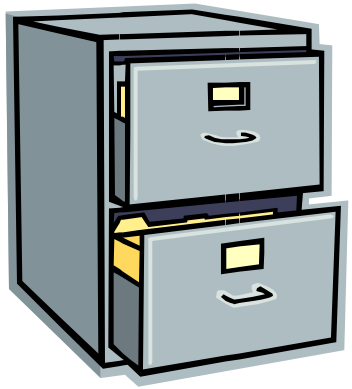
Arquivos

São estruturas de dados manipuladas fora do ambiente do programa.



Arquivos

- São estruturas de dados manipuladas fora do ambiente do programa;
- Considera-se como ambiente do programa a memória principal (MP);
- Nem sempre é possível ou conveniente manter certas estruturas de dados na MP;

Arquivos

- Nem sempre é possível ou conveniente manter certas estruturas de dados na MP;
- Uma alternativa é armazenar os dados em um dispositivo de memória secundária;

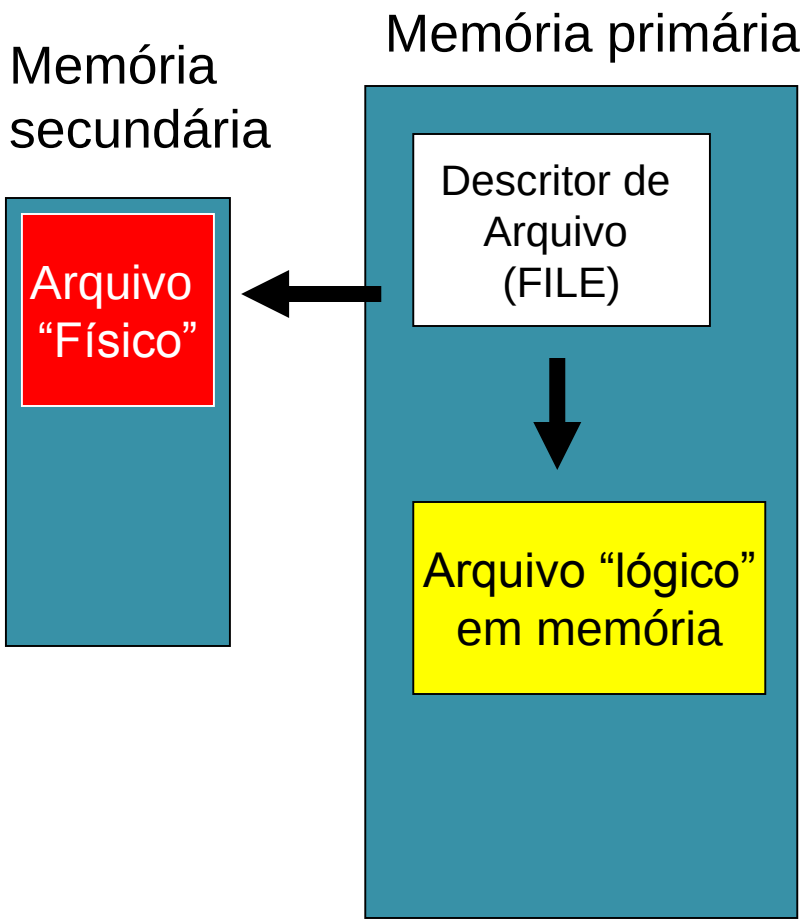
  Exemplos: discos, pendrive, etc.
- Podem ser lidos ou gravados por um programa.

Arquivos

Tipos de arquivo:

- Sequencial (texto)
  - Caracteres armazenados sequencialmente.
  - É possível determinar o primeiro, segundo terceiro .. caracteres que compõem o arquivo.
- Binário
  - Formado por uma sequência de bytes sem correspondência com um tipo de dado.
  - Cabe ao programador fazer esta correspondência quando lê e escreve estes arquivos.

Descritor de Arquivos



Descritor de arquivo:

- Em C: Estrutura de dados denominada **FILE**
- Armazena informações sobre o arquivo em memória secundária (arquivo físico).
- O arquivo lógico é armazenado em variáveis de memória definidas pelo programador.
- Criando um descritor:

```
FILE *arquivo;
```

Declaração de Arquivos

Definição de variáveis tipo “Arquivo”:

```
FILE *arquivo;
```

Abrindo Arquivos

```
FILE *arquivo;
arquivo = fopen("nome", "modo");
```

Onde:

nome = nome externo do arquivo físico.

modo = tipo do arquivo (texto ou binário) e objetivo de uso (leitura, escrita ou anexação)

Abrindo Arquivos

```
FILE *arquivo;
arquivo = fopen("nome", "modo");
if(arquivo!=NULL) printf("Arquivo aberto.");
```

O comando **fopen** retorna um número inteiro (ponteiro para o arquivo aberto) ou NULL quando houver um erro de abertura do arquivo.

Abrindo Arquivos

```
arquivo = fopen ("nome_externo", "modo");
```

Modos de utilização de arquivos	
Modo	**Significado**
r	Abre um arquivo texto para leitura.
w	Cria um arquivo texto para escrita.
a	Anexa a um arquivo-texto.
rb	Abre um arquivo binário para leitura.
wb	Cria um arquivo binário para escrita.
ab	Anexa um arquivo binário.
r+	Abre um arquivo-texto para leitura/escrita.
w+	Cria um arquivo-texto para leitura/escrita.
a+	Anexa ou cria um arquivo-texto para leitura/escrita.
r+b	Abre um arquivo binário para leitura/escrita.
w+b	Cria um arquivo binário para leitura/escrita.
a+b	Anexa a um arquivo binário para leitura/escrita.

Abrindo Arquivos

Exemplo: Criando um arquivo texto

```
FILE *parq;
parq = fopen("texto.txt", "w");
```

Onde “w” indica arquivo criado para escrita (write).

O modo ‘w’ também pode ser utilizado para escrever sobre um arquivo já existente, perdendo os dados anteriores.

Abrindo Arquivos

Exemplo: Criando um arquivo texto (apenas escrita)

```
FILE *parq;
parq = fopen("texto.txt", "w");
```

Exemplo: Criando um arquivo para leitura/escrita

```
FILE *parq;
parq = fopen("texto.txt", "w+");
```

Leitura de arquivos

A função **fopen()** também pode ser utilizada para abrir um arquivo existente e permitir a leitura dos dados que foram gravados.

Exemplo:

parq= **fopen**(“nome_arq”, “r”)

Onde “r” indica arquivo aberto para leitura (read).

Abrindo Arquivos

Exemplo: Abrindo um arquivo existente

```
FILE *parq;
parq = fopen("texto.txt", "r");
if (parq == NULL) {
    printf("\n Nao foi possivel abrir o arquivo.\n");
    }
```

Escrevendo em Arquivos

fprintf(): Utilizado para gravar um conteúdo em um arquivo existente.

Exemplo:

```
fprintf(parq, ”Aula de Algoritmos.”);
fprintf(parq, “3 vezes por semana.”);
```

Fechando Arquivos

fclose(): Utilizado para fechar arquivos que foram abertos pelo método fopen().

As atualizações em um arquivo só são efetuadas quando ele é fechado.

Fechando Arquivos

Exemplo:

```
fclose(parq);
```

Fechar um arquivo protege os seus dados, garante que atualizações feitas serão salvas e libera o arquivo para outros usuários ou programas poderem utilizá-lo

Lendo arquivos

- Função fgets()
- Função fscanf()

Função fgets()

Utilizado para ler linha de dados de um arquivo texto.

```
char frase[100];
fgets(frase,100, parq);  //Até 100 caracteres
```

Le uma string do arquivo *parq* para a variável frase.

Exemplo

Abra um bloco de notas e crie o arquivo *texto.txt* contendo:

Arquivo: “texto.txt”

```
Primeira linha
Linha 2
Ultima linha
```

Função fgets()

Exemplo:

```
FILE *parq;
parq = fopen(“texto.txt", "r");
char frase[100];
// Ler uma string do arquivo parq para a variável frase.
fgets(frase,100,parq);  //Até 100 caracteres
printf("\n%s\n", frase);
```

Função fgets()

Exemplo:

```
FILE *parq;
parq = fopen(“texto.txt", "r");
char frase[100];
// Ler para a variável frase várias strings do arquivo parq.
while (fgets(frase,100,parq) != NULL)
   printf("\n%s\n", frase);
```

Para ler números a partir de um arquivo texto, utilize a função fscanf().

Lendo arquivos

- Função fgets()

- **Função fscanf()**

Função fscanf()

- Le um valor de qualquer tipo a partir de um arquivo texto.
- Permite entrada de valor formatado conforme a sua natureza (tipo).
- Os valores devem estar separados por **um espaço** no arquivo.

Função fscanf()

Sintaxe:

fscanf(arquivo, formato, variável);

Onde:

- arquivo = nome externo do arquivo (origem);
- formato = tipo do valor lido (%i, %f ou %c);
- variável = destino para o valor lido do arquivo.

Exemplo

Considere um arquivo texto contendo o seguinte.

arqTexto.txt

```
10 20.5 w palavra
```

Monte um programa para ler os valores do arquivo e mostrar na tela.

Exemplo

Le os valores do arquivo e mostra na tela.

```
FILE *parq;
parq = fopen(“arqTexto.txt ", "r");
if ( parq == NULL) {
    printf("\n Nao foi possivel abrir o arquivo para escrita.\n");
    return;                }
char letra, palavra[20];
int num; float num2;
fscanf(parq, "%i", &num);
printf("\nNumero inteiro = %i", num);
fscanf(parq, "%f", &num2);
printf("\nNumero real = %5.2f", num2);
fscanf(parq, " %c", &letra);
printf("\nLetra = %c", letra);
fscanf(parq, " %s", &palavra);
printf("\nPalavra = %s", palavra);
fclose(parq);
```

Exemplo

Le os valores do arquivo e mostra na tela **em uma linha**.

```
FILE *parq;
parq = fopen("numeros.txt", "r");
if ( parq == NULL) {
    printf("\n Nao foi possivel abrir o arquivo para escrita.\n");
    return;                }
char letra, palavra[20];
int num; float num2;
fscanf(parq, "%i %f %c %s", &num, &num2, &letra, &palavra);
printf("\nValores = %i %5.2f %c %s", num, num2, letra, palavra);
fclose(parq);
```

Função ftell()

Retorna o número da posição corrente no arquivo.

Exemplo:

```
printf("Posicao do ponteiro no arquivo:  %d",
   ftell(parq));
```

17

MÉTODOS DE ARQUIVOS

Comando para re-abertura de arquivos:

```
rewind(parq);
```

Reinicia o ponto de leitura/escrita do arquivo.

A cada leitura/escrita, o arquivo vai para a "próxima" variável.

O comando rewind reinicia a leitura/escrita a partir da **primeira posição** do arquivo.

MÉTODOS DE ARQUIVOS

Comando para verificar o final do arquivo:

```
feof(arquivo);
```

O comando **feof** retorna um número inteiro;

Um número maior que zero significa que o arquivo está no final, e não há mais dados no arquivo;

0(zero) indica que ainda existem dados.

Exemplo

Abra um bloco de notas e crie o arquivo *linhas.txt* contendo:

Arquivo: “linhas.txt”

```
Linha ativa
#Linha com sharp
#Outra linha com sharp
Linha 2
#Ultima linha com sharp
Fim
```

Exemplo

Abra um bloco de notas e crie o arquivo *linhas.txt.*

Arquivo: “linhas.txt”

```
Linha ativa
#Linha com sharp
#Outra linha com sharp
Linha 2
#Ultima linha com sharp
Fim
```

Arquivo: “linhas2.txt”

```
Linha ativa
Linha 2
Fim
```

Crie uma função para copiar o arquivo *linhas.txt* omitindo quaisquer linhas que comecem por #.

Exemplo

Copiar *velho* omitindo quaisquer linhas que comecem por #.

```
void filtraArquivo(char velho[], char novoArquivo[]){
    // retira as linhas que começam com #
    FILE *f1, *f2;
    f1 = fopen(velho, "r");
    if ( f1 == NULL) {
        printf("\n Nao foi possivel abrir o arquivo para leitura.\n");
        return;            }
    f2 = fopen(novoArquivo, "w");
    char texto[50];
    while (1){
        fgets(texto,50,f1);
        if (feof(f1))    break;
        if (texto[0] != '#')       fprintf(f2, texto);
    }
    fclose(f1);    fclose(f2);
}
```

Exemplo

Copiar velho omitindo quaisquer linhas que comecem por #.

```
void filtraArquivo(char velho[], char novoArquivo[]){
    // retira as linhas que começam com #
    FILE *f1, *f2;
    f1 = fopen(velho, "r");
    if ( f1 == NULL) {
        printf("\n Nao foi possivel abrir o arquivo para leitura.\n");
        return;            }
    f2 = fopen(novoArquivo, "w");
    char texto[50];
    while (1){
        fgets(texto,50,f1);
        if (feof(f1))    break;
        if (texto[0] != '#')       fprintf(f2, texto);
    }
    fclose(f1);    fclose(f2);
}
```

Arquivo: “linhas.txt”

```
Linha ativa
#Linha com sharp
#Outra linha com sharp
Linha 2
#Ultima linha com sharp
Fim
```

filtraArquivo(“linhas.txt”, “linhas2.txt”)

Exemplo

Copiar velho omitindo quaisquer linhas que comecem por #.

Arquivo: “linhas.txt”

```
Linha ativa
#Linha com sharp
#Outra linha com sharp
Linha 2
#Ultima linha com sharp
Fim
```

```
filtraArquivo(“linhas.txt”, “linhas2.txt”)
```

Arquivo: “linhas2.txt”

```
Linha ativa
Linha 2
Fim
```

Exemplo

Editar o arquivo linhas3.txt contendo as frases abaixo.

Arquivo: “linhas3.txt”

```
Linha ativa
Linha com sharp #aqui
Outra linha com sharp #ali
Linha 4
Ultima linha com sharp #final
Fim
```

Criar uma função para copiar *linhas3.txt* omitindo quaisquer linhas a partir de #.

Copiar *linhas3.txt* omitindo quaisquer linhas a partir de #.

Arquivo: “linhas3.txt”

```
Linha ativa
Linha com sharp #aqui
Outra linha com sharp #ali
Linha 4
Ultima linha com sharp #final
Fim
```

filtraArquivo(“linhas3.txt”, “linhas4.txt”)

Arquivo: “linhas4.txt”

```
Linha ativa
Linha com sharp
Outra linha com sharp
Linha 4
Ultima linha com sharp
Fim
```

Copiar *linhas3.txt* omitindo quaisquer linhas a partir de #.

```
void filtraArquivo2(char velho[], char novoArquivo[]){
    FILE *f1, *f2;
    f1 = fopen(velho, "r");
    if ( f1 == NULL) {
       printf("\n Nao foi possivel abrir o arquivo para leitura.\n");
       return;            }
    f2 = fopen(novoArquivo, "w");
    char texto[50], texto2[50];
    int i, acheiSharp;
    while (1){
        fgets(texto,50,f1);
         ...
```

filtraArquivo2(“linhas3.txt”, “linhas4.txt”)

Copiar *linhas3.txt* omitindo quaisquer linhas a partir de #.

```
...
while (1){
        fgets(texto,50,f1);
        if (feof(f1))  break;
        acheiSharp = 0; i = -1;
        for (int k=0; k<strlen(texto); k++) {
            i++;
            if (texto[k] != '#')  texto2[i] = texto[k];
            else {
                acheiSharp = 1;
                break;            }
        }
        if (acheiSharp)    texto2[i] = '\n';
        texto2[i+1] = '\0';
        fprintf(f2, texto2);
    }
    fclose(f1);    fclose(f2);
}
```

Arquivo: “linhas3.txt”

```
Linha ativa
Linha com sharp #aqui
Outra linha com sharp #ali
Linha 4
Ultima linha com sharp #final
Fim
```

filtraArquivo2(“linhas3.txt”, “linhas4.txt”)

EXERCÍCIO 1

Monte uma função para criar um arquivo chamado “alunos.dad” e armazenar os dados (NOME e CURSO) de diversos alunos, uma informação por linha.

Exemplo:

Ana

Pedagogia

Antonio

Engenharia

EXERCÍCIO 1

Monte uma função para criar um arquivo chamado “alunos.dad” e armazenar os dados (NOME e CURSO) de diversos alunos, uma informação por linha.

```
void criar(char arqAlunos[], int Nalunos){
    //Obter e gravar dados para Nalunos
    FILE *arq;
    arq = fopen(arqAlunos, "w");
    printf("\nArquivo de ALUNOS criado com sucesso!!!");
    char nome[30], curso[30];

    //Obter e gravar dados para Nalunos
    ...
```

criar(“alunos.dad”, 2)

Exercício 1

```
void criar(char arqAlunos[], int Nalunos){
    //Obter e gravar dados para Nalunos
    FILE *arq;
    arq = fopen(arqAlunos, "w");
    printf("\nArquivo de ALUNOS criado com sucesso!!!");
    char nome[30], curso[30];
    for (int i=0; i<Nalunos; i++){
        printf("\nNome do aluno: ");  gets(nome);
        printf("Nome do Curso: ");    gets(curso);
        strcat(nome, "\n");
        fprintf(arq, nome);
        strcat(curso, "\n");
        fprintf(arq, curso);
    }
    fclose(arq);
    printf("\n\tArquivo fechado.");
}
```

criar(“alunos.dad”, 2)

Exercício 2

Crie uma função para ler e mostrar os dados do arquivo “alunos.dad”.

Exercício 2

Crie uma função para ler e mostrar os dados do arquivo “alunos.dad”.

```
void lerArq(char arqAlunos[]){
    //Ler arquivo e mostrar os dados
    FILE *arq;
    arq = fopen(arqAlunos, "r");
    if ( arq == NULL){
        printf("\n Nao foi possivel abrir o arquivo para leitura.\n");
        return;      }
    printf("\n\nArquivo de ALUNOS aberto para leitura.\n");
    char nome[30], curso[30];

    #Ler arquivo e mostrar os dados
    ...
```

lerArq(“alunos.dad”)

Exercício 2

```
void lerArq(char arqAlunos[]){
    FILE *arq;
    arq = fopen(arqAlunos, "r");
    if ( arq == NULL){
        printf("\n Nao foi possivel abrir o arquivo para leitura.\n");
        return;      }
    printf("\n\nArquivo de ALUNOS aberto para leitura.\n");
    char nome[30], curso[30];
    while (1){
        fgets(nome,30,arq);    //Le uma linha do arquivo
        if (feof(arq)){
            printf("\nAcabou o arquivo.");
            break;      }
        printf("\nNome = %s", nome);
        fgets(curso,30,arq);    //Le outra linha do arquivo
        if (feof(arq)){
            printf("\nAcabou o arquivo.");
            break;      }
        printf("Curso = %s", curso);
    }
    fclose(arq);    printf("\n\tArquivo fechado.");
}
```

Exercício 2

```
void lerArq(char arqAlunos[]){
    FILE *arq;
    arq = fopen(arqAlunos, "r");
    if ( arq == NULL){
        printf("\n Nao foi possivel abrir o arquivo para leitura.\n");
        return;      }
    printf("\n\nArquivo de ALUNOS aberto para leitura.\n");
    char nome[30], curso[30];
    while (1){
        fgets(nome,30,arq);    //Le uma linha do arquivo
        if (feof(arq)){
            printf("\nAcabou o arquivo.");
            break;      }
        printf("\nNome = %s", nome);
        fgets(curso,30,arq);    //Le outra linha do arquivo
        if (feof(arq)){
            printf("\nAcabou o arquivo.");
            break;      }
        printf("Curso = %s", curso);
    }
    fclose(arq);    printf("\n\tArquivo fechado.");
}
```

lerArq(“alunos.dad”)

Exercício 3

Faça uma função que copie todos os alunos do arquivo **alunosVelhos.txt** para um segundo arquivo chamado **alunos.ext** incluindo novos alunos fornecidos pelo usuário.

“alunosVelhos.txt”

```
80823
ALISSON NOGUEIRA TEIXEIRA
78788
ANDREY SOUTO MAIOR
...
```

Exercício 3

Faça uma função que copie todos os alunos do arquivo **alunosVelhos.txt** para um segundo arquivo chamado **alunos.ext** incluindo novos alunos fornecidos pelo usuário.

Solução:

- Abrir o arquivo antigo para leitura;
- Abrir o arquivo novo para escrita;
- Copiar os dados do antigo para o novo;
- Fechar o arquivo velho;
- Digitar os dados que serão incluídos;
- Escrever os novos dados no arquivo novo;
- Fechar o arquivo novo.

Exercício 3

```
void copiaInsere(char velho[], char novo[]){
    //Ler arquivo e mostrar os dados
    FILE *arq, *arq2;
    arq = fopen(velho, "r");
    if ( arq == NULL){
        printf("\n Nao foi possivel abrir %s para leitura.\n", velho);
        return;      }
    printf("\n\nArquivo %s aberto para leitura.\n", velho);
    char ra[7], nome[30];
    arq2 = fopen(novo, "w");
    printf("\n\tArquivo %s criado com sucesso!!!", novo);

    // COPIAR DADOS DO ARQUIVO VELHO PARA O NOVO
    ...
```

copiaInsere(“alunosVelhos.txt”, “alunos.ext”)

Exercício 3

```
...
   // COPIAR DADOS DO ARQUIVO VELHO PARA O NOVO
   while (1){
      fgets(ra,7,arq);    //Le uma linha do arquivo
      if (feof(arq)){
         printf("\nAcabou leitura do arquivo %s.", velho);
         break;    }
      fprintf(arq2, ra);
      fgets(nome,30,arq);    //Le outra linha do arquivo
      if (feof(arq)){
         printf("\nAcabou o arquivo %s.", velho);
         break;    }
      fprintf(arq2, nome);
   }
   printf("\n\tDados copiados p/ arquivo %s", novo);
   fclose(arq);
   printf("\nArquivo %s fechado.", velho);

   //INCLUIR NOVOS DADOS
...
```

Exercício 3

```
...
 printf("\n\nINCLUINDO NOVOS DADOS");
    while (1){
        printf("\nRA: "); gets(ra);
        if (strlen(ra) == 0)      break;
        strcat(ra, "\n");
        fprintf(arq2, ra);
        printf("Nome do aluno: ");    gets(nome);
        if (strlen(nome) == 0)      break;
        strcat(nome, "\n");
        fprintf(arq2, nome);
    }
    fclose(arq2);
    printf("\nArquivo %s fechado.\n", novo);
}
```

copiaInsere(“alunosVelhos.txt”, “alunos.ext”)

Modo append

fopen(parq, “**modo**”);

Para **inclusão** de novos elementos deve se abrir o arquivo utilizando o modo “a” (append) que permite inserir novos dados ao final do arquivo preservando os dados já existentes.