        printf("impossível abrir o arquivo para escrita...");
        //exit(0);
    }      // Inicio a leitura do nome fora do laço para que, ao digitar 'fim'não seja necessário inserir uma idade também
    printf("\nDigite um nome ou 'fim' para terminar:\n");
    gets(nome); fflush(stdin);
    while (strcmp(nome,"fim")!=0) {
      printf("Digite um idade:");
      scanf("%d",&idade); fflush(stdin);
      // Imprime no arquivo um ';' para facilitar na leitura do arquivo a identificar onde o nome termina, pois ele pode
     conter diversos sobrenomes
      fprintf(fid,"%s;%d\n",nome, idade);
      printf("\nDigite um nome ou 'fim' para terminar:\n");
      gets(nome); fflush(stdin);    }
   fclose(fid);
}
```

Código para ler do arquivo

```
#include <stdio.h>
#include <string.h> //strcmp()
int main() { char nome_arq[20], nome[20], temp, i;
    int idade;
    FILE *fid;
    printf("\nEntre com o nome do arquivo: ");
    scanf("%s",nome_arq);  fflush(stdin);
    fid = fopen(nome_arq,"rt");
    if (fid == NULL) {
        printf("impossível abrir o arquivo para leitura...");   // exit(0);
    }
    while (!feof(fid)) {
      fscanf(fid,"%c",&temp);
      i=0;
      while (temp != ';') {
            nome[i++] = temp;
                  fscanf(fid,"%c",&temp);
            if (feof(fid)) break; // para evitar de ficar lendo se o final do arquivo foi atingido
      }
      if (feof(fid)) break;
      nome[i] = '\0';
              fscanf(fid,"%d",&idade);
              printf ("\n%s %d",nome,idade);
    }
    fclose(fid);
    getchar(); }
```

Exercícios

1. Criar um programa para ler os dados (nomes e idades) do arquivo criado anteriormente e mostrar o resultado na tela

2. Ler 10 valores para um vetor. Guardar no arquivo texto “pares.txt” os valores pares do vetor e, no arquivo “impares.txt” os valores ímpares.
    1. Faça um menu para as opções: / gravar / ler pares/ ler ímpares /sair
    2. Sempre que o usuário escolher a opção ler ou gravar, não se deve fechar a aplicação, somente quando selecionar sair.

3. Reescreva os primeiro exercício utilizando arquivos binários

4. Faça um código em C que permita o usuário escolher fazer: leitura, gravação ou sair.

   O código deve gravar em arquivo dados sobre livros: código, nome do livro, nome do autor, ano de publicação e editora.

Respostas – ex1

```
char nome_arq[20], nome[20], temp, i;
    int idade;
    FILE *fid;

    printf("\nEntre com o nome do arquivo: ");
    scanf("%s",nome_arq);  fflush(stdin);
    fid = fopen(nome_arq,"rt");
    if (fid == NULL) {
        printf("impossível abrir o arquivo para leitura...");
        exit(0);
    }

    while (!feof(fid)) {
      fscanf(fid,"%c",&temp);
      i=0;
      while (temp != ';') {
           nome[i++] = temp;
                       fscanf(fid,"%c",&temp);
           if (feof(fid)) break; // para evitar de ficar lendo se o final do arquivo foi atingido
      }
      if (feof(fid)) break;
      nome[i] = '\0';
                    fscanf(fid,"%d",&idade);

                    printf ("\n%s %d",nome,idade);
    }
    fclose(fid);
```

Manipulação de Strings em C

Sabemos que C não foi feito para manipular strings!!!

- Em C não existe um tipo de dado string, no seu lugar é utilizado uma matriz de caracteres.
- Uma string é uma matriz tipo char que termina com '\0'.
- Por essa razão uma string deve conter uma posição a mais do que o número de caracteres que se deseja.
- Constantes strings são uma lista de caracteres que aparecem entre aspas, não sendo necessário colocar o '\0', que é colocado pelo compilador.

```
main()
{
static re[]=“lagarto”;
puts(re);
puts(&re[0]);
putchar('\n');
}
```

Função fgets

- Sintaxe: gets(nome_matriz);
- É utilizada para leitura de uma string através do dispositivo padrão, até que o ENTER seja pressionado.
- A função gets() não testa limites na matriz em que é chamada.

Ex:

```
main()
{
char str[80];
gets(str);
printf("%s",str);
}
```

Função fputs

- Sintaxe: puts(nome_do_vetor_de_caracteres);

- Escreve o seu argumento no dispositivo padrão de saída (vídeo), coloca um '\n‘ no final.
- Reconhece os códigos de barra invertida.

Ex:

```
main()
{
puts("mensagem");
}
```

Função strcopy

- Sintaxe: strcpy(destino,origem);
- Copia o conteúdo de uma string.

Ex:

```
main(){
char str[80];
strcpy(str,"alo");
puts(str);
}
```

Função strcat

- Sintaxe: strcat(string1,string2);

- Concatena duas strings. Não verifica tamanho.

Ex:

```
main()
{
char um[20],dois[10];
strcpy(um,"bom");
strcpy(dois," dia");
strcat(um,dois);
printf("%s\n",um);
}
```

Função strcomp

- Sintaxe: strcmp(s1,s2);

- Compara duas strings, se forem iguais devolve 0.

Ex:

```
main()
{
char s[80];
printf("Digite a senha:");
gets(s);
if (strcmp(s,"laranja"))
printf("senha inválida\n");
else
printf("senha ok!\n") ;
}
```

Exemplo- Comprimento da cadeia de caracteres.


```
#include <stdio.h>

int comprimento (char* s)
{
int i;
int n = 0; /* contador */
for (i=0; s[i] != '\0'; i++)
n++;
return n;
}

int main (void)
{
int tam;
char cidade[] = "Rio de Janeiro";
tam = comprimento(cidade);
printf("A string \"%s\" tem %d caracteres\n", cidade, tam);
return 0;
}
```

Exemplo

Escreva um programa para armazenar em um arquivo texto *nomes completos e telefones* que vão sendo inseridos, até que o nome "fim" seja inserido. O usuário deve digitar o nome, sobrenome e telefone. Você pode usar o strcat para concatenar o nome + sobrenome.

Deve ser permitido ao usuário poder ***ler o arquivo*** e poder ***gravar no arquivo***. Crie um menu para essas opções.

Exercício

1. Ler 10 valores para um vetor. Guardar no arquivo texto “pares.txt” os valores pares do vetor e, no arquivo “impares.txt” os valores ímpares.
    1. Faça um menu para as opções: / gravar / ler pares/ ler ímpares /sair
    2. Sempre que o usuário escolher a opção ler ou gravar, não se deve fechar a aplicação, somente quando selecionar sair.