# *Arquivos*

Prof. Angelo Loula
UEFS



## Arquivos

- Estruturas de dados armazenadas fora da memória principal do computador
  - Memória secundária (HD, disquete etc);
  - Maior quantidade de informação
  - Uso futuro (persistência)
  - Nome e caminho para identificação



## Arquivos

- Arquivos Texto
  - Composto por uma sequência de caracteres organizados por linhas (ex. código fonte na linguagem C).
- Arquivos Binários
  - Composto por uma sequência de bits, da mesma forma como fica na memória principal (ex. programa executável).



## Arquivos

- Arquivos Texto e Entrada/Saída Padrão


# Arquivos

- Arquivos de Texto no C (**stdio.h**)
  - tipo **FILE** para representar um arquivo

```
FILE *arquivo;
```

  - mas qual arquivo será usado?
    - função **fopen** recebe o nome e abre o arquivo

```
arquivo = fopen("meutexto.txt","w");
arquivo = fopen("c:\doc\arq.c","w");
```

- **stdin** e **stdout**



# Arquivos

- Abertura de um arquivo
  - *ponteiroarq* **= fopen(***nomeexterno, modo***);**
  - Modos de Abertura:
    - **r** : Abre o arquivo somente para leitura. Se o arquivo não existir, dá erro.
    - **w** : Abre o arquivo somente para escrita. SEMPRE cria um novo arquivo. Portanto, se o arquivo já existir ele é apagado.
    - **a** : Abre o arquivo para leitura/escrita. Escreve apenas no final do arquivo. Caso o arquivo já exista, ele abre, caso contrário ele cria o arquivo.
    - **r+** : Abre o arquivo para leitura/escrita. Se o arquivo não existir, dá erro.
    - **w+** : Abre o arquivo para leitura/escrita. SEMPRE cria um novo arquivo. Portanto, se o arquivo já existir ele é apagado.

# Arquivos

- Fechamento de um arquivo

```
fclose (ponteiroarquivo);
```

- Comandos de leitura:

```
fscanf (ponteiroarquivo, …);
```

igual ao *scanf*, acrescentando o ponteiro para arquivo no começo.
retorna núm. de var. lidas ou EOF
Obs: somente para arquivos abertos com leitura

- Comandos de escrita

```
fprintf (ponteiroarquivo, …);
```

igual ao *printf*, acrescentando o ponteiro para arquivo no começo.
retorna núm. de caracteres escritos ou inteiro negativo
Obs: somente para arquivos que abertos com escrita



```
#include <stdio.h>
int main ( ){
  FILE *arq ;

  /* abrir arquivo para escrita*/
  arq = fopen ( "teste.txt", "w") ;

  /* escrever um texto no arquivo */
  fprintf( arq, "Testando escrita.");

  /* fechar arquivo */
  fclose( arq ) ;
}
```

```c
#include <stdio.h>
int main(){
  char texto[100];
  FILE *leitura, *escrita;
  leitura = fopen("arq1.txt", "r");
  if( leitura ==0 ){
    printf("Nao abriu o arq1.txt");
  } else {
    escrita = fopen("arq2.txt", "w");
    if( escrita ==0 ){
      printf("Nao abriu o arq2.txt");
    } else {
      fscanf(leitura, "%s", &texto);
      fprintf(escrita, "%s", texto);
      fclose(leitura);
      fclose(escrita);
}}}
```



```c
#include <stdio.h>
int main ( ){
 int i, n, vet[100];
 FILE *arq ;

 printf ("Digite a quantidade: ");
 scanf("%d", &n);
 printf ("Digite os elementos: ");
 for ( i=0; i<n; i++){
   scanf ("%d" , &vet[i] );
 }
 arq = fopen ("vetor1.txt" , "w" ) ;
 fprintf (arq, "%d ", n) ;
 for ( i=0; i<10; i++){
   fprintf ( arq, "%d ", vet[i]);
 }
 fclose(arq) ;
}
```

note o espaço em branco!



```c
#include <stdio.h>
int main ( ){
 int i, n, vet[100];
 FILE *arq ;

 arq = fopen ( "vetor1.txt", "r" ) ;
 fscanf ( arq, "%d", &n) ;
 for ( i=0; i<n; i++){
   fscanf ( arq, "%d", &vet[i]);
 }
 fclose( arq ) ;
 printf("Foram lidos %d elementos:", n);
 for ( i=0; i<n; i++){
    printf( "%d ", vet[i]);
 }
}
```



```c
#include <stdio.h>
int main(){
 FILE *arq;
 char str[80],c;
 printf("Digite o nome do arquivo:\n");
 gets(str);
 arq = fopen(str,"w");
 fprintf(p,"Este arquivo chama-se:\n%s\n",str);
 fclose(p);
 arq = fopen(str,"r");
 fscanf(p,"%c",&c);
 while(feof(p)==0) {
   printf("%c",c);
   fscanf(p,"%c",&c);
 }
 fclose(p);
}
```

`feof(`*ponteiroarquivo*`)` é uma função que retorna 0 se ainda não foi atingido o final do arquivo, e um número diferente de 0, caso contrário.
(Cuidado com esta função, veja: http://www.gidnetwork.com/b-58.html)

# Arquivos Texto

- Outros comandos:

**`char * fgets (char *str, int n, FILE *f);`** - Lê *n* caracteres de *f* e guarda em *str*, retorna nulo ou *str*

**`int fputs ( const char * str, FILE * f);`** - Escreve *str* em *f*, retorna EOF ou valor não-negativo

**`int ferror ( FILE * stream );`** - indica se houve erro n última operação

**`EOF`** – int especial de final de arquivo

**`int fgetc ( FILE * stream );`** - lê caracter

**`int fputc (int character, FILE * stream);`** - escreve caracter



# Arquivos Binários

- Orientado a bytes e não caracteres
- Abertura de um arquivo
  - *`ponteiroarq`* **`= fopen(`***`nomeexterno, modo`***`);`**
  - Modos de Abertura:
    - **`b`** : Indica arquivo binário.
    - Combinando: **`rb,wb`**,**`ab,rb+, wb+`**
- Fechamento:
  - **`fclose`**



# Arquivos Binários

- **`size_t fwrite(const void *ptr, size_t size, size_t count, FILE *f );`** - Escreve em *f* um vetor de *count* elementos, cada um com tamanho *size* em bytes, do bloco de memória apontado por *prt*. Retorna número de elementos escrito.
- **`size_t fread(void *ptr, size_t size, size_t count, FILE * f);`**
- **`int fseek(FILE *f, long int offset, int origin);`** - Coloca o indicador de posição de *f* na posição definida pela adição de *offset* a posição de referência indicada por *origin* (`SEEK_SET, SEEK_CUR, SEEK_END`).



```
  FILE *arq ;
  int vet[] = { 10 , 20, 30 };
  int i;

  arq = fopen ("dados.bin", "wb");
  fwrite (vet, 1 , sizeof(vet), arq);
  fclose (pFile);

  arq = fopen ("dados.bin", ”rb” );
  fread (vet, 1 , sizeof(vet), arq);
 fclose( arq ) ;
  for ( i=0; i<n; i++){
    printf( ”%d ”, vet[i]);
  }
}
```