Linguagem C: Arquivo-Texto

Prof. Paulo R. S. L. Coelho
paulo@facom.ufu.br

Faculdade de Computação
Universidade Federal de Uberlândia

GEQ007

Organização

Organização

Introdução

- Estruturas de dados manipuladas fora do ambiente do programa (memória principal) são conhecidas como **arquivos**.
- Um arquivo é armazenado em um dispositivo de memória secundária e pode ser lido ou escrito por um programa.

Arquivos em C I

- Um arquivo em C pode representar diversas coisas, como arquivos em disco, uma impressora, um teclado, ou qualquer dispositivo de entrada ou saída.
- Consideraremos apenas arquivos texto em disco.
- A linguagem C dá suporte à utilização de arquivos por meio da biblioteca `stdlib.h`.
- Esta biblioteca fornece várias funções para manipulação de arquivos e define alguns tipos de dados para serem usados especificamente com arquivos.

Arquivos em C II

- O principal tipo definido nessa biblioteca que será usado é o tipo `FILE`.
- Um variável do tipo `FILE` é capaz de identificar um arquivo no disco, direcionando-lhe todas as operações.
- Essas variáveis são declaradas da seguinte maneira:
  `FILE *arq;`

Organização

Abertura de Arquivo

- A função `fopen` abre um arquivo, retornando um referência para o arquivo aberto ou `NULL` caso ocorra algum erro.
- Seu protótipo é:

```
FILE *fopen(nome_do_arquivo, modo_de_abertura);
```

- O argumento `nome_do_arquivo` é o caminho do arquivo que se deseja abrir.
- O argumento `modo_de_abertura` representa como o arquivo será aberto. Veja a tabela:

Modo	Descrição
r	Abre arquivo texto somente para leitura
w	Cria arquivo texto somente para escrita
a	Anexa dados a um arquivo texto
r+	Abre arquivo texto para leitura e escrita
w+	Cria arquivo onde poderão ser realizadas leituras e escritas
a+	Anexa dados a um arquivo texto ou cria um para leitura e escrita

Fechamento de Arquivo

- Um arquivo aberto sempre deve ser fechado antes do fim do programa.
- A função que realiza essa tarefa é `fclose`.
- Seu protótipo é:

```
int fclose(FILE *arq);
```

- O argumento `arq` é a variável que representa o arquivo aberto (com `fopen`)
- O retorno dessa função é zero em caso de sucesso. Qualquer valor diferente de zero, significa erro.

Leitura e Escrita de Arquivo - I

- A função `fputc` escreve um caractere em um arquivo.
- Seu protótipo é:
  ```
  int fputc(char ch, FILE *arq);
  ```
- A função `fputs` escreve um cadeia de caracteres em um arquivo.
- Seu protótipo é:
  ```
  char *fputs(char *cadeia, FILE *arq);
  ```
- A função `fgetc` lê um caractere em um arquivo.
- Seu protótipo é:
  ```
  int fgetc(FILE *arq);
  ```
- A função `fgets` lê um cadeia de caracteres em um arquivo.
- Seu protótipo é:
  ```
  char *fgets(char *cadeia, int tam, FILE *arq);
  ```
  onde `tam` define o tamanho da cadeia que será lida: tam-1 ou fim de linha, o que ocorrer primeiro.

Exemplo

```
#include <stdio.h>
#include <stdlib.h>

int main(){
  FILE *arq;
  char nome1[20], nome2[20], linha[20];
  arq = fopen("teste.txt", "w");
  printf("entre nome1: "); scanf("%s", nome1);
  printf("entre nome2: "); scanf("%s", nome2);
  fputs(nome1, arq);
  fputs("\n", arq);
  fputs(nome2, arq);
  fclose(arq);
  arq = fopen("teste.txt", "r");
  printf("lendo do arquivo:\n");
  fgets(linha, 20, arq);
  printf("\tnome1: %s", linha);
  fgets(linha, 20, arq);
  printf("\tnome2: %s\n", linha);
  return 0;
}
```

Leitura e Escrita de Arquivo - II

- A função `fprintf` permite escrever em um arquivo da mesma forma que escrevemos na tela com o `printf`
- Seu protótipo é:

```
int fprintf(FILE *arq, char *formato, ...);
```

- A função `fscanf` lê informações de um arquivo da mesma forma que lemos do teclado com o `scanf`
- Seu protótipo é:

```
int fscanf(FILE *arq, char *formato, ...);
```

Exemplo

```
#include <stdio.h>
#include <stdlib.h>

int main(){
  FILE *arq;
  char nome1[20], nome2[20], linha[20];
  arq = fopen("teste.txt", "w");
  printf("entre nome1: "); scanf("%s", nome1);
  printf("entre nome2: "); scanf("%s", nome2);
  fprintf(arq, "%s\n%s", nome1, nome2);
  fclose(arq);
  arq = fopen("teste.txt", "r");
  printf("lendo do arquivo:\n");
  fscanf(arq, "%s", nome1);
  fscanf(arq, "%s", nome2);
  printf("nome1: %s\nnome2: %s\n", nome1, nome2);
  return 0;
}
```

Organização

Exercícios

1. Faça um programa que leia o nome e sobrenome de 30 alunos e armazene em um arquivo, de tal forma que o arquivo tenha um aluno por linha.
2. Faça um programa que leia um vetor de inteiros A de tamanho 20 e guarde seus valores em um arquivo, um por linha. Em seguida, reabra o arquivo e leia os elementos para o vetor B, verificando se os valores foram gravados corretamente.

Respostas I

1.
```c
#include <stdio.h>
#include <stdlib.h>

int main(){
  FILE *arq;
  char nome[20], sobrenome[20], i;

  arq = fopen("t1.txt", "w");
  printf("entre nome e sobrenome de 30 alunos : \n");
  for (i = 0; i < 30; i++) {
    printf("Aluno %d. Entre nome: ");
    scanf("%s", nome);
    printf("Aluno %d. Entre sobrenome: ");
    scanf("%s", sobrenome);
    fputs(nome, arq);
    fputs(" ", arq);
    fputs(sobrenome, arq);
    fputs("\n", arq);
  }
  fclose(arq);
  return 0;
}
```

Respostas II

2.
```
#include <stdio.h>
#include <stdlib.h>

int main(){
  FILE *arq;
  int A[20], B[20], i;
  char aux[10];

  arq = fopen("t2.txt", "w");
  printf("entre 20 numeros: \n");
  for (i = 0; i < 20; i++) {
    scanf("%d", &A[i]);
    fprintf(arq, "%d\n", A[i]);
  }
  fclose(arq);

  arq = fopen("t2.txt", "r");
  printf("\nlendo 20 numeros: \n");
  for (i = 0; i < 20; i++) {
    fscanf(arq, "%d", &B[i]);
    printf("%d ", B[i]);
  }
  fclose(arq);
```

Respostas III

```
  printf("\n");
  return 0;
}
```