



Arquivos

Algoritmos e Programação 2 – Ciências da Computação – UFJ
Professora: Ana Paula Freitas Vilela Boaventura

Até então, dados voláteis;
Situações do “mundo real”,
dados permanentes;
Exemplo: Transação
bancária;



O que são arquivos?

- **Coleção de bytes** armazenados em estruturas secundárias: *pendrive*, SSD por exemplo.
- **Estrutura de dados** manipulada **fora** do ambiente do programa;

Arquivos

Atenção!!

- Cuidado! A **extensão** do arquivo **não define o seu tipo**. (Serve para o SO saber qual programa abrirá o arquivo);
- O que define um arquivo é a **maneira como os dados são organizados e as operações** usadas por um programa para ler ou escrever esse arquivo.

Arquivos

O que são arquivos?

- Acontecem **operações** sobre arquivos (abrir, alterar, criar);
- É preciso criar um **vínculo** entre o **programa executável** e um determinado **endereço de memória**, onde está localizado (ou será alocado) o arquivo;

Arquivos

Como manipular arquivos em C?

Biblioteca **stdio.h** tem funções para manipulação de arquivos;

Define novo tipo de dado a ser usado especificamente com arquivos, como o **tipo FILE**.

Uma variável do tipo **ponteiro para FILE** é capaz de identificar um arquivo no disco, direcionando para ele todas as operação;

Arquivos em linguagem C

Quais os tipos de arquivos podemos manipular em C?

Em C/C++, os dados podem ser gravados em arquivos do tipo de **texto** ou **binários**;

Arquivos em linguagem C

Arquivos em linguagem C

Arquivo de texto

Podem ser lidos diretamente por qualquer editor de texto, como .txt e .dat;

A extensão DAT representa um arquivo que contém dados e não está associado a nenhum programa em particular para realizar a sua abertura, pode ser vídeo, música, outros, inclusive é recomendável cautela, pois pode conter arquivos maliciosos)

Arquivos em linguagem C

Arquivo Binário

Um arquivo em C/C++ pode representar diversas coisas, como arquivo de disco, uma impressora, um teclado, ou qualquer dispositivo de entrada e saída;

Para a disciplina, consideramos apenas arquivos de disco, mas caso queiram manipular outros dispositivos, a interface é a mesma;

Arquivos em linguagem C

Arquivo Binário

Qualquer arquivo que não esteja em formato de texto, é considerado um arquivo binário, ou seja, é todo arquivo de computador que não está em formato texto;

Pode ser um programa de computador, arquivo de imagem digital, arquivo de som, arquivo de dados e vários outros arquivos;



Arquivos em linguagem C

Arquivos: Declaração

FILE

Escrito em maiúsculo para reforçar a ideia de que não se trata de um tipo básico da própria linguagem

Sintaxe

```
FILE * <nome_arquivo>;
```

Exemplo

```
FILE *arq; /*arq – é a variável que relacionará o  
programa com o arquivo que será manipulado*/
```



Arquivos em linguagem C

Arquivos: Abertura

Sabendo que...

Um ponteiro aponta para “alguém”;
Só ter referência (endereço de memória onde o ponteiro aponta não serve para muita coisa);
Fazemos o uso da função fopen;

Arquivos: Abertura

Sintaxe

fopen (char *nome_arquivo, char *modo);

f:FILE **open:** abertura

char *nome arquivo: Endereço do arquivo no seu computador (representado por uma string)

char *modo: Outra string contendo o método de abertura

Arquivos: Abertura

Curiosidades da fopen

Necessário fornecer a localização deste arquivo;

A função `fopen()` não é capaz de criar pastas. Pode, apenas, criar arquivos quando o modo de abertura assim permitir;

Se precisar, use a função `system()`* para criar a pasta;

```
system ("md C:\\exemplo\\teste"); //Windows e barra dupla para evitar confusão entre \n, por exemplo
```

```
system ("mkdir ~p/home/user/exemplo"); //Padrão Unix
```

* `system` pertence à biblioteca `stdlib.h`

Arquivos: Abertura

Modo de abertura (**r**-read, **w**-write, **a**-append)

r	Abre um arquivo de texto para leitura
w	Cria arquivo de texto permitindo a escrita
a	Anexa novos dados a um arquivo de texto
r+	Abre um arquivo de texto para leitura e escrita
w+	Cria arquivo de texto onde poderão ser realizadas operações de leitura e escrita
a+	Anexa novos dados ou cria arquivo de texto para operações de leitura e escrita

Arquivos: Abertura

Modo de abertura (**r**-read, **w**-write, **a**-append)

rb	Abre um arquivo binário para leitura
wb	Cria arquivo binário permitindo a escrita
ab	Anexa novos dados a um arquivo binário
rb+	Abre um arquivo binário permitindo leitura e escrita
wb+	Cria arquivo binário onde pode fazer operações de leitura e escrita
ab+	Anexa novos dados a um arquivo binário para leitura e escrita

Arquivos: Abertura

Modo de abertura (**r**-read, **w**-write, **a**-append)

Modo de abertura

```
FILE *arq;
```

```
arq=fopen("c:\\ arquivo1.txt", "w");
```

Quando a função `fopen()` é utilizada para abrir um arquivo no modo escrita (“w” ou “wb”), duas situações podem ocorrer:

- Se o arquivo não existir, ele será criado;
 - Se o arquivo existir, ele será sobreposto por um arquivo vazio;
-

Arquivos: Abertura

Erros em abertura de arquivos

```
int main()
{ FILE *arq;
  arq=fopen("c:\\arquivo1.txt","w");
  if (arq==NULL) //A variável arq receberá o valor NULL
  { printf("\nOcorreu um erro!\n");
  }
  else
  { //demais instruções do código }
}
```

Cada arquivo aberto possui seu próprio file pointer;
Por exemplo, se um programa vai manipular dois arquivos diferentes arq1 e arq2 simultaneamente (no exemplo abaixo, um para leitura e outro para escrita), dois file pointer devem ser usados:

```
FILE *fp1, *fp2;  
fp1 = fopen (“arq1”, “r”);  
fp2 = fopen (“arq2”, “w”);
```

Arquivos: Abertura

Cada arquivo aberto possui seu próprio file pointer;
Por exemplo, se um programa vai manipular dois arquivos diferentes arq1 e arq2 simultaneamente (no exemplo abaixo, um para leitura e outro para escrita), dois file pointer devem ser usados:

```
FILE *fp1, *fp2;  
fp1 = fopen (“arq1”, “r”);  
fp2 = fopen (“arq2”, “w”);
```

Arquivos: Abertura

- A função **error** () retorna um número inteiro e deve ser chamada logo depois da invocação de outra função de manipulação de arquivo;
- Se o valor retornado for diferente de zero, significa que ocorreu um erro durante a última operação realizada com o arquivo, caso contrário, não ocorreu erro;

Arquivos: Abertura – Capturando erro

Arquivos: Abertura – Capturando erro

```
int ferror (FILE *arq);
```

arq é a referência para o arquivo onde o caractere está escrito;



Arquivos em linguagem C

Quando ocorrer algum erro durante a execução da função, **poderá haver perda de dados ou até mesmo, perda do arquivo**, portanto é necessário **finalizar corretamente a manipulação com arquivos**;

Arquivos: Fechamento

Para saber o final do arquivo (que para o sistema, é uma sequência de bytes), o compilador C vai procurar um sinal, uma constante conhecida por **EOF**, que sinaliza o final do arquivo.

Para identificar o final de um arquivo, podemos usar a função `fclose`, que recebe um ponteiro para o tipo `FILE` e retorna um inteiro:

```
int fclose (FILE *arq); //f-file close- fechar
```

Arquivos: Fechamento

Caso o byte lido represente o EOF, a função “fecha” a abertura do arquivo. Ou seja, libera a memória associado ao ponteiro do FILE.

E caso trabalhe com diversos arquivos, é possível fechar todos de uma vez, através da função:

```
int fcloseall( ); //f-file close- fechar all - tudo
```

Arquivos: Fechamento

```
int main()
{
    FILE *fp;
    int resposta;
    fp=fopen("arquivo.txt","w");
    if (fp==NULL){
        printf("\nErro");
    }
    else {
        printf("Sucesso na abertura!\n");
        resposta = fclose(fp);
        if (resposta==0) {
            printf("\nFechou corretamente;");
        }
    }
}

else
{
    printf("\nErro no fechamento;");
}

return 0;
}
```

Arquivos: Fechamento



Arquivos em linguagem C

Escrever caracteres num arquivo

fputc(char var, FILE *arq);

var é a variável que será escrita no arquivo
arq é a referência para o arquivo onde o caractere está escrito;

Funções de escrita de texto

Escrever cadeia de caracteres num arquivo

fputs(char *cadeia, FILE *arq); //cadeia é uma string, declarada como char cadeia[30], por exemplo
cadeia armazena a cadeia de caracteres que será escrita no arquivo;
arq é a referência para o arquivo onde o caractere está escrito;

Funções de escrita de texto

O comando `fprintf()` nos permite escrever *strings* inteiras em arquivos.

Sintaxe simplificada é:

```
int fprintf(FILE *arq, char string[ ]);
```

Recebe o local onde deve direcionar a saída (para um arquivo, apontado pelo ponteiro `arq` do tipo `FILE`) e a *string* que devemos adicionar em tal arquivo.

Essa função retorna `EOF` em caso de erro.

Funções de escrita de texto

Exemplo

Escreva um programa em C que peça 3 notas de um aluno (Matemática, Física e Química) e salve estes dados num arquivo denominado “notas.txt”, que deve ter, ao final, a média das três disciplinas.

Funções de escrita de texto

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    char url[]="notas.txt";
```

```
    float nota,
```

```
        media=0.0;
```

```
    FILE *arq;
```

```
    arq = fopen(url, "w");
```

```
    if(arq == NULL)
```

```
        printf("Erro, nao foi possivel abrir o arquivo\n");
```

```
    else{
```

```
        printf("Nota de Matematica: ");
        scanf("%f", &nota);
        fprintf(arq, "Matematica: %.2f\n", nota);
        media+=nota;
```

```
        printf("Nota de Fisica: ");
        scanf("%f", &nota);
        fprintf(arq, "Fisica: %.2f\n", nota);
        media+=nota;
```

```
        printf("Nota de Quimica: ");
        scanf("%f", &nota);
        fprintf(arq, "Quimica: %.2f\n", nota);
        media+=nota;
```

```
        media /= 3;
```

```
        fprintf(arq, "Media final: %.2f\n", media);
```

```
    }
```

```
    fclose(arq);
```

Funções de escrita de texto

- Arquivos em C/C++ não podem ser associados a um tipo primitivo de dados ou a um registro (struct);
- Os arquivos simplesmente armazenam uma **sequência de caracteres** ou uma **sequência de bytes**.
- Entretanto, em vários momentos, é necessário ler parte do conteúdo de um arquivo e gravar diretamente numa variável int, float ou mesmo num registro.
- Também é importante conseguir pegar um conteúdo de variáveis desses tipos e gravá-los diretamente num arquivo;

Arquivos Binários - Escrita

- Sendo **necessário** trabalhar com **arquivos binários**;
- Toda vez que uma operação de leitura ou escrita for realizada, deverá ser informado o **número de bytes** que serão lidos ou gravados nos bytes uma variável (de qualquer tipo, incluindo struct) ocupa;
- A função **fwrite()** faz parte do arquivo `stdio.h` e é responsável por escrever um bloco de bytes existente em memória para um **arquivo aberto em modo binário**;

Arquivos Binários - Escrita

Escrever cadeia de caracteres num arquivo

fwrite(void *mem, int qtdbytes, int cont, FILE *arq);

mem: representa a variável que armazena o conteúdo a ser gravado no arquivo;

qtdbytes: total em bytes que será escrito no arquivo;

cont: representa o número de blocos de tamanho qtd bytes que serão escritos no arquivo;

arq: referência para o arquivo;

Arquivos Binários - Escrita

```
FILE *cli;
struct teste{
    int numero;
    char nome[30];
}c;

cli=fopen("cliente.txt","ab+");
if (cli==NULL)
    printf("\Erro");
else
{
    printf("\nNúmero do cliente:");
    scanf("%d",&c.numero);
    printf("\nNome do cliente:");
    scanf(" %[^\\n]s",c.nome);
}
fwrite(&c,sizeof(struct cliente),1,cli);

if (ferror(cli))
    printf("Erro\\n");
else
{
    printf("Gravação OK");
    fclose(cli);
}
}
```

Arquivos Binários - Escrita

```
#include <stdio.h>
main() {
    FILE *arq;
    // Esses dados vão ser gravados !
    int ret, vet[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
    // arquivo alvo
    char nomearq[] = "vet.dat";
    // arquivo tem que ter permissão w para escrita e b para abrir
    como binario
    arq = fopen(nomearq, "wb");
    if (arq != NULL) {
        // aqui é feita a escrita !!
        ret = fwrite(vet, sizeof(int), 10, arq);
        if (ret == 10)
            printf("Gravacao com sucesso\n");
        else
            printf("Foram gravados apenas %d elementos\n", ret);
        fclose(arq);
    }
    else
        puts("Erro: criacao do arquivo");
}
```

Arquivos Binários - Escrita

```
#include <stdio.h>
int na = 6;
typedef struct {
    char nome[10];
    int nota;
} tp_aluno;
main() {
    tp_aluno alunos[] = {{ "Luiz", 5}, {"Paulo", 5}, {"Maria", 3},
                          {"Luiza", 4}, {"Felipe", 8}, {"Fabiana", 6}};

    int ret;
    FILE *arq;
    char nomearq[] = "turma.dat";
    arq = fopen(nomearq, "wb");
    if (arq != NULL) {
        ret = fwrite(alunos, sizeof(tp_aluno), na, arq);
        if (ret == na)
            printf("Gravacao %d registros com sucesso\n", ret);
        else
            printf("Foram gravados apenas %d elementos\n", ret);
        fclose(arq);
    }
}
```

Arquivos Binários - Escrita



Arquivos em linguagem C

Ler caracteres num arquivo

fgetc(FILE *arq);

arq é a referência para o arquivo onde o caractere está escrito;

Funções de leitura de texto

Ler cadeia de caracteres num arquivo

fgets(char *cadeia, int tam, FILE *arq); //cadeia é uma string, declarada como char cadeia[30], por exemplo

cadeia armazena a cadeia de caracteres que será escrita no arquivo;

tam indica que a quantidade máxima de caracteres lidos será tam-1;

arq é a referência para o arquivo onde o caractere está escrito;

Funções de leitura de texto

Particularmente interessante se tivermos um arquivo com um determinado formato. Em outras palavras, quando o conteúdo do arquivo obedecer um determinado padrão.

Sintaxe:

```
int fscanf(FILE *arq, char *string formatada);
```

Funções de leitura de texto

Por exemplo, vamos supor que haja uma lista com notas de 3 alunos;

Na primeira coluna as notas de Matemática, na segunda as de Física e na terceira coluna as notas de Química;

Cada linha é representada como: “%f %f %f \n”

Ou seja: número, espaço em branco, número, espaço em branco, número e new line. Este é o formato, este é o padrão.

Funções de leitura de texto

Lendo cadeia de caracteres de um arquivo

Exemplo

Em um arquivo chamado "notas.txt" está os dados dos nomes e notas de alunos:

Maria 8 8 10

Jose 6 6 8

Carlos 7 9.5 7.5

Programador 10 10 10

Crie um programa que exiba o nome de cada aluno e sua média.

Padrão: %s %f %f %f \n

Funções de leitura de texto

Comece
aqui

```
        else
            while( (fscanf(arq,"%s %f %f %f\n", nome, &nota1, &nota2, &nota3))!=EOF )
                printf("%s teve media %.2f\n", nome, (nota1+nota2+nota3)/3);

        fclose(arq);

        return 0;
#include <stdio.h>

int main(void)
{
    char url[]="notas.txt",
        nome[20];
    float nota1, nota2, nota3;
    FILE *arq;

    arq = fopen(url, "r");
    if(arq == NULL)
        printf("Erro, nao foi possivel abrir o arquivo\n");
```

Funções de leitura de texto

A função `fread()` pode ler qualquer tipo de dados e não apenas caracteres ou cadeias de caracteres.

Sintaxe:

`fread(void *mem, int qtdbytes, int tam, FILE *arq);`

`mem`: representa a variável que vai receber o conteúdo lido no arquivo;

`qtdbytes`: tamanho do bloco, em bytes, que será lido do arquivo;

`tam`: número de blocos (tamanho qtd bytes) que escreveremos;

`arq`: referência para o arquivo;

Arquivos Binários - Leitura

```

cli=fopen("cliente.txt","ab+");
if (cli==NULL)
    printf("\Erro");
else
{
    fread(&c,sizeof(struct
        cliente),1,cli);
    if (ferror(cli))
        printf("Erro na leitura\n");
    else
        printf("leitura....\n");
    while(!feof(cli))
    {
        printf("\n%d",c.numero);
        printf("\n%s\n\n",c.nome);
        fread(&c,sizeof(struct
            cliente), 1, cli);
        if (ferror(cli))
            printf("Erro\n");
        else
            printf("Ok.\n");
    }
    fclose(cli);
}

```

Arquivos Binários - Leitura

```
#include <stdio.h>
main() {
    FILE *arq;
    int i, ret, vet[10];
    char nomearq[] = "vet.dat";
    arq = fopen(nomearq, "rb");
    if (arq != NULL) {
        // estou recuperando AQUI
        ret = fread(vet, sizeof(int), 10, arq);
        if (ret == 10) {
            printf("Elementos: ");
            for (i = 0; i < 10; i++)
                printf("%d ", vet[i]);
        }
        else
            printf("Foram lidos apenas %d elementos\n", ret);
        fclose(arq);
    }
    else
        puts("Erro: abertura do arquivo");
}
```

Arquivos Binários - Leitura



Funções interessantes...

“Descobre” se o fim de arquivo foi encontrado

```
int feof (FILE *arq);
```

arq é a referência para o arquivo onde o caractere está escrito;

Encontrar fim de arquivo

Muda a posição do cursor sem que haja necessidade de leitura ou escrita no arquivo

int fseek (FILE *arq, long qtd_bytes, int posicao);

arq é a referência para o arquivo onde o caractere está escrito;

qtd_bytes representa a quantidade de bytes que o cursor será movimentado a partir zero (onde está o primeiro byte do arquivo);

Posicao:

SEEK_SET (movimenta a partir da posição inicial do arquivo)

SEEK_CUR (movimenta a partir da posição atual do cursor)

SEEK_END (movimenta a partir da posição final do arquivo)

Reposicionar o cursor de um arquivo

Remove o arquivo

Lembrem-se de fechar o arquivo antes de removê-lo

```
int remove (FILE *arq);
```

arq é a referência para o arquivo onde o caractere está escrito;

Apagar um arquivo

Renomeia o arquivo

```
int rename (FILE *arq);
```

arq é a referência para o arquivo onde o caractere está escrito;

Renomear um arquivo

- Canal: Programação descomplicada linguagem C
 - <https://youtu.be/LNu-0bzxpos> (Aula 66: Arquivos - Parte 1 - Conceitos)
 - <https://youtu.be/ueg-IE8cZH4> (Aula 67 : Arquivos - Parte 2 - Arquivos texto e binário)
 - https://youtu.be/uYymG_oUPeY (Aula 68 - Arquivos - Parte 3 - Abrir e Fechar)
 - <https://youtu.be/X6BcBhRCR8M> (Aula 69 - Arquivo - Parte 4 - FPUTC())
 - <https://youtu.be/FwW2T3jGvdg> (Aula 70 - Arquivos - Parte 5 – fgetc)
 - <https://youtu.be/WdZv1gCpDjg> Aula 71 - Arquivo - Parte 6 - Exemplo: Trabalhando com Arquivos
 - <https://youtu.be/xN61MLUgkSg> (Aula 72 - Arquivo - Parte 7 – EOF)
 - <https://youtu.be/ODjgyg6WbPk> (Aula 73 - Arquivos Parte 8 – fputs)
 - https://youtu.be/GDVPYnD-T_w (Aula 74 - Arquivos Parte 9 – fgets)
 - <https://youtu.be/rBnZTxbWqZQ> (Aula 75 - Arquivos Parte 10 - fwrite)
 - <https://youtu.be/ZxuacsaCdaI> Aula 76 - Arquivos Parte 11 - fread
 - <https://youtu.be/4WlsKHHVda0> Aula 77 - Arquivos Parte 12 – fprintf
 - <https://youtu.be/jnotzdaKjOI> Aula 78 - Arquivos Parte 13 - fscanf
 - <https://youtu.be/cdXGEy-6jMU> Aula 79 - Arquivos Parte 14 - fseek e rewind

Assistam aos vídeos
