Departamento
de Ciência da
COMPUTAÇÃO

UDESC
Joinville

Linguagem C

arquivos

André Tavares da Silva

andre.silva@udesc.br

Arquivos

- Em C, para poder trabalhar em um arquivo, precisamos abri-lo, associando-o a uma variável do tipo FILE*.
- FILE é uma estrutura de dados que varia de acordo com o Sistema Operacional (por enquanto não precisamos nos preocupar como funciona).

```
FILE *entrada;
FILE *saida;
```

Abrindo um arquivo

- Abrir um arquivo em C significa criar uma stream e conectá-la ao arquivo em disco.
- A associação entre variável e arquivo é feita através da função **fopen**.

```
entrada = fopen("arquivo.in", "rt");
saida = fopen("arquivo.out", "wb");
```

- Essa função precisa de dois parâmetros. O primeiro é o nome do arquivo em disco e o informa como pretendemos abrir o arquivo (leitura, escrita, binário, texto,...).
- Obs.: ao abrir um arquivo para gravação informando um nome de arquivo já existente apagará o arquivo antigo.

Abrindo um arquivo

- Caso o arquivo possa ser aberto, a função retorna um ponteiro para FILE;
- Caso contrário, retorna NULL (é preciso testar!);
- Alguns fatores que levam a erro na abertura:
  - Arquivo inexistente (no caso de leitura);
  - Espaço insuficiente em disco (para gravação);
  - S.O. não dá permissão de acesso ao arquivo.

Abrindo um arquivo

```
FILE *entrada = fopen("arquivo.in", "rt");
if(entrada == NULL) {
    printf("Erro na abertura do arquivo de entrada!");
    return 1;
}

FILE *saida;
if((saida = fopen("arquivo.out", "wb")) == NULL) {
    printf("Erro na abertura do arquivo de saída!");
    return 2;
}
```

fopen – opções de abertura

r	Abre um arquivo texto para leitura. O arquivo deve existir antes de ser aberto.
w	Abre um arquivo texto para gravação. Se o arquivo não existir, ele será criado. Se já existir, o conteúdo anterior será destruído.
a	Abre um arquivo texto para gravação. Os dados serão adicionados no fim do arquivo ("append"), se ele já existir, ou um novo arquivo será criado, no caso de arquivo não existente anteriormente.
rb	Abre um arquivo binário para leitura. Igual ao modo "r" anterior, só que o arquivo é binário.
wb	Cria um arquivo binário para escrita, como no modo "w" anterior, só que o arquivo é binário.
ab	Acrescenta dados binários no fim do arquivo, como no modo "a" anterior, só que o arquivo é binário.
r+	Abre um arquivo texto para leitura e gravação. O arquivo deve existir e pode ser modificado.
w+	Cria um arquivo texto para leitura e gravação. Se o arquivo existir, o conteúdo anterior será destruído. Se não existir, será criado.
a+	Abre um arquivo texto para gravação e leitura. Os dados serão adicionados no fim do arquivo se já existir, ou um novo arquivo será criado, no caso de arquivo não existente anteriormente.
r+b	Abre um arquivo binário para leitura e escrita. O mesmo que "r+" acima, mas para binário.
w+b	Cria um arquivo binário para leitura e escrita. O mesmo que "w+" acima, mas para binário.
a+b	Acrescenta dados ou cria uma arquivo binário para leitura e escrita. O mesmo que "a+" acima, só que o arquivo é binário.

Fechando um arquivo

- A função **fclose** determina que o arquivo deve ser fechado.
- Ela recebe como parâmetro a variável associada ao arquivo que se deseja fechar.
- Ela retorna zero no caso de sucesso.

```
FILE *arquivo = fopen("meu_arquivo", "rt");

. . .

fclose(arquivo);
```

Lendo dados de um arquivo texto

- O comando **fscanf** lê um conjunto de caracteres de um arquivo, funcionando como a função scanf. Neste último os dados são lidos da entrada padrão (normalmente o teclado) e não de um arquivo.

Protótipo: int fscanf (FILE *fp, const char *format,...);

Parâmetro 1: "FILE *fp" - o arquivo a ser lido.

Parâmetro 2: "const char *format" - qual a variável que receberá os dados do arquivo.

Parâmetros 3 e demais: endereço das variáveis a serem “lidas”;

Lendo dados de um arquivo texto

- O comando **fscanf** lê um conjunto de caracteres de um arquivo, funcionando como a função scanf. Neste último os dados são lidos da entrada padrão (normalmente o teclado) e não de um arquivo.

```
int a, b;
float x, y;
FILE *arq = fopen("arquivo.in", "rt");

if(arq) {
   fscanf(arq, "%d %d %f %f", &a, &b, &x, &y);
   ...
}
```

Gravando dados em um arquivo texto

- O comando **fprintf** escreve um conjunto de caracteres em um arquivo, como a **printf**. Neste último os dados são gravados na saída padrão (normalmente o monitor) e não em um arquivo.

Protótipo: int fprintf (FILE *fp, const char *format,...);

Parâmetro 1: "FILE *fp" - o arquivo a ser lido.

Parâmetro 2: "const char *format" – formato a ser escrito.

Parâmetros 3 e demais: endereço das variáveis a serem escritas;

Gravando dados em um arquivo texto

- O comando **fprintf** escreve um conjunto de caracteres em um arquivo, como a **printf**. Neste último os dados são gravados na saída padrão (normalmente o monitor) e não em um arquivo.

```
int a=1, b=2;
float x=0.0, y=3.14;
FILE *arq = fopen("arquivo.out", "wt");

if(arq) {
   fprintf(arq, "%d %d %f %f\n", a, b, x, y);
   ...
}
```

Lendo dados de um arquivo texto

- O comando **fgets** lê uma linha de texto em um arquivo. Lê caractere a caractere até encontrar um final de linha (\n) ou final do arquivo (EOF). Coloca o conteúdo lido em uma *string*. Precisa ser informado o tamanho máximo de caracteres (tamanho máximo do *string*).

Protótipo: char *fgets(char *s, int size, FILE *stream);

Parâmetro 1: string que recebe uma linha do arquivo;

Parâmetro 2: tamanho máximo de caracteres a serem lidos.

Parâmetros 3: "FILE *fp" - o arquivo a ser lido.

Lendo dados de um arquivo texto

- O comando **fgets** lê uma linha de texto em um arquivo. Lê caractere a caractere até encontrar um final de linha (\n) ou final do arquivo (EOF). Coloca o conteúdo lido em uma string. Precisa ser informado o tamanho máximo de caracteres (tamanho máximo do string).

```
int str[100];
FILE *arq = fopen("arquivo.in", "rt");

if(arq) {
   for(int i=0;i<10;i++)
      fgets(str, 100, arq);
   ...
}
```

Gravando dados de um arquivo texto

- O comando **fputs** grava uma linha de texto em um arquivo. Coloca o conteúdo de uma *string* gravando em um arquivo caractere a caractere até o final (\0) da *string* que é substituído por uma quebra de linha (\n).

Protótipo: int *fputs(char *s, FILE *stream);

Parâmetro 1: string que será gravado no arquivo;

Parâmetro 2: "FILE *fp" - o arquivo onde o conteúdo será gravado.

Gravando dados de um arquivo texto

- O comando **fputs** grava uma linha de texto em um arquivo. Coloca o conteúdo de uma *string* gravando em um arquivo caractere a caractere até o final (\0) da *string* que é substituído por uma quebra de linha (\n).

```
FILE *arq = fopen("arquivo.out", "wt");

if(arq) {
   fputs("Esta linha será gravada no arquivo.", arq);
   ...
}
```

Outros comandos

- O comando **fgetc** lê um caractere de um arquivo e aponta para o próximo caractere.
- O comando **fputc** grava um caractere no final de um arquivo texto.
- O comando **feof** verifica se a leitura de dados do arquivo apontam para o final do arquivo.

```
int fgetc(FILE *fp);

int fputc(char ch, FILE *fp);

int feof(FILE *fp);
```

Exemplo

```
int cont = 0;
FILE *f = fopen( "exemplo1.txt", "rt" );
if( f == NULL) {
   printf("Erro no arquivo!\n");
   return -1;
}
while( !feof(f) ) {
   cont++;
   fgetc( f );
}
fclose( f );
printf("Arquivo possui %d caracteres.\n", cont);
```

Exercício

- Leia 10 números inteiros de um arquivo texto, ordene os mesmos e grave o resultado em um outro arquivo.
- Leia 10 números inteiros de um arquivo texto, ordene os mesmos e grave o resultado no mesmo arquivo.
- Leia 10 números inteiros de um arquivo texto, ordene os mesmos e grave o resultado no final do arquivo lido.