Aulas 25 - Arquivos*

Delano Beder

*Baseado no livro: Linguagem C Completa e Descomplicada, de André Backes

Arquivos

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

Tipos de Arquivos

- A linguagem C trabalha com dois tipos de arquivos:
  - texto
  - binários
- Arquivo texto
  - Pode ser alterado por meio de um editor de textos.
  - Armazena caracteres (8 bits - 1 byte)
    - Ex.: Um número inteiro de 32 bits com 8 dígitos ocupará 64 bits no arquivo (8 bits por dígito).

Tipos de Arquivos

- Arquivo binário
  - armazena uma sequência de bits.
    - Ex: arquivos executáveis, arquivos compactados, arquivos de registros, etc.
  - os dados são gravados na forma binária (do mesmo modo que estão na memória).
    - Ex.: um número inteiro de 32 bits com 8 dígitos ocupará 32 bits no arquivo.

Manipulando arquivos em C

- A linguagem C possui uma série de funções para manipulação de arquivos, cujos protótipos estão reunidos na biblioteca padrão de entrada e saída, **stdio.h**.

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

Manipulando arquivos em C

- Funções para manipulação de arquivos na linguagem C:
  - Abrir/Fechar arquivos e
  - Ler/Escrever caracteres/bytes

- Não possui funções que automaticamente leiam todas as informações de um arquivo. É tarefa do programador criar a funcionalidade que lerá um arquivo de uma maneira específica.

Manipulando arquivos em C

- Todas as funções de manipulação de arquivos trabalham com o conceito de "ponteiro de arquivo". Podemos declarar um ponteiro de arquivo da seguinte maneira:

```
FILE *p;
```

- **p** é o ponteiro para arquivos que nos permitirá manipular arquivos no C.

Abrindo um arquivo

- Para a abertura de um arquivo, usa-se a função **fopen**

```
FILE *fopen(char *nome_arquivo,char *modo);
```

- O parâmetro **nome_arquivo** determina qual arquivo deverá ser aberto, sendo que o mesmo deve ser válido no sistema operacional que estiver sendo utilizado.

- Exemplo:

```
FILE* f;
f = fopen_e_teste("teste.txt", "w");
```

Abrindo um arquivo

- No parâmetro **nome_arquivo** pode-se trabalhar com caminhos absolutos ou relativos.
  - **Caminho absoluto**: descrição de um caminho desde o diretório raiz.
    - **C:\\Projetos\\dados.txt (windows)**
    - **/home/joice/Projetos/dados.txt (Linux)**

  - **Caminho relativo**: descrição de um caminho desde o diretório corrente (onde o programa está salvo)
    - **arq.txt**
    - **./Projetos/dados.txt(linux)**

Modos de abertura de um arquivo Texto

Modo	Arquivo	Função
"r"	Texto	Leitura. Arquivo deve existir.
"w"	Texto	Escrita. Cria o arquivo se não houver. Apaga o conteúdo anterior se ele existir.
"a"	Texto	Escrita. Os dados serão adicionados no fim do arquivo ("append").
"r+"	Texto	Leitura/Escrita. O arquivo deve existir e pode ser modificado.
"w+"	Texto	Leitura/Escrita. Cria o arquivo se não houver. Apaga o conteúdo anterior se ele existir.
"a+"	Texto	Leitura/Escrita. Os dados serão adicionados no fim do arquivo ("append").

Modos de abertura de um arquivo binário

Modo	Arquivo	Função
"rb"	Binário	Leitura. Arquivo deve existir.
"wb"	Binário	Escrita. Cria o arquivo se não houver. Apaga o conteúdo anterior se ele existir.
"ab"	Binário	Escrita. Os dados serão adicionados no fim do arquivo ("append").
"r+b"	Binário	Leitura/Escrita. O arquivo deve existir e pode ser modificado.
"w+b"	Binário	Leitura/Escrita. Cria o arquivo se não houver. Apaga o conteúdo anterior se ele existir.
"a+b"	Binário	Leitura/Escrita. Os dados serão adicionados no fim do arquivo ("append").

Abrindo um arquivo

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

FILE* fopen_e_teste(char *caminho, char* modo) {
    FILE *f;
    f = fopen(caminho, modo);
    if (f == NULL) {
        perror("Erro ao tentar abrir o arquivo.\n");
        exit(1);
    }
    return f;
}

int main() {
    FILE* f;
    f = fopen_e_teste("teste.txt", "w");
    return 0;
}
```

No caso de erro,fopen() retorna um ponteiro nulo(**NULL**).

A condição **f==NULL** testa se o arquivo foi aberto com sucesso.

Erro ao abrir um arquivo

- Caso o arquivo não tenha sido aberto com sucesso
  - Provavelmente o programa não poderá continuar a executar;
  - Nesse caso, utilizamos a função **exit(),** presente na biblioteca **stdlib.h**, para abortar o programa

```
void exit(int codigo_de_retorno);
```

Erro ao abrir um arquivo

- A função **exit()** pode ser chamada de qualquer ponto no programa e faz com que o programa termine e retorne, para o sistema operacional, o **código_de_retorno**.
- A convenção mais usada é que um programa retorne
  - **zero** no caso de um término normal
  - um número **diferente de zero**, no caso de ter ocorrido um problema
- A função **perror()** exibe uma mensagem explicativa

Posição do arquivo

- Ao se trabalhar com arquivos, existe uma espécie de posição onde estamos dentro do arquivo. É nessa posição onde será lido ou escrito o próximo caractere.
    - Quando utilizando o acesso sequencial, raramente é necessário modificar essa posição.
    - Quando lemos um caractere, a posição no arquivo é automaticamente atualizada.

Fechando um arquivo

- Sempre que terminamos de usar um arquivo que abrimos, devemos fechá-lo. Para isso usa-se a função **fclose()**

- O ponteiro **fp** passado à função **fclose()** determina o arquivo a ser fechado. A função retorna zero no caso de sucesso.

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

Fechando um arquivo

- 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. Apenas quando o "buffer" está cheio é que seu conteúdo é escrito no disco.

Fechando um arquivo

- Por que utilizar um “buffer”?
  - Para ler e escrever arquivos no disco temos que posicionar a cabeça de gravação em um ponto específico do disco.
  - Se tivéssemos que fazer isso para cada caractere lido/escrito, a leitura/escrita de um arquivo seria uma operação muita lenta.
  - Assim a gravação só é realizada quando há um volume razoável de informações a serem gravadas ou quando o arquivo for fechado.

- A função **exit()** fecha todos os arquivos que um programa tiver aberto.

Escrita/Leitura em Arquivos

- Uma vez aberto um arquivo, podemos ler ou escrever nele.
- Para tanto, a linguagem C conta com uma série de funções de leitura/escrita que variam de funcionalidade para atender as diversas aplicações.

Escrita/Leitura por fluxo padrão

- As funções de fluxos padrão permitem ao programador ler e escrever em arquivos da maneira padrão, ou seja, da mesma forma com que lemos e escrevemos na tela.
- As funções **fprintf** e **fscanf** funcionam de maneiras semelhantes a **printf** e **scanf**, respectivamente.
- A diferença é que elas direcionam os dados para arquivos.

Escrita/Leitura por fluxo padrão

- **int fprintf** (FILE * arq, char* formato,...)

```
printf("Total = %d",x);//escreve na tela
fprintf(fp,"Total = %d",x);//grava no arquivo fp
```

- **int fscanf**(FILE *arq, char *string_formatada,...)

```
scanf("%d",&x);//lê do teclado
fscanf(fp,"%d",&x);//lê do arquivo fp
```

Escrita/Leitura por fluxo padrão

- Atenção
  - Embora **fprintf** e **fscanf** sejam mais fáceis de ler/escrever dados em arquivos, nem sempre elas são as escolhas mais apropriadas.
  - Como os dados são escritos em ASCII e formatados como apareceriam em tela, um tempo extra é perdido.

Escrita/Leitura Por fluxo padrão

- Se não houver mais caracteres no arquivo, a função **fscanf** devolve a constante **EOF** (*end of file*), que está definida na biblioteca **stdio.h**. Em muitos computadores o valor de **EOF** é **-1**.

Escrita/Leitura por fluxo padrão

- Exemplo da função **fprintf**

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

FILE* fopen_e_teste(char *caminho, char* modo) {
    FILE *f;
    f = fopen(caminho, modo);
    if (f == NULL) {
        perror("Erro ao tentar abrir o arquivo.\n");
        exit(1);
    }
    return f;
}

int main() {
    FILE* f;
    char nome[20] = "Ricardo";
    int idade = 30;
    float altura = 1.74;
    f = fopen_e_teste("teste.txt", "w");
    fprintf(f, "Nome: %s\n", nome);
    fprintf(f, "Idade: %d\n", idade);
    fprintf(f, "Altura: %4.2f\n", altura);
    fclose(f);
    return 0;
}
```

Escrita/Leitura por fluxo padrão

- Exemplo da função **fscanf**

```c
#include <stdlib.h>

FILE* fopen_e_teste(char *caminho, char* modo) {
    FILE *f;
    f = fopen(caminho, modo);
    if (f == NULL) {
        perror("Erro ao tentar abrir o arquivo.\n");
        exit(1);
    }
    return f;
}

int main() {
    FILE* f;
    char nome[20], texto[20];
    int idade;
    float altura;
    f = fopen_e_teste("teste.txt", "r");
    fscanf(f, "%s %s", texto, nome);
    printf("%s %s\n", texto, nome);
    fscanf(f, "%s %d", texto, &idade);
    printf("%s %d\n", texto, idade);
    fscanf(f, "%s %f", texto, &altura);
    printf("%s %4.2f\n", texto, altura);
    fclose(f);
    return 0;
}
```

Exercício

Considere arquivos contendo dados inteiros, um por linha, podendo haver valores positivos ou negativos. O arquivo possui, em sua última linha, um valor sentinela igual a -9999999, que não deve ser considerado no processamento. Deseja-se saber a soma de todos os valores contidos no arquivo. Portanto, escreva um programa em C que leia o nome de um arquivo que siga essa especificação e apresente a soma de todos os valores.

EXERCÍCIOS

Faça um programa que imprima na tela (saída padrão) o conteúdo de um arquivo texto.

Faça um programa que copie o conteúdo de um arquivo texto para outro arquivo texto.

Escrita/Leitura de Caracteres

- A maneira mais fácil de se trabalhar com um arquivo texto é a leitura/escrita de um único caractere.
- A função mais básica de entrada de dados é a função **fputc** (*put character*).

```
int fputc (int ch, FILE *fp);
```

- Cada invocação dessa função grava um único caractere **ch** no arquivo especificado por **fp**.

Escrita/Leitura de Caracteres

- Exemplo da função **fputc**

```c
int main() {
    FILE* f;
    char string[100];
    int i, tam;
    f = fopen_e_teste("saida.txt", "w");

    printf("Entre com a string a ser gravada no arquivo: \n");
    fgets(string, 100, stdin);
    tam = strlen(string);
    for (i = 0; i < tam; i++) {
        fputc(string[i], f);
    }
    fclose(f);
    return 0;
}
```

Escrita/Leitura de Caracteres

- Da mesma maneira que gravamos um único caractere no arquivo, também podemos ler um único caractere.
- A função correspondente de leitura de caracteres é **fgetc** (*get character*).

```
int fgetc(FILE *fp);
```

Escrita/Leitura de Caracteres

- Cada chamada da função **fgetc** lê um único caractere do arquivo especificado
  - Se **fp** aponta para um arquivo, então **fgetc(fp)** lê o caractere atual no arquivo e se posiciona para ler o próximo caractere do arquivo.

```
char c;
c = fgetc(fp);
```

Escrita/Leitura de Caracteres

- Exemplo da função **fgetc**

Sugestão de exercício:

Cópia de arquivos (usando fgetc e fputc)

```c
int main() {
    FILE* f;
    int i;
    char caractere;
    f = fopen_e_teste("entrada.txt", "r");

    for (i = 0; i < 5; i++) {
        caractere = fgetc(f);
        printf("%c", caractere);
    }

    fclose(f);
    return 0;
}
```

Escrita de Strings

- **fputs()**

```
int fputs(char *str, FILE *fp);
```

- Retorno da função
  - Se o texto for escrito com sucesso um valor inteiro diferente de zero é retornado.
  - Se houver erro na escrita, o valor **EOF** é retornado.

Escrita/Leitura de Strings

- Exemplo da função **fputs**

```c
int main() {
    FILE* f;
    char str[20] = "Hello World";
    int result;
    f = fopen_e_teste("saida.txt", "w");

    result = fputs(str, f);

    if (result == EOF) {
        printf("Problema na gravação do arquivo\n");
    }
    fclose(f);
    return 0;
}
```

Leitura de Strings

- **fgets()**

```
char *fgets(char *str, int tamanho,FILE *fp);
```

- A função **fgets** recebe 3 parâmetros
  - **str**: onde a string lida será armazenada, **str**;
  - **Tamanho**: o número máximo de caracteres a serem lidos;
  - **fp**: ponteiro que está associado ao arquivo de onde a string será lida.

- E retorna
  - NULL em caso de erro ou fim do arquivo;
  - O ponteiro para o primeiro caractere recuperado em **str**.

Escrita/Leitura de Strings

- Funcionamento da função **fgets**
  - A função lê a string até que um caractere de nova linha seja lido ou *tamanho-1* caracteres tenham sido lidos.
  - Se o caractere de nova linha ('\n') for lido, ele fará parte da string,
  - A string resultante sempre terminará com '\0' (por isto somente *tamanho-1* caracteres, no máximo, serão lidos).
  - Se ocorrer algum erro, a função devolverá um ponteiro nulo em **str**.

Escrita/Leitura de Strings

- Exemplo da função **fgets**

```c
int main() {
    FILE* f;
    char str[20];
    char* result;
    f = fopen_e_teste("entrada.txt", "r");

    result = fgets(str, 13, f);

    if (result == NULL) {
        printf("Problema na leitura do arquivo\n");
    } else {
        printf("%s", str);
    }

    fclose(f);

    return 0;
}
```

Exercícios

- Faça um procedimento/função em C que recebe como parâmetros: (a) o nome de um arquivo texto de entrada e (b) o nome de um arquivo texto de saída. Cada linha do arquivo de entrada possui no máximo 80 caracteres. No arquivo de saída cada linha deve ser escrita no formato dançante. Uma sentença é chamada de dançante se sua primeira letra for maiúscula e cada letra subsequente for o oposto da letra anterior. Faça um programa para testar o procedimento/função. **Dica:** usar as funções isalpha(), toupper(), tolower() da biblioteca ctype.h

Exemplo: Arquivo de Entrada	Exemplo: Arquivo de Saída
Hoje e dia de aula de CAP Manipulacao de arquivos eh muito facil	HoJe E dIa De AuLa De CaP MaNiPuLaCaO dE aRqUiVoS eH mUiTo FaCiL

Exercícios

- Faça um procedimento/função em C que recebe como parâmetros: (a) o nome de um arquivo texto de entrada, (b) o nome de um arquivo texto de saída e (c) um valor inteiro k (0 a 25). Cada linha do arquivo de entrada possui no máximo 80 caracteres. Cada letra (‘A’-‘Z’), presente no arquivo de entrada, deve ser copiada para o arquivo de saída (encriptada pela **Cifra de César)**. Os demais caracteres são copiados sem nenhuma encriptação. Faça um programa para testar o procedimento/função. Considere que todas as letras são maiúsculas.

Exemplo: Arquivo de Entrada (k = 2)	Exemplo: Arquivo de Saída (k = 2)
CONSTRUCAO DE ALGORITMOS E PROGRAMACAO UNIVERSIDADE FEDERAL DE SAO CARLOS ABCDEFGHIJKLMNOPQRSTUVWXYZ	EQPUVTWECQ FG CNIQTKVOQU G RTQITCOCECQ WPKXGTUKFCFG HGFGTCN FG UCQ ECTNQU CDEFGHIJKLMNOPQRSTUVWXYZAB