

Algoritmos e Estruturas de Dados

Disciplina 301477

Programa de Pós-graduação em
Computação Aplicada

Prof. Alexandre Zaghetto
<http://alexandre.zaghetto.com>
zaghetto@unb.br



Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

O presente conjunto de *slides* não pode ser reutilizado ou republicado sem a permissão do instrutor.

Módulo 11

Arquivos

1. Introdução

- Os comandos de entrada e saída que usamos até o momento foram:
 - ✓ printf – mostra dados formatados no vídeo; e
 - ✓ scanf – lê dados formatados digitados do teclado.
- Todavia, dados podem ser lidos e gravados em arquivos.
- A cada arquivo está associado um nome, pelo qual o mesmo é conhecido externamente, isto é, o nome que consta no diretório do disco.
- Uma vez que um arquivo é uma sequência de *bytes* temos o marcador do final desse arquivo que é: EOF (EndOfFile)

1. Introdução

- Vamos tratar de dois tipos de arquivos:
 - ✓ **TEXTO**, onde são gravados caracteres e pode ser editado por um editor de texto.
 - ✓ **BINÁRIO**, onde são gravados dados como estão na memória. Por exemplo, uma variável inteira é gravada com 4 *bytes* com o conteúdo exato que está na memória.

2. Abertura de Arquivos

- Para tratar de arquivos, a linguagem C fornece um nível de abstração entre o programador e o dispositivo que está sendo acessado para gravação e leitura:

FILE *fp;

- Ou seja, um ponteiro *fp*, do tipo FILE.

2. Abertura de Arquivos

- Para realizar a abertura de um arquivo para leitura ou gravação, temos a função ***fopen*** (file Open), que possui dois parâmetros:

```
FILE *fopen(const char *filename, const char *mode);
```

- ✓ **filename**: o nome do arquivo (string); e
- ✓ **mode**: modo de abertura do arquivo.

2. Abertura de Arquivos

- Modos possíveis:
 - ✓ **r**: abre um arquivo **TEXTO** para **leitura**.
 - ✓ **w**: cria um arquivo **TEXTO** para **gravação**, ou se o arquivo já existe, elimina seu conteúdo e recomeça a gravação a partir do seu início.
 - ✓ **a**: abre um arquivo **TEXTO** já existente para gravação, a partir de seu final.

 - ✓ **rb**: abre um arquivo **BINÁRIO** para **leitura**.
 - ✓ **wb**: cria um arquivo **BINÁRIO** para **gravação**, ou se o arquivo já existe, elimina seu conteúdo e recomeça a gravação a partir do seu início.
 - ✓ **ab**: abre um arquivo **BINÁRIO** já existente para gravação, a partir de seu final.

2. Abertura de Arquivos

- Modos possíveis:
 - ✓ **r+**: abre um arquivo **TEXT**O para **leitura e gravação**. O arquivo deve existir e pode ser modificado.
 - ✓ **w+**: cria um arquivo **TEXT**O para **leitura e gravação**. Se o arquivo existir, o conteúdo anterior será destruído. Se não existir, será criado.
 - ✓ **a+**: abre um arquivo **TEXT**O para **leitura e gravação**. Os dados serão adicionados no fim do arquivo se ele já existir, ou um novo arquivo será criado, no caso do arquivo não existir.

2. Abertura de Arquivos

- Modos possíveis:
 - ✓ **r+b**: abre um arquivo **BINÁRIO** para **leitura e gravação**. O mesmo que "r+" acima, só que o arquivo é binário.
 - ✓ **w+b**: cria um arquivo **BINÁRIO** para **leitura e gravação**. O mesmo que "w+" acima, só que o arquivo é binário.
 - ✓ **a+b**: acrescenta dados ou cria um arquivo **BINÁRIO** para **leitura e gravação**. O mesmo que "a+" acima, só que o arquivo é binário.

2. Abertura de Arquivos

- Exemplo:

```
FILE *pFile;  
char nomeArquivo[] = "c:\\arquivo.txt";
```

```
pFile = fopen(nomeArquivo, "w");
```

- Caso a função **fopen** não encontre o arquivo indicado, ela retorna NULL.

```
FILE *pFile;  
  
pFile = fopen("teste.txt", "w");  
if (pFile == NULL) {  
    printf("Falha.\n");  
    exit(1);  
}
```

3. Fechamento de Arquivos

- Da mesma forma que abrimos um arquivo utilizando a função *fopen*, devemos fechá-lo quando não formos mais utilizá-lo, pois assim realmente garantimos que o arquivo será salvo em disco, e não ficará simplesmente no buffer (região de memória).
- Para isso utilizamos o comando ***fclose***, da seguinte forma:

fclose(pFile);

4. Arquivos Texto

- Para fazermos a leitura e gravação, podemos utilizar as funções ***fscanf*** e ***fprintf***, respectivamente.
- Por exemplo, se queremos ler números inteiros de um arquivo:

```
FILE *pFile  
int numero;  
pFile = fopen("teste.txt", "r");  
fscanf(fp, "%d", &numero);
```

- Para gravar no arquivo:

```
FILE *pFile  
int numero = 10;  
pFile = fopen("teste.txt", "w");  
fprintf(fp, "%d", numero);
```

4. Arquivos Texto

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

int main ()
{
    FILE *pFile;
    int numero;

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

    if(pFile==NULL) {
        printf("Cannot open file.\n");
        exit(1);
    }
```

4. Arquivos Texto

```
do {  
    printf("Digite um numero inteiro positivo:");  
    scanf("%d", &numero);  
    if(numero != -1) fprintf(pFile, "%d\n", numero);  
} while (numero != -1);  
  
fclose(fp);  
return 0;  
}
```

4. Arquivos Texto

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

int main()
{
    FILE *pFile;
    int n;
    char name [100];

    pFile = fopen ("myfile.txt", "w");
    for (n=0 ; n<3 ; n++)
    {
        puts ("Please, enter a name: ");
        gets (name);
        fprintf (pFile, "Name %d [%-10.2s]\n", n, name);
    }

    fclose (pFile);
    return 0;
}
```


4. Arquivos Texto

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

int main()
{
    FILE *pFile;
    int numero;

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

    if(pFile==NULL) {
        printf("Cannot open file.\n");
        exit(1);
    }
```



4. Arquivos Texto

```
fscanf (pFile, "%d", &numero);

while (!feof(pFile)) {
    printf("%d\n", numero); /* print on screen */
    fscanf(pFile, "%d", &numero);
}

fclose(fp);
return 0;
}
```

4. Arquivos Texto

- putc()

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

```
int main()
{
    FILE *pFile;
    char ch;
```

4. Arquivos Texto

- `putc()`

```
if ((pFile=fopen("teste.txt", "w"))==NULL) {  
    printf("Cannot open file.\n");  
    exit(1);  
}  
  
do {  
    ch = getchar();  
    putc(ch, pFile);  
} while (ch != '$');  
  
fclose(pFile);  
return 0;  
}
```

4. Arquivos Texto

- `getc()`

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

```
int main()
{
    FILE *pFile;
    char ch;
```

4. Arquivos Texto

- `getc()`

```
if ((pFile = fopen("teste.txt", "r")) == NULL) {  
    printf("Cannot open file.\n");  
    exit(1);  
}  
  
ch = getc(pFile); /* read one character */  
while (!feof(pFile)) {  
    putchar(ch); /* print on screen */  
    ch = getc(pFile);  
}  
  
fclose(pFile);  
return 0;  
}
```

4. Arquivos Texto

- fputs()

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

int main()
{
    char str[80];
    FILE *pFile;

    if((pFile = fopen("teste.txt", "w"))==NULL) {
        printf("Cannot open file.\n");
        exit(1);
    }
```

4. Arquivos Texto

- `fputs()`

```
do {  
    printf("Digite string (ENTER => fim):\n");  
    gets(str);  
    strcat(str, "\n"); /* add a newline */  
    fputs(str, pFile);  
} while(*str!='\n');  
  
    return 0;  
}
```


4. Arquivos Texto

- `fgets()` e `rewind()`

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

int main()
{
    char str[80];
    FILE *pFile;

    if((pFile = fopen("teste.txt", "w+"))==NULL) {
        printf("Cannot open file.\n");
        exit(1);
    }
```

4. Arquivos Texto

- `fgets()` e `rewind()`

```
do {  
    printf("Enter a string (CR to quit):\n");  
    gets(str);  
    strcat(str, "\n");  
    fputs(str, pFile);  
} while(*str!='\n');  
  
rewind(pFile);  
  
while(!feof(pFile)) {  
    fgets(str, 80, pFile);  
    printf("%s", str);  
}
```

4. Arquivos Texto

- `fgets()` e `rewind()`

```
fclose(fp);  
return 0;  
  
}
```



5. Arquivos Binários

- fwrite()

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

int main()
{
    FILE *pFile;
    int buffer[] = {1,2,3};

    pFile = fopen ("myfile.bin" , "wb");
    fwrite (buffer , sizeof(int), 3, pFile );

    fclose (pFile);

    return 0;
}
```

5. Arquivos Binários

- fread()

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

int main()
{
    FILE *pFile;
    double d = 12.23;
    int i = 101;

    if((pFile=fopen("test.bin", "wb+"))==NULL) {
        printf("Cannot open file.\n");
        exit(1);
    }

    fwrite(&d, sizeof(double), 1, pFile);
    fwrite(&i, sizeof(int), 1, pFile);
```

5. Arquivos Binários

- fread()

```
rewind(pFile);

fread(&d, sizeof(double), 1, pFile);
fread(&i, sizeof(int), 1, fp pFile

printf("%.2f %d \n", d, i);

fclose(pFile);

return 0;

}
```