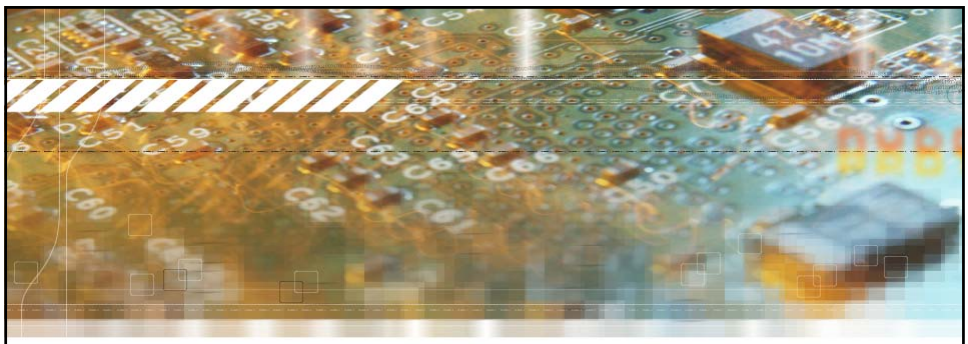


Processamento de arquivos

Capítulo 11

Processamento de Arquivos (Cap.11)

Sumário

- Arquivos e Fluxo;
- Acesso Seqüencial e Aleatório;
- Biblioteca e funções;
- Abertura;
- Leitura;
- Gravação;
- Fechamento.

Processamento de Arquivos (Cap.11)

Objetivo

Armazenamento de dados em meio físico **não volátil** (disco rígido).

Hierarquia de Dados

- Bit;
- Byte (caractere);
- Campo (grupo de caracteres com significado);
- Registro (composto por vários campos);
- Arquivo (grupo de registros relacionados);
- Banco de dados (conjunto de arquivos);

Processamento de Arquivos (Cap.11)

Arquivos e Fluxos (*Streams*)

As operações de E/S em C são realizadas por funções de bibliotecas, que possibilitam transmitir dados em representação tanto binária como em formato de texto.

Esse sistema de arquivos é projetado para trabalhar com uma variedade de dispositivos, como: impressora, monitor, disco rígido, etc. Embora os dispositivos sejam diferentes, eles são transformados em dispositivos lógicos chamados *streams*. Isso permite com que a mesma função exiba um texto na tela ou o grave no disco.

As *streams* podem ser de texto (seqüência de caracteres, "legível") ou binárias (seqüência de bytes, sem "tradução").

Processamento de Arquivos (Cap.11)

Arquivos e Fluxos (*Streams*)

Arquivo: O C visualiza cada arquivo como um fluxo seqüencial de informações (bytes) terminando com um marcador.

Fluxos: Canais de comunicação entre arquivos/dispositivos e programas.
Fluxo binário e de texto;
Buffer de disco;

Fluxos automaticamente abertos:

- Entrada Padrão;
- Saída Padrão;
- Erro.

Processamento de Arquivos (Cap.11)

Tipos de Acesso a Arquivos

- **Acesso seqüencial**: dados armazenados na forma de caracteres, dígitos, símbolos, ordenados segundo o campo-chave de cada registro (campos de tamanho variável).
- **Acesso aleatório**: dados armazenados de forma binária (campos de tamanho fixo)

Processamento de Arquivos (Cap.11)

Funções de Acesso a Arquivos

Biblioteca: **stdio.h**

-Funções: **fopen( )** – abre arquivos

fclose( ) – fecha arquivos

putc ( ) – escreve caractere

fputc ( ) – escreve caractere

getc ( ) – lê um caractere

fgetc ( ) – lê um caractere

Processamento de Arquivos (Cap.11)

Funções de Acesso a Arquivos

Biblioteca: **stdio.h**

fseek ( ) – posiciona o ponteiro de leitura/gravação

fprintf ( ) – escreve uma string/caractere/número

fscanf ( ) – lê uma string/caractere/número

feof ( ) – devolve v se o fim for atingido

rewind ( ) – posiciona o ponteiro de leitura/gravação no início

remove ( ) – apaga o arquivo

fflush ( ) – descarrega um arquivo

Processamento de Arquivos (Cap.11)

Ponteiro de Arquivo

A referência ao conjunto de informações de um arquivo (nome, estado, posição atual) é feita através de um ponteiro do tipo **FILE**

Ex:

```
FILE *fp;
```

Processamento de Arquivos (Cap.11)

Abertura de Um Arquivo

A função **fopen** → abre o fluxo para um arquivo de saída.
Protótipo:

```
FILE* fopen (const char *nome, const char
                *modo);
```

O primeiro termo é o nome do arquivo que vai ser aberto e o segundo se refere ao modo de abertura.

Processamento de Arquivos (Cap.11)

Modo de Abertura de Arquivo

"r" – abre arquivo texto para leitura
"w" – cria arquivo texto para escrita. Se existe um arquivo, elimina conteúdo.
"a" – anexa; abre um arquivo existente para gravação, e continua a partir do fim de seu conteúdo.

"rb" – abre arquivo binário para leitura
"wb" – cria arquivo binário para leitura
"ab" – anexa a um arquivo binário

"r+" – abre arquivo texto para leitura/gravação (atualização).
"w+" – cria arquivo texto para leitura/gravação. Se existe um arquivo, elimina conteúdo.
"a+" – anexa; abre um arquivo existente para leitura/gravação, e continua a partir do fim de seu conteúdo.

Processamento de Arquivos (Cap.11)

Exemplos de abertura de arquivo

Ex.1:

```
FILE *fp;
fp = fopen("teste.dat", "w");
```

Cria um arquivo para escrita ("w") chamado **teste.dat** na mesma pasta em que está o programa.

Processamento de Arquivos (Cap.11)

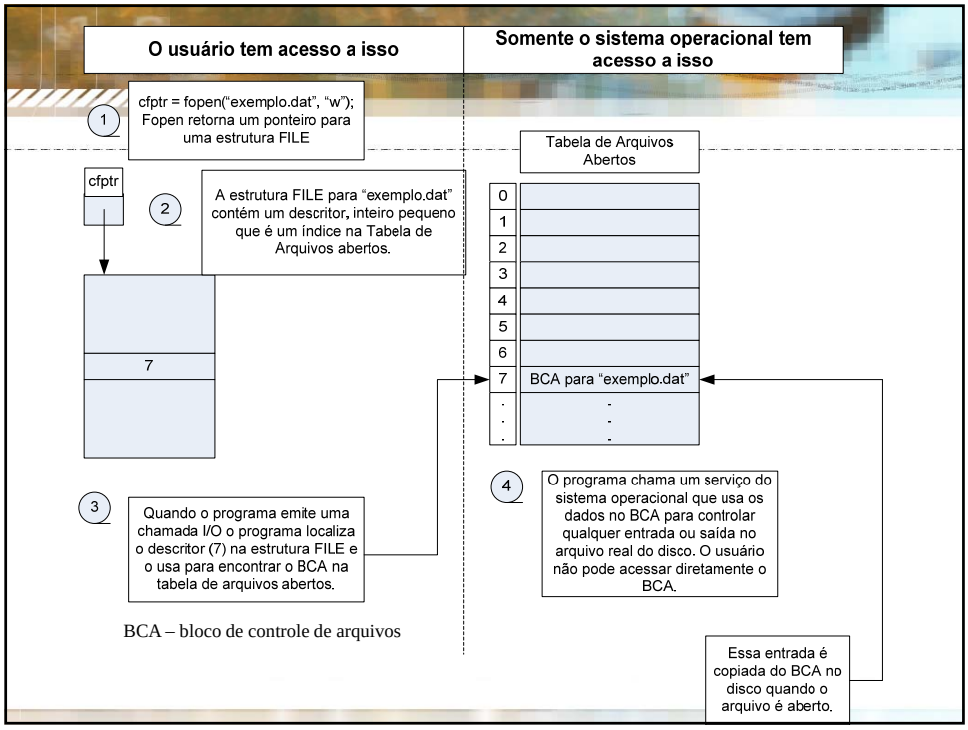
Exemplos de abertura de arquivo

Ex.2:

```
FILE *fp

if((fp=fopen("teste.dat", "w"))==NULL)
   printf("Arquivo não pode ser aberto");
```

A adição do condicional **if** testa se o arquivo foi aberto corretamente. Caso não tenha sido, a função **fopen()** retorna um ponteiro nulo (NULL) e o programa indica que o arquivo não foi aberto.



BCA – bloco de controle de arquivos

Processamento de Arquivos (Cap.11)

Fechamento de um Arquivo

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

Exemplo:

```
FILE *fp
if((fp=fopen("teste.dat", "w"))==NULL)
   printf("Arquivo não pode ser aberto");

...

...

fclose(fp);
```

Recomenda-se fechar os arquivos após o término de seu uso. Essa prática libera os recursos para serem utilizados por outros programas.

Processamento de Arquivos (Cap.11)

Acesso seqüêncial

Escrevendo em um Arquivo de Texto

Protótipo:

```
fprintf(FILE *arquivo, const char *formato, ...);
```

Exemplo:

```
fprintf(fp, "%s", teste);
```

Lendo de um Arquivo de Texto

Protótipo

```
fscanf(FILE *arquivo, const char *formato, ...);
```

Exemplo:

```
fscanf(fp, "%s", &teste);
```

Processamento de Arquivos (Cap.11)

Criando um Arquivo de Acesso Seqüencial

```c
#include <stdlib.h>

int main()
{
int conta;
char nome[30];
float saldo;
FILE *cptr;
if ((cptr = fopen("clientes.dat","w")) == NULL)
printf("Arquivo nao pode ser criado.");
else
{
printf("Digite a conta, o nome e o saldo. \n");
printf("Digite EOF para encerrar a entrada de dados.\n");
printf("? ");
scanf("%d%s%f",&conta, nome, &saldo);
while (!feof(stdin))
{
fprintf(cptr,"%d %s %.2f\n", conta, nome, saldo);
printf("? ");
scanf("%d%s%f", &conta, nome, &saldo);
}
fclose(cptr);
}
system("pause");
return 0;
}
```

```
Digite a conta, o nome e o saldo.
Digite EOF para encerrar a entrada de dados.
?
```

Processamento de Arquivos (Cap.11)

Lendo e Imprimindo um Arquivo de Acesso Seqüencial

```c
#include <stdio.h>
int main()
{
int conta;
char nome[30];
float saldo;

FILE *cptr;
if ((cptr = fopen("clientes.dat","r")) == NULL)
printf("Arquivo nao pode ser aberto.\n");
else
{
printf("%-10s%-13s%s\n", "Conta", "Nome", "Saldo");
fscanf(cptr,"%d%s%f",&conta, nome, &saldo);
while (!feof(cptr))
{
printf("%-10d%-13s%7.2f\n", conta, nome, saldo);
fscanf(cptr, "%d%s%f", &conta, nome, &saldo);
}
fclose(cptr);
}
system("PAUSE");
return 0;
}
```

```
Conta     Nome          Saldo
1         Samir          300.00
2         Pedro          650.00
4         Maria          700.00
Pressione qualquer tecla para continuar.
```

Processamento de Arquivos (Cap.11)

Acesso Seqüencial ao Arquivo

- Fácil de implementar

- Não permite atualizações que ocupem mais espaço do que a informação anterior.

Ex.: 12 João 12,3 72 Roberto 20,0

Não seria possível substituir João por Augusto, pois os 3 caracteres excedentes sobreporiam parte da informação 12,3. Logo, seria necessário ler toda informação do arquivo e reescrevê-lo com as devidas alterações.

- Para ler uma determinada posição geralmente parte-se do início do arquivo e a leitura do mesmo é feita até que seja alcançado o ponto de interesse.

Processamento de Arquivos (Cap.11)

Acesso Aleatório ao Arquivo

Os arquivos de acesso aleatório tem a informação armazenada não como dígitos, símbolos, caracteres, mas como bytes. Logo, a representação de 1 ou 65000 ocupará o mesmo espaço no arquivo. Torna possível estimar a posição dos campos dentro do arquivo de dados.

- Permitem o acesso aleatório a uma posição do arquivo
- A atualização de informações torna-se segura, sem comprometer outras informações.
- Mais complexo de trabalhar que a forma seqüencial.

Processamento de Arquivos (Cap.11)

Acesso Aleatório ao Arquivo

- `fread()` – Lê dados do arquivo;
- `fwrite()` - Escreve dados no arquivo
- `fseek()` - Busca um determinado dado no arquivo e retorna sua posição caso encontre.

Processamento de Arquivos (Cap.11)

Exercício:

Você é proprietário de uma loja de componentes eletroeletrônicos e precisa manter um inventário que lhe indique qual o código, a quantidade, custo unitário e descrição do material que está em estoque. Escreva um programa para tal aplicação, seguindo as seguintes especificações:

- Armazenar os dados em um arquivo "**estoque.dat**";
- Utilizar "estruturas" para os componentes contendo número de registro (único para cada tipo de produto), descrição, quantidade em estoque e custo unitário.

Processamento de Arquivos (Cap.11)

Exemplos para estudo:

Pg 357-> fig 11.3

Pg 362 -> fig 11.7