UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

INF101202
Algoritmos e Programação
Modalidade Ead – Turma H

Material de apoio: introdução a arquivos

Informática UFRGS

.inf INSTITUTO DE INFORMÁTICA UFRGS

Arquivo:

1. É uma fonte de dados para o programa:- arquivo de entrada (*input*);
2. é um destino de dados de um programa: - arquivo de saída (*ouput*).

É um conjunto de *bytes* colocados um após outro armazenados em um dispositivo de memória auxiliar.

Arquivos: importância

Permitem o armazenamento permanente da informação, isto é, após a execução de um programa que cria e manipula um arquivo, todo o dado ali gravado se torna permanente,bem além do tempo de execução do programa.

Podem ser armazenados em dispositivos de memória auxiliar (discos, CDs, *pen drives*, DVDs).

Podem armazenar mais dados que a memória principal.

Arquivo:

Pode-se adotar a idéia que todo o processamento em arquivos é realizado em um conjunto sequencial de caracteres (conjunto de *bytes*) sem qualquer estrutura interna.

A esse conjunto é dado o nome de fluxo de dados (ou *stream*).

Pode-se pensar, que ao se referenciar a arquivos, está-se pensando em fluxo de dados (ou *streams*),

ou,

em um conjunto de dados existentes em um suporte como disco, *pen drive*, etc.

Conceito de fluxo de dados em C

✓ Conjuntos sequenciais de *bytes*, sem qualquer tipo de estrutura interna.

✓ São vistos por programas ou comandos como um conjunto ordenado de caracteres.

✓ O caractere *New Line (nova linha, ou seja, \n)* também é incluído em um fluxo de dados na forma de um *byte*, ou seja, de um caractere como outro qualquer.

✓No *C*, tanto a entrada como a saída de dados acontecem através do processamento de *fluxos de dados*, independentemente do periférico utilizado (*device independent*).

✓Os fluxos de dados são armazenados em dispositivos de memória auxiliar de forma permanente, a partir do uso de periféricos que aceitam tanto a leitura como a gravação de dados.

Arquivos:

✓No C, **arquivos** podem ser definidos como repositórios permanentes de **fluxos de dados**.

✓As operações de entrada ou saída de dados em arquivos (aí incluídos também os arquivos padrão: *stdin* - entrada e *stdout* - saída) acontecem apoiadas pelo armazenamento temporário dos **fluxos de dados** em áreas de memória intermediárias (*buffers*).

Áreas intermediárias (*buffers*) para acesso a arquivos

- Os dados que fluem do mundo externo para arquivos, ou de arquivos para o mundo externo, antes de serem gravados em dispositivos de memória auxiliar ou apresentados na tela, são regra geral armazenados incialmente em áreas intermediárias de memória (*buffers*).

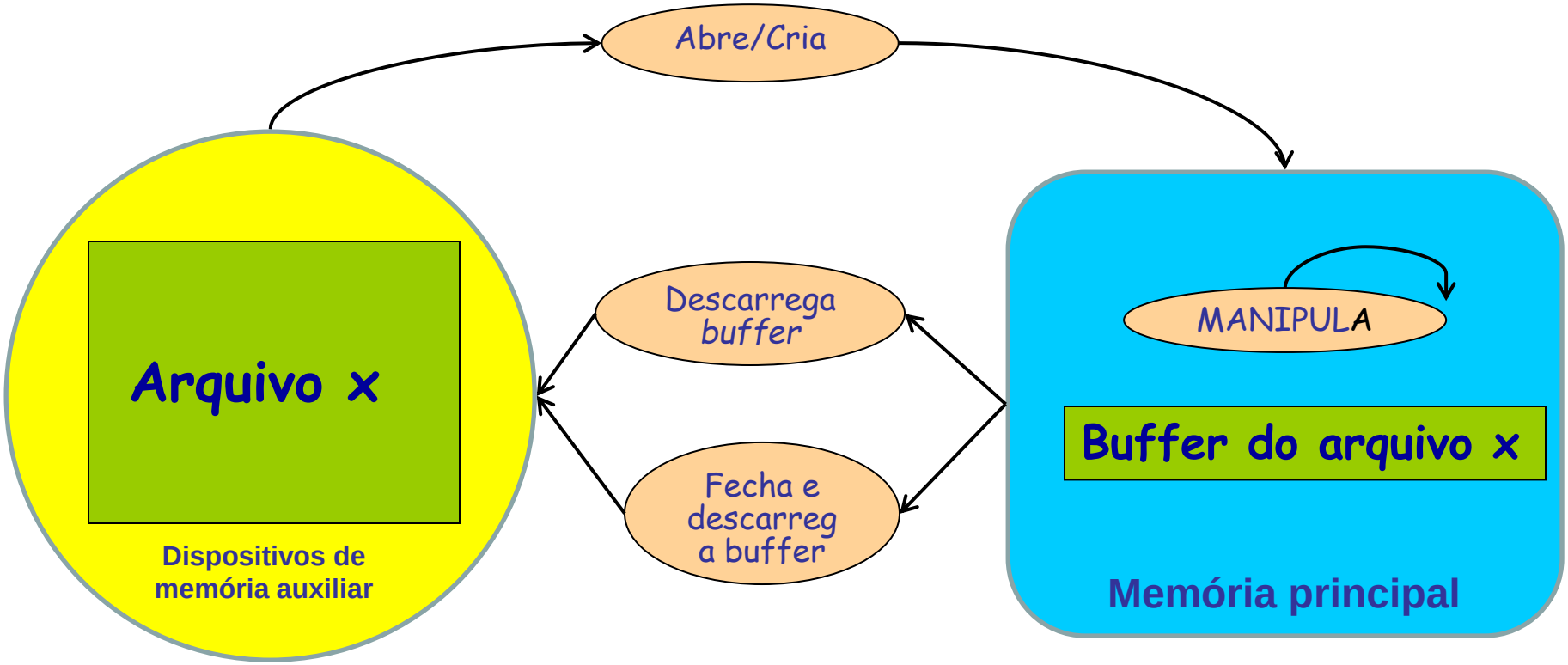
- Essa intermediação o mais das vezes só é percebida pelo usuário em caso de problemas. Ex.: leitura de caracteres ou *strings* do teclado não acontecer da forma prevista ou esperada.

Áreas intermediárias (*buffers*) para acesso a arquivos

- O conteúdo das áreas intermediárias pode ser transferido para os arquivos de forma automática ou com o acionamento de comandos ou funções específicas para tal.

Ex.: ao serem acionadas funções de leitura *scanf* , dados são retirados de fluxos de dados e colocados em variáveis.

Esquema de utilização de arquivos



Movimentação de dados

Funções

Os dados inicialmente ficam em memória principal (nos *buffers*) e após são descarregados (gravados) no dispositivo de memória auxiliar.

Arquivos: quanto à sua utilização

A linguagem C não impõe estrutura alguma aos arquivos e, do ponto de vista **físico**, um arquivo é, simplesmente, uma sequência (ou *fluxo*) de *bytes*.

Mas, do ponto de vista **lógico**, nós, os projetistas, imaginamos os dados armazenados em variáveis ou organizados em estruturas (*struct*), vetores, matrizes, etc.

Arquivo: declaração de ponteiro para arquivo

Para se poder utilizar um arquivo de dados, é necessário declarar um ponteiro do tipo FILE para uma estrutura com seus dados, que servirá no programa para manipulá-lo.

Exemplos:

```
FILE  *pf;
FILE  *arq;
FILE  *arq1,*arq2;
```

Observar que FILE é escrito em maiúsculas, para destacar que não é um tipo nativo da linguagem!!!

Arquivo: declaração de ponteiro para arquivo (cont.)

```
FILE  *pf;
FILE  *arq;
FILE  *arq1,*arq2;
```

com as declarações acima, as variáveis pf, arq, arq1 e arq2 passam a representar, dentro do programa, arquivos de dados.

Observação:

Um arquivo de dados é reconhecido como uma entidade externa ao programa.

Arquivo: abertura

Para processar um arquivo, deve-se conectar um ponteiro FILE em nosso programa a esse arquivo, em uma operação denominada Abertura de Arquivo.

A associação utiliza a declaração FILE.

A abertura de um arquivo é feita através da função **fopen** que retorna um ponteiro válido, se a operação for bem sucedida, ou NULL, se ocorrer erro.

Protótipo:

FILE *fopen (char *nome_de_arquivo, char *modo de abertura)

Ex:

```
arq =fopen("dadosmeus.txt","r");
```

Arquivo: abertura (cont.)

Os dois parâmetros da função fopen:

1º. Nome do arquivo ou onde localizá-lo (se constante, entre " ")

```
Ex:

arq =fopen("dadosmeus.txt","r");
```

2º. O modo de abertura

```
Ex:

arq =fopen("dadosmeus.txt","r");
```

Arquivo: abertura (cont.)

No modo de abertura indica-se:

- o tipo do arquivo, se binário ou texto (na sequência esses tipos são detalhados);[1]
- que operações podem ser realizadas sobre o arquivo: só leitura, só escrita, leitura e escrita;
- se um arquivo com mesmo nome for encontrado, se ele deve ser eliminado e um novo arquivo criado em seu lugar;
- a partir de onde os dados devem ser gravados no arquivo: se do início, ou se do fim válido no momento da abertura.

Conforme o modo de abetura, igualmente a não existência prévia de um arquivo pode resultar em erro.

[1] Ex.:

```
arq = fopen("dadosmeus.txt","a"); // abertura de arquivo texto
pf = fopen("outros dados","wb+"); // abertura de arquivo binário
```

Arquivo: abertura (cont.)

```
arq =fopen("dados.dat","ab");

pf =fopen("dadosteus.txt","r");
```

Nos exemplos acima **não** foi indicado onde os arquivos se encontram, assim o sistema entenderá que os **arquivos estão na mesma pasta** que **contém o programa** que os está manipulando.

Caso o arquivo não esteja na mesma pasta em que se encontra o executável do programa, é preciso informar explicitamente a pasta onde ele está:

```
pf=fopen("c:\\tuapasta\\dadosteus.txt","r");
```

Arquivo: abertura (cont.)

```
pf=fopen("c:\\tuapasta\\dadosteus.txt","r");
```

Nesse exemplo estamos informando que o arquivo **dadosteus.txt** está presente na pasta **c:\tuapasta**

Observar que o símbolo \ tem um significado especial para a linguagem C, então deve-se escrever mais uma barra \ no *string* de localização do arquivo:

```
c:\\tuapasta\\dadosteus.txt
```

Outro exemplo:

```
arq=fopen("c:\\tuapasta\\ccarq\\dadosteus.txt","r");
```

Arquivo: abertura (cont.)

Se, por alguma razão o arquivo **não** puder ser aberto, a função **fopen** irá retornar a constante **NULL**

```
if(!(arq=fopen("c:\\tuapasta\\ccarq\\dadosteus.txt","r")))
    {
     printf("Erro na abertura");
     system("pause");
    }
else
  {
    (...) //processamento do arquivo
    fclose(arq);
    system("pause");
    return 0;
 }
```

Se der erro de abertura, o processamento encerra sem return 0, o que significa execução não ok.

Arquivo: abertura (cont.)

Em caso de ocorrer erro na execução da função fopen, ela tão somente retorna a constante NULL.

O programa não aborta, como ocorre em outras linguagens.

Em linguagem C é da responsabilidade do programador o tratamento dos erros no processamento de arquivos.

Arquivo: fechamento

Da mesma forma que um programa deve abrir um arquivo antes de usá-lo, ele deve fechá-lo quando não mais precisar dele.

Isso é realizado pela função *fclose*, que fecha um fluxo de dados:

Protótipo: `int fclose (FILE *f)`

Ex.:

```
fclose(arq);
```

O valor retornado por *fclose* é 0 (zero) em caso de sucesso ou a constante EOF, em caso de erro.

Se for crítico descobrir se o fechamento do arquivo aconteceu de forma correta, ele deve ser verificado.

Arquivo: fechamento (cont.)

Ao fechar um fluxo de dados todos os dados que ainda estavam em áreas auxiliares associadas são salvos permanentemente no arquivo correspondente, no dispositivo de memória auxiliar.

Recomenda-se que sempre que um arquivo seja criado ou alterado, seja fechado explicitamente no programa.

Mas de qualquer forma, ao encerrar-se um programa, o sistema fecha os arquivos abertos e libera as áreas auxiliares (*buffers*) associadas.

Arquivo:

Arquivos em C podem ser classificados conforme 2 (dois) modos de processamento dos caracteres que compõem o fluxo de *dados*:

binário

texto

Arquivo: operações importantes

Resumindo, as operações importantes quando se manipula arquivos são:

1. Declarar um ponteiro FILE para cada arquivo a utilizar.
2. Abertura:

   Associação do ponteiro FILE com o arquivo físico: com a função *fopen*.

   Com indicação se o arquivo sendo aberto é binário ou texto.
3. Fechamento:

   Com a função *fclose*.
4. Ler, escrever, posicionar-se no arquivo:

   Com funções que podem ser diferentes conforme o modo de processamento da informação, seja TEXTO ou BINÁRIO.