unipampa

Universidade Federal do Pampa

Laboratório de Programação I

Assunto de Hoje: Manipulação de Arquivos Texto

Professor Luciano Brum
lucianobrum@unipampa.edu.br

Materiais de aula adaptados dos slides do professor Júlio Saraçol e do livro de Celes, Cerqueira e Rangel (2004).

Manipulação de Arquivos em C

- A finalidade desta apresentação é discutir formas para salvar e recuperar informações em arquivos.

- Será possível implementar funções para salvar e recuperar informações armazenadas em ED.

Manipulação de Arquivos em C

- Arquivos em disco representam um elemento de informação contido em um dispositivo de memória secundário.

- Dispositivos de memória primários e secundários diferem em vários aspectos.

Manipulação de Arquivos em C

- Eficiência.
- Persistência.
- Volatilidade.

Manipulação de Arquivos em C

- Muitas vezes diferentes pessoas e programas leem tais informações destes arquivos, pois estas são persistentes.

- Por isso, arquivos são identificados pelo nome e o diretório em que estão armazenados na unidade de disco.

Manipulação de Arquivos em C

- Por que arquivos?
  - Permitem armazenar grande quantidade de informação;
  - Persistência dos dados (disco);
  - Acesso aos dados poder ser não sequencial;
  - Acesso concorrente aos dados (mais de um programa pode usar os dados ao mesmo tempo).

Manipulação de Arquivos em C

- Um arquivo pode ser visto pelo SO de duas maneiras:
  - **Modo texto: sequência de caracteres**.
  - Modo binário: sequência de bytes.

Manipulação de Arquivos em C

- Qual a diferença entre arquivos binários e arquivos texto?

Manipulação de Arquivos em C

- A linguagem C disponibiliza serviços para ler/escrever informações em disco com suporte do S.O. Principais deles são:

  - Abertura de Arquivos.
  - Leitura do Arquivo.
  - Escrita no Arquivo.
  - Fechamento de Arquivo.

Abertura de Arquivos em C

- O primeiro passo para manipular um arquivo em C, é abri-lo.
- Durante a abertura do arquivo, a linguagem C necessita obter do sistema operacional algumas informações sobre este arquivo.
- Estas informações obtidas ficam armazenadas em um ponteiro.

Abertura de Arquivos em C

- A função básica para abertura de arquivos é fopen:

```
FILE *fopen (char *nome_do_arquivo, char *modo_de_abertura);
"char *nome_do_arquivo": local onde o arquivo se encontra ou será criado.
"char *modo": especifica: como o arquivo deve ser aberto.
```

- FILE é um tipo definido pela biblioteca padrão que representa uma abstração do arquivo.

- Quando abrimos um arqiuivo, a função retorna um ponteiro para o FILE.

- Todas operações receberão esse endereço como parâmetro de entrada.

Abertura de Arquivos em C

```
FILE *fp;
if((fp=fopen("entrada.txt","rt"))==NULL){
     printf("Erro na abertura do arquivo.\n");
     exit(1);
}
```

Modos de Abertura de Arquivos

Mode	Significado
"r"	Abre um arquivo texto para leitura
"w"	Cria um arquivo texto para escrita
"a"	Acrescenta em um arquivo texto
"rb"	Abre um arquivo binário para leitura
"wb"	Cria um arquivo binário para escrita
"ab"	Acrescenta em um arquivo binário
"r+"	Abre um arquivo texto para leitura/escrita
"w+"	Cria um arquivo texto para leitura/escrita
"a+"	Abre um arquivo texto para leitura/escrita
"rb+"	Abre um arquivo binário para leitura/escrita
"wb+"	Cria um arquivo binário para leitura/escrita
"ab+"	Abre um arquivo binário para leitura/escrita

Fonte: CELES, CERQUEIRA e RANGEL (2004).

Abertura e fechamento de Arquivos

```c
#include <stdio.h>
int main(void) {

FILE *arquivo; //cria o ponteiro do arquivo

//cria um arquivo de nome teste.txt, o "w" quer dizer escrita
if((arquivo=fopen("teste.txt", "w")) == NULL){
  printf("Permissão negada!");
  return 1;
}
else{
fclose(arquivo); //fecha o arquivo "teste.txt"
}
return 0;
}
```

Fechamento de Arquivos em C

- Por que devemos fechar o arquivo?
  - Ao fechar um arquivo, todo caractere que tenha permanecido no "buffer" é gravado.
  - O "buffer“ é uma região de memória que armazena temporariamente os caracteres a serem gravados em disco imediatamente.
  - Apenas quando o "buffer" está cheio é que seu conteúdo é escrito no disco.

Manipulação de Arquivos

- Existem 4 grupos de funções:

1. Gravar e ler 1 caractere por vez: funções fputc() e fgetc().

2. Gravar e ler 1 linha por vez: funções fputs() e fgets().

3. Gravar e ler dados formatados: funções fprintf() e fscanf().

4. Gravar e ler blocos de bytes: funções fwrite() e fread().

fgetc

- Captura o próximo caractere do arquivo (e o cursor avança para o próximo caractere).

- `int fgetc(FILE* fp);`

- O valor retornado é o código do caractere lido.

- Se o fim do arquivo for alcançado, é retornada a constante EOF (*end of file*).

fgetc

```c
#include <stdio.h>
int main(void) {

char texto; //cria a variável do tipo char, ou seja aceita somente 1 caracter
FILE *arquivo; //cria o ponteiro do arquivo
//abre um arquivo de nome teste.txt, o "r" quer dizer leitura
if((arquivo=fopen("teste.txt", "r")) == NULL){
  printf("Permissão negada!");
  return 1;
}
else{
//o fgetc lê o arquivo que foi aberto e grava o caracter na variável texto
texto = fgetc(arquivo);
printf("Conteúdo do arquivo teste.txt: %c", texto);
fclose(arquivo); //fecha o arquivo "teste.txt"
}
return 0;
}
```

fputc

- Escreve um caractere no arquivo (e o cursor avança para o próximo caractere).

- `int fputc(FILE* fp);`

- O valor retornado é o código do próprio caractere escrito.

- É retornada a constante EOF (*end of file*) se ocorrer algum erro.

fputc

```c
#include <stdio.h>
int main(void) {

char texto; //cria a variável do tipo char, ou seja aceita somente 1 caracter
FILE *arquivo; //cria o ponteiro do arquivo

//cria um arquivo de nome teste.txt, o "w" quer dizer escrita
if((arquivo=fopen("teste.txt", "w")) == NULL){
  printf("Permissão negada!");
    return 1;
}
else{
printf("Digite um caracter: ");
texto=getchar(); //grava na variável o caracter digitado

//escreve o valor da variável texto no arquivo "teste.txt" que foi aberto
fputc(texto,arquivo);
fclose(arquivo); //fecha o arquivo "teste.txt"
}
return 0;
}
```

fgets

- Captura uma sequência de caracteres do arquivo, até que encontre um ‘\n’.
- `char* fgets(char* s, int tam, FILE* fp);`
- O primeiro parâmetro é a variável que vai armazenar a sequência de caracteres.
- O segundo parâmetro informa o n° máximo de caracteres a serem lidos.
- Como o final da string resultante possui o ‘\0’, no máximo, tam-1 caracteres serão lidos.

fgets

```c
#include <stdio.h>
int main(void) {

char texto[100]; //cria a variável do tipo string aceitando 99 caracteres
FILE *arquivo; //cria o ponteiro do arquivo

//abre um arquivo de nome teste.txt, o "r" quer dizer leitura
if((arquivo=fopen("teste.txt", "r")) == NULL){
  printf("Permissão negada!");
  return 1;
}
else{
//o fgets lê o arquivo que foi aberto até 99 posições
//e grava a string na variável texto
fgets(texto,99,arquivo);
printf("Conteúdo do arquivo: %s", texto);
fclose(arquivo); //fecha o arquivo
}

return 0;
}
```

fputs

- Escreve uma sequência de caracteres no arquivo de saída.
- `char* fputs(char* s, FILE* fp);`
- O primeiro parâmetro é a variável que vai armazenar a sequência de caracteres.

fputs

```c
#include <stdio.h>
int main(void) {

char texto[100]; //cria a variável do tipo string aceitando 99 caracteres
FILE *arquivo; //cria o ponteiro do arquivo
//cria um arquivo "teste.txt", o "w" quer dizer escrita
if((arquivo=fopen("teste.txt", "w")) == NULL){
  printf("Permissão negada!");
  return 1;
}
else{
printf("Digite uma string: ");
gets(texto);
//o fputs escreve o valor da variável texto no arquivo que foi aberto
fputs(texto,arquivo);
fclose(arquivo); //fecha o arquivo
printf("\n Arquivo criado com sucesso!");
}

return 0;
}
```

fprintf

- Escreve dados formatados em um arquivo de saída.
- `int* fprintf(FILE* fp, char* formato, ...);`
- É similar a função printf, porém a saída é um arquivo.
- O valor de retorno é o número de bytes escritos no arquivo.

fprintf

```
5   int main(void){
6
7       char nome[50], telefone[15], endereco[200];
8       FILE *arquivo;
9
10      printf("Digite seu nome: ");
11      gets(nome);
12      printf("Digite seu telefone: ");
13      gets(telefone);
14      printf("Digite seu endereço: ");
15      gets(endereco);
16
17      if((arquivo=fopen("string_formatada.txt", "w"))==NULL){
18          printf("Erro no arquivo.");
19          return 1;
20      }
21      else{
22      fprintf(arquivo,"Nome: %s\n",nome);
23      fprintf(arquivo,"Telefone: %s\n",telefone);
24      fprintf(arquivo,"Endereço: %s",endereco);
25      fclose(arquivo);
26
27      printf("Arquivo criado com sucesso. PRESSIONE ENTER PARA SAIR");
28      }
29  return 0;
30  }
```

fscanf

- Lê dados formatados de um arquivo de entrada.
- `int*fscanf(FILE* fp, char* formato, ...);`
- É similar a função scanf, porém a entrada de dados é do arquivo.
- O valor de retorno é o número de bytes escritos no arquivo.
- Cada valor lido é transferido para a memória e o cursor avança apontando para o próximo dado do arquivo.

fscanf

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

int main(void){

    char texto[250];
    FILE *arquivo;

    if((arquivo=fopen("frutas.txt", "r"))==NULL){
        printf("Erro no arquivo.");
        return 1;
    }
    else{
  //lê e imprime cada string até chegar no fim do arquivo
  while((fscanf(arquivo,"%s\n", &texto)!=EOF)){
      printf("%s\n", texto);
      }

    fclose(arquivo);

    printf("\nArquivo lido com sucesso. PRESSIONE ENTER PARA SAIR");
    }

return 0;
}
```

Exercício de Fixação

1. Crie um programa que escreva dados em um arquivo teste.txt. Faça isso por caractere, por linha ou por dado formatado. Crie um menu para o usuário escolher a forma utilizada de escrita.
2. Crie um programa que leia os dados do exercício anterior por caractere, por linha e por dado formatado. Crie um menu para o usuário escolher. Mostrar na tela a informação lida.
3. Crie um programa que copie todos os dados do teste.txt para o teste_copia.txt.
4. Crie um programa que una em um arquivo teste_final.txt o conteúdo do teste.txt e do teste2.txt. Preencha estes arquivos antes de chamar tal função.
5. Ler números inteiros e armazená-los no arquivo (numeros.txt). Ler o arquivo (numeros.txt) e armazenar no arquivo (pares.txt) os valores pares e no (impares.txt) os ímpares.

Atividade Semipresencial 4

1. Faça um programa de cadastro de alunos. A estrutura aluno é composta por:
   - Nome.
   - E-mail.
   - Matrícula.
   - Telefone.
2. No programa principal, crie um menu com as seguintes opções:
   a) Cadastrar aluno – Insere um aluno novo no arquivo.
   b) Excluir aluno – Excluir um aluno do arquivo por MATRÍCULA (Método: Copiar todos alunos, exceto o que deve ser removido, para um novo arquivo, excluir o antigo e renomear o novo).

Atividade Semipresencial 4

c) Imprimir alunos – Mostrar todos alunos do arquivo.

d) Opção para encerrar o programa.

3. Requisitos da aplicação:

a) Utilizar modularização nas soluções (funções/procedimentos).

b) Utilize comentários para auxiliar no entendimento do código.

c) Não usar variáveis globais.

d) Utilizar arquivos texto para a solução.

e) O programa só deve encerrar quando o arquivo não puder ser aberto ou o usuário digitar a opção d) ou 4.

Funções Adicionais

- int remove(file_name);
  - Apaga o arquivo mencionado no campo “file_name”.
  - Se o arquivo for removido com sucesso, a função retorna 0.
- int rename (const char *caminho_antigo, const char *caminho_novo);
  - Renomeia o arquivo ou diretório especificado pelo parâmetro *caminho_antigo* para o nome *caminho_novo*.
  - Se o *caminho_antigo* e o *caminho_novo* estão localizados em diretórios diferentes, o arquivo é movido para o novo diretório, se suportado pelo sistema.
  - Se o arquivo foi renomeado com sucesso, retorna 0.

Funções Adicionais

- void rewind (FILE * fluxo);
  - Volta o indicador de posição do fluxo para a posição inicial. Isto é, o começo do arquivo.

unipampa

Universidade Federal do Pampa

Dúvidas ?