Programação de Computadores I

Arquivos na Linguagem C

PROFESSORA CINTIA CAETANO

Introdução

- As informações que os programas utilizam são perdidas quando eles são finalizados ou quando o computador é desligado.
- Isso porque as variáveis de um programa ficam armazenadas na memória primária, que é volátil, isto é, perde seu conteúdo.
- Quando você não quer perder as informações de seu programa, tendo-as a mão para a sua próxima execução, você deve guardá-las em um **ARQUIVO**.

Arquivos

- Os arquivos são estruturas especiais que ficam armazenadas na memória secundária do computador (disquete, disco rígido...).
- Servem para guardar as informações enquanto um programa não está em execução, pois elas não são voláteis.

Tipos de Arquivos

- Os arquivos podem ser de dois tipos:
  - Texto
  - Binário

- A escolha se um arquivo é texto ou binário depende do tipo de sua aplicação.

A melhor escolha do tipo de Arquivo

- TEXTO: utilizado para armazenar textos para serem lidos em outro lugar (como um editor de texto ou um relatório).

- BINÁRIO: utilizado para armazenar dados sobre alguma pessoa ou objeto (tipo registro ou estrutura), pois, além de ocupar menos espaço no armazenamento das informações, ele as protege um pouco de outros programas bisbilhoteiros.

Etapas

- O processo de utilização de um arquivo envolve, no mínimo, três etapas:
  1. Criação ou abertura do arquivo;
  2. Gravação ou leitura de dados no arquivo; e
  3. Fechamento do arquivo.

1. Criação ou abertura do arquivo

- Se o arquivo ainda não existir na memória secundária, ele deve ser criado.
- Caso o arquivo já existir (pelo fato de ter sido criado em uma execução anterior do programa) ele pode ser aberto para que novos dados sejam acrescentados ou para que os dados guardados nele possam ser lidos.

2. Gravação ou leitura de dados

- Lê ou grava dados no arquivo.
- A linguagem C oferece funções específicas de leitura e de escrita de dados em um arquivo.
- Se o arquivo for do tipo texto deve-se utilizar funções específicas para arquivos-texto.
- Se ele for binário (do tipo que armazena registros ou estruturas), deve-se utilizar as funções de leitura e/ou escrita em arquivos binários.

3. Fechamento do arquivo

- A terceira etapa consiste em fechar o arquivo, para que seus dados sejam efetivamente gravados e fiquem protegidos até que o arquivo seja aberto novamente.

Ponteiro

- O sistema de manipulação de arquivos do ANSI C é composto por uma série de funções interelacionadas, cujos protótipos estão reunidos na <stdio.h>.
- Todas estas funções trabalham com o conceito de "ponteiro de arquivo".
- Um ponteiro de arquivo aponta para uma estrutura que contém informações sobre o arquivo, como o local de um buffer, a posição do caracter corrente no buffer, se o arquivo está sendo lido ou gravado e se foram encontrados erros ou o fim do arquivo.

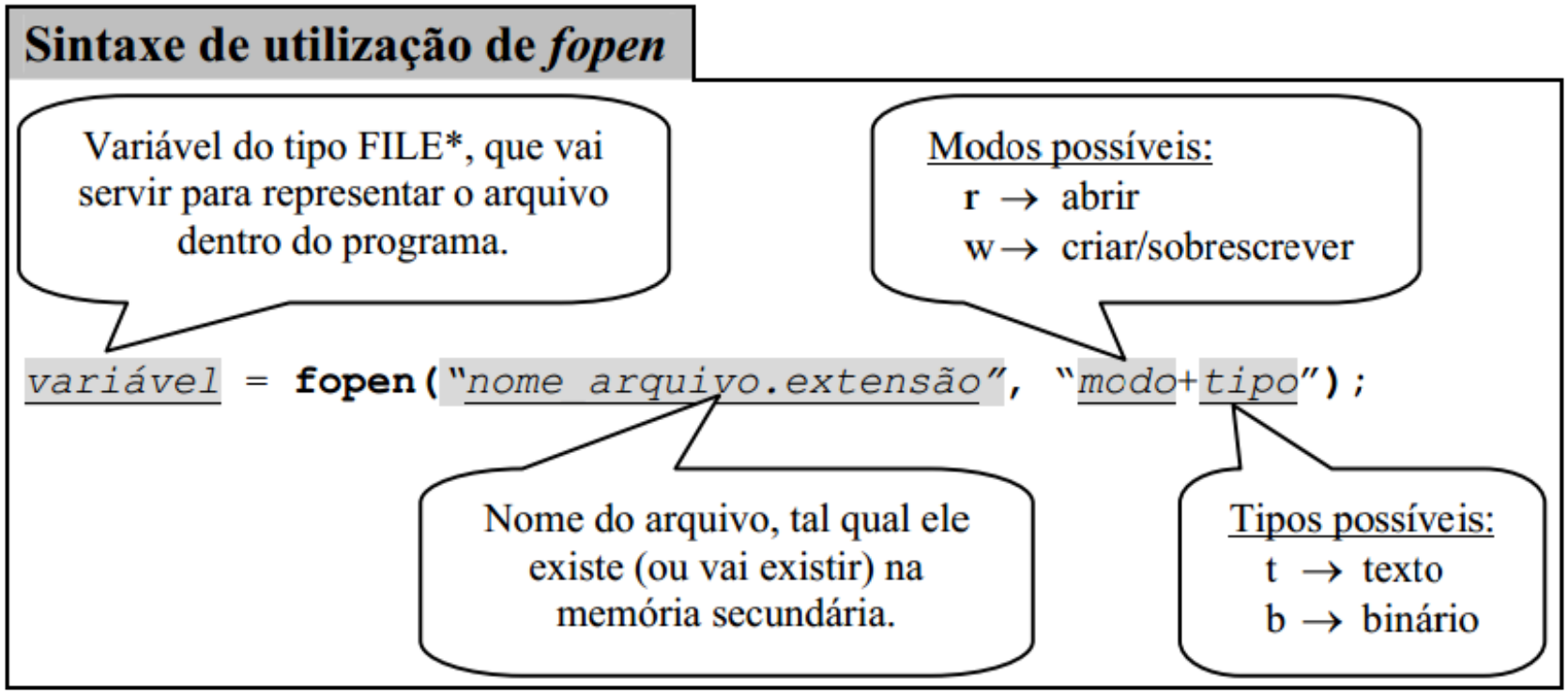
Abrindo ou criando um arquivo

- Dentro da linguagem C, os arquivos não podem ser manipulados diretamente.
- Eles precisam referenciados por uma variável do tipo FILE*.
- Logo, sempre que for utilizar um arquivo cria-se uma variável do tipo FILE* para poder utilizá-lo.
- Todas as funções de manipulação de arquivo necessitam de uma variável deste tipo para poder manipular esse arquivo.
- Declarar da variável do tipo FILE:
  - FILE* arquivo; // cria uma variável que manipula arquivos

Abrindo ou criando um arquivo

- Após ter declarado uma variável que vai representar o arquivo, deve-se efetivamente tentar abrir ou criar o arquivo.
- Isso é feito com a função fopen.
- Porém, antes de usar a função fopen, deve-se decidir se vai abrir ou criar o arquivo, e também se ele vai ser um arquivo do tipo texto ou do tipo binário.
- Dependendo da sua escolha, a função *fopen* vai ser chamada de uma forma ou de outra, com parâmetros diferentes que indicam as escolhas tomadas.

Função fopen



Valores Válidos para o Modo de Abertura de Arquivos

Modo	Significado
"r"	Abre um arquivo texto para leitura. O arquivo deve existir antes de ser aberto.
"w"	Abrir um arquivo texto para gravação. Se o arquivo não existir, ele será criado. Se já existir, o conteúdo anterior será destruído.
"a"	Abrir um arquivo texto para gravação. Se ele já existir, os dados serão adicionados no fim do arquivo ("append"). Caso contrário, um novo arquivo será criado.
"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. Se o arquivo já existir, os dados serão adicionados no fim do mesmo. Caso contrário, um novo arquivo será criado.
"rb"	Abre um arquivo binário para leitura (similar ao modo "r").
"wb"	Cria um arquivo binário para escrita (similar ao modo "w").
"ab"	Acrescenta dados ao final do arquivo binário (similar ao modo "a").
"r+b"	Abre um arquivo binário para leitura e escrita (similar ao modo "r+").
"w+b"	Cria um arquivo binário para leitura e escrita (similar ao modo "w+")
"a+b"	Acrescenta dados ou cria uma arquivo binário para leitura e escrita (similar ao modo "a+").

Exemplo

Abrindo um arquivo do tipo texto

```
arquivo = fopen("teste.txt", "r+t");
```

Criando um arquivo do tipo texto

```
arquivo = fopen("teste.txt", "w+t");
```

Exemplo

Abrindo um arquivo do tipo binário

```
arquivo = fopen("teste.txt", "r+b");
```

Criando um arquivo do tipo binário

```
arquivo = fopen("teste.txt", "w+b");
```

Teste de abertura e criação

- Antes de seguir em frente, você deve testar se o arquivo foi realmente aberto ou criado.
- Isso porque as demais funções de manipulação de arquivo não funcionarão se a função fopen não tiver funcionado.
- Para saber se a função fopen realmente criou ou abriu o arquivo que você solicitou, teste se a variável que representa o arquivo (a variável arquivo, no exemplo) possui algum valor dentro dela.
- Se ela não possuir um valor (se ele for NULL) é porque o arquivo não foi aberto ou criado.

Teste de abertura e criação

- Sintaxe

```
if(arquivo!=NULL)//Só entra aqui se o arquivo foi aberto/criado
{
   // funções de manipulação de arquivo
}
```

Fechando arquivo

- Quando acabamos de usar um arquivo que abrimos, devemos fechá-lo.
- Para tanto usa-se a função fclose().
- Esta função fecha um arquivo que tenha sido aberto através da função fopen().

- Sua sintaxe geral é:

  fclose(arquivo);

- Onde arquivo é o nome do ponteiro para o arquivo (ponteiro do tipo FILE) aberto pela função fopen().

Fechando arquivo

- O não fechamento de um arquivo pode causar vários tipos de problemas, incluindo perda de dados, destruição de arquivos e possíveis erros intermitentes no programa.
- Um fechamento de arquivo com fclose() também libera o bloco de controle de arquivo, deixando-o disponível para reutilização.

Lendo ou gravando dados no arquivo

- Para gravar, utilize a função fprintf, que é muito parecida com a função printf.
- A única diferença é que há um parâmetro (o primeiro) indicando qual é o arquivo onde os dados serão gravados (ou melhor, impressos).
- Exemplo:

```
int idade = 20;
fprintf(arquivo, "Essa frase vai ser gravada no arquivo\n");
fprintf(arquivo, "%d", idade); // grava a idade no arquivo
fprintf(arquivo, "%s tem %d anos\n", "Leo", idade);
// grava "Leo tem 20 anos"
```

Exemplo 1

```
#include <stdio.h>

void main()
{
    FILE *arq;
    arq = fopen("teste.txt","w");
    if (arq != NULL)
    {
        int num = 10;
        fprintf(arq, "Ola Mundo!\n");
        fprintf(arq, "%d\n", num);
        fprintf(arq, "O numero gravado foi %d", num);
        fclose(arq);
    }
    else
        printf("Erro na abertura do arquivo.");
}
```

Exemplo 2

```
#include <stdio.h>

void main()
{
    FILE *arq;

    arq = fopen("teste.txt","w");

    if (arq != NULL)
    {
        int num;
        printf("Entre com o numero: ");
        scanf("%d", &num);
        fprintf(arq, "O numero gravado foi %d", num);
        fclose(arq);
    }
    else
        printf("Erro na abertura do arquivo.");
}
```

Lendo ou gravando dados no arquivo

- Você também pode utilizar fprintf para gravar os dados de um Registro, mas seus campos têm que ser gravados um-a-um e serão gravados no formato texto:

```
struct tipoAluno{
   char nome[30];
   int idade;
}cad;
:
fprintf(arquivo,"%s %d\n", cad.nome, cad.numero);
```

Exemplo 3

```
#include <stdio.h>

struct tipoAluno
{
    char nome[30];
    int idade;
} cad;

void main()
{
    FILE *arq;

    arq = fopen("teste.txt","w");
    if (arq != NULL)
    {
        printf("Entre com o nome: ");
        gets(cad.nome);
        printf("Entre com a idade: ");
        scanf("%d", &cad.idade);
        fprintf(arq, "%s %d", cad.nome,
    cad.idade);
        fclose(arq);
    }
    else
        printf("Erro na abertura do
    arquivo.");
}
```

Exemplo 4

```
#include<stdio.h>
#define MAX 5

struct contato{
    char nome[30], tel[40];
};

void main()
{
    struct contato vet[MAX];
    int i;
    FILE *arq;
    arq = fopen("teste.txt","w");
    //Lendo vetor de registros
    for(i=0;i<MAX;i++)
    {
        printf("Nome: ");
        gets(vet[i].nome);
        printf("Telefone: ");
        gets(vet[i].tel);
        fprintf(arq, "%s %s\n", vet[i].nome,
    vet[i].tel);
    }
    fclose(arq);
}
```

Lendo a partir de arquivos-texto

- Para ler, utilize a função fscanf, que é muito parecida com a função scanf.
- A única diferença é que há um parâmetro (o primeiro) indicando qual é o arquivo de onde os dados serão lidos:

- Exemplo:

```
int  idade;
char nome[20];
fscanf(arquivo,"%d", &idade); // Lê uma idade do arquivo
fscanf(arquivo,"%s %d", nome, &idade);
//Lê nome e idade separados por um espaço
```

Exemplo 5

```
//Alteração do Exemplo3
#include <stdio.h>
struct tipoAluno
{
    char nome[30];
     int idade;
} cad;

void main()
{
    FILE *arq;

    arq = fopen("teste.txt","r"); //abre para
     leitura
    if (arq != NULL)
    {
        fscanf(arq, "%s %d", cad.nome,
     &cad.idade);
        fclose(arq);
    }
    else
        printf("Erro na abertura do arquivo.");

    printf("Lido do arquivo: %s %d", cad.nome,
     cad.idade);
}
```

Exemplo 6

```
//Alteração do Exemplo 4
#include<stdio.h>
#define MAX 5

struct contato{
    char nome[30], tel[40];
}vet[MAX];

void main()
{
    int i;

    FILE *arq;

    arq = fopen("teste.txt","r");

    //Lendo vetor de registros
    for(i=0;i<MAX;i++)
    {
        fscanf(arq, "%s %s\n", vet[i].nome,
vet[i].tel);
    }
    fclose(arq);

    for(i=0;i<MAX;i++)
    {
        printf("%s %s\n", vet[i].nome, vet[i].tel);
    }
}
```

Lendo a partir de arquivos binários

- Antes de ler ou gravar dados em arquivos binários, é importante se posicionar no local correto.
- Quando um arquivo é aberto, você fica posicionado na primeira posição do mesmo.
- Caso você queira fazer uma leitura em outra posição, deve-se posicionar corretamente.
- Da mesma forma, se você desejar gravar alguma coisa, deve-se posicionar no final do arquivo para que os dados existentes nele não sejam substituídos.

Gravando em arquivos-binários

- Para gravar, utilize a função *fwrite*, que grava estruturas inteiras de uma única vez:

```
struct tipoEndereco{
   char rua[30];
   int  numero;
   int  apartamento;
};

tipoEndereco endereço;
char         nome[20];

strcpy(nome, "Laura");
strcpy(endereco.rua, "Lima e Silva");
endereco.numero      = 1047;
endereco.apartamento = 215;

fwrite(&nome, sizeof(nome), 1, arquivo);
//      ^^^^ grava o nome no arquivo
fwrite(&endereco, sizeof(endereco), 1, arquivo);
//      ^^^^^^^^ grava toda a estrutura do tipoEndereco
```

Exemplo 7

```
#include<stdio.h>
#define MAX 5

struct tipoEndereco{
    char rua[30];
    int  numero;
    int  apartamento;
} endereco;

void main()
{
    FILE *arq;
    char nome[20];

    arq = fopen("teste.dat","wb");

    printf("Nome: ");
    gets(nome);
    printf("Rua: ");
    gets(endereco.rua);
    printf("Numero: ");
    scanf("%d", &endereco.numero);
    printf("Apartamento: ");
    scanf("%d", &endereco.apartamento);

    fwrite(&nome, sizeof(nome), 1, arq);  //grava
     o nome no arquivo
    fwrite(&endereco, sizeof(endereco), 1, arq);
     //grava toda a estrutura do tipoEndereco
}
```

Lendo a partir de arquivos-binários

- Para ler, utilize a função fread, que é muito semelhante com a função fwrite:

```
  // Utiliza as mesmas variáveis do exemplo anterior:

fread(&nome, sizeof(nome), 1, arquivo);
fread(&endereco, sizeof(endereco), 1, arquivo);

printf("%s mora na rua %s, numero %d, apartamento %d.\n",
  nome, endereco.rua, endereco.numero,  endereco.apartamento);
```

Exemplo 8

```
//Alteração do Exemplo 7

#include<stdio.h>
#define MAX 5

struct tipoEndereco{
    char rua[30];
    int  numero;
    int  apartamento;
} endereco;

 void main()
{
    FILE *arq;
    char nome[20];

    arq = fopen("teste.dat","rb");

    fread(&nome, sizeof(nome), 1, arq);
    fread(&endereco, sizeof(endereco), 1, arq);

    printf("\n\n%s mora na rua %s, numero %d,
apartamento %d.\n", nome, endereco.rua,
endereco.numero,  endereco.apartamento);

 }
```

Exemplo 9

```
#include<stdio.h>
#define MAX 5
struct contato{
    char nome[30], tel[40];
}vet[MAX];

 void main()
 {
    FILE *arq;
    int i;
    //Escreve no arquivo binário
    arq = fopen("teste.dat","wb");
    for(i=0;i<MAX;i++) {
        printf("Nome: ");
        gets(vet[i].nome);
        printf("Tel: ");
        gets(vet[i].tel);
        fwrite(&vet[i], sizeof(vet[i]), 1, arq);
    }
    fclose(arq);

    //Le do arquivo binário
    arq = fopen("teste.dat","rb");
    for(i=0;i<MAX;i++) {
        fread(&vet[i], sizeof(vet[i]), 1, arq);
    }
    for(i=0;i<MAX;i++) {
        printf("\n\nNome: %s - Telefone: %s\n",
     vet[i].nome, vet[i].tel);
    }
    fclose(arq);
}
```

Posicionando-se em um arquivo binário

- Para se posicionar em um arquivo, utilize a função *fseek*. Aconselha-se que você sempre se posicione no local correto antes de ler ou gravar qualquer dado.

```
// Utiliza as mesmas variáveis do exemplo anterior:

fseek(arquivo, 0, SEEK_END); // posiciona-se no final do arquivo
fwrite(&endereco, sizeof(endereco), 1, arquivo); // grava

fseek(arquivo, 0, SEEK_SET); // posiciona-se no início do
arquivo
fread(&endereco, sizeof(endereco), 1, arquivo); // lê o registro

// posiciona-se no segundo registro:
fseek(arquivo, 2*sizeof(endereço), SEEK_SET);
// obs: sizeof retorna o tamanho do registro!

fread(&endereco, sizeof(endereco), 1, arquivo); // lê o registro
```

Busca em arquivo-binário

```
#include<stdio.h>
#include<string.h>
#define MAX 5

struct contato{
    char nome[30], tel[40];
}a;

 void main()
 {
    FILE *arq;
    int achei = 0;
    char busca[20],aux[20];

    printf("\nQual o nome que deseja buscar? ");
    gets(busca);
    arq = fopen("teste123.dat","rb");
     fseek(arq, 0, SEEK_SET); //posiciona no inicio do
    arquivo
    while(!feof(arq)){
       fread(&a, sizeof(a), 1, arq); //le o registro
       if(strcmp(busca,a.nome)){ //comparação de string
       }
       else{
           printf("\n\nRegistro Encontrado!\n Nome: %s -
     Telefone: %s", a.nome, a.tel);
           achei = 1;
       }
    }
    if (achei == 0)
    {
       printf("Registro não encontrado!");
    }
    fclose(arq);
}
```