

# Sistemas Operacionais Embarcados

## Acesso a Arquivos

# Arquivos no Linux

- **Arquivos regulares:** tipo comum
  - **Texto:** os bytes do arquivo devem ser interpretados por uma tabela de caracteres (geralmente ASCII)
  - **Binário:** bytes interpretados por outros programas
- **Diretórios:** separadores de arquivos
- **Dispositivos:** componentes instaláveis de hardware (placas de vídeo, som, CD-ROM, hardware USB, memória RAM etc.)
  - Dispositivo de bloco: usa buffer para leitura/gravação (HD, CD-ROM etc.)
  - Dispositivo de caractere: não usa buffer (impressoras, mouse etc.)

# Arquivos no Linux

- **Links:** atalhos (referências a outros arquivos)
  - Simbólicos: referência pelo endereço lógico
  - Absolutos: referência pelo endereço físico
- **FIFO:** canal de comunicação entre processos

# stdio.h

- Biblioteca padrão C de entrada e saída

```
#include <stdio.h>

int main(int argc, const char * argv[])
{
    FILE *fp;
    fp = fopen("exemplo.bin", "wb");
    if(fp==NULL)
        printf("Erro na abertura do arquivo.");
    fclose(fp);
    return 0;
}
```

Code/04\_File\_stdio/Ex1.c

# stdio.h

- Biblioteca padrão C de entrada e saída

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

```
int main(int argc, const char * argv[])  
{
```

```
    FILE *fp;
```

```
    fp = fopen("exemplo.bin", "wb");
```

```
    if(fp==NULL)
```

```
        printf("Erro na abertura do arquivo");
```

```
    fclose(fp);
```

```
    return 0;
```

```
}
```

Ponteiro para arquivo

Guarda referências do arquivo aberto, do ponto onde ele está aberto etc.

Code/04\_File\_stdio/Ex1.c


# stdio.h

- Biblioteca padrão C de entrada e saída

```
#include <stdio.h>

int main(int argc, const char * argv[])
{
    FILE *fp;
    fp = fopen("exemplo.bin", "wb");
    if(fp==NULL)
        printf("Erro na abertura do arquivo.");
    fclose(fp);
    return 0;
}
```

Abra o arquivo  
“exemplo.bin” para escrita  
“w” em modo binário “b”



Code/04\_File\_stdio/Ex1.c

# stdio.h

- Bibliotec

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

```
int main(int argc, char *argv[])  
{
```

```
    FILE *fp;
```

```
    fp = fopen("arquivo.txt", "w");
```

```
    if(fp == NULL)
```

```
    {
```

```
        printf("Erro ao abrir o arquivo\n");
```

```
        return 1;
```

```
    }
```

```
    return 0;
```

```
}
```

Código

Modo	Significado
w	Abre arquivo para escrita Se já existe, apaga o conteúdo Se não existe, cria o arquivo.
r	Abre arquivo para leitura O arquivo deve existir.
a	Abre arquivo para escrita Se já existe, não apaga o conteúdo Se não existe, cria o arquivo.
+	Usado em conjunto com "r" ou com "a", abre um arquivo já existente para leitura e escrita, sem apagar o conteúdo  Usado em conjunto com "w", abre um arquivo para leitura e escrita, apagando o conteúdo
b	Abre o arquivo em formato binário. Se omitido, abre o arquivo em formato de texto

para escrita  
binário "b"

# stdio.h

- Biblioteca padrão C de entrada e saída

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

```
int main(int argc, const char * argv[])  
{
```

```
    FILE *fp;
```

```
    fp = fopen("exemplo.bin", "wb");
```

```
    if(fp==NULL)
```

```
        printf("Erro na abertura do arquivo!");
```

```
    fclose(fp);
```

```
    return 0;
```

```
}
```

Se o arquivo existir, apaga  
o conteúdo

Se não existir, é criado

Code/04\_File\_stdio/Ex1.c



# stdio.h

- Biblioteca padrão C de entrada e saída

```
#include <stdio.h>

int main(int argc, const char * argv[])
{
    FILE *fp;
    fp = fopen("exemplo.bin", "wb");
    if(fp==NULL)
        printf("Erro na abertura do arquivo.");
    fclose(fp);
    return 0;
}
```

Code/04\_File\_stdio/Ex1.c

Se for retornado o ponteiro nulo (“NULL”), houve algum problema na criação ou abertura do arquivo

Por exemplo, pelo usuário não ter permissão para isso

# stdio.h

- Biblioteca padrão C de entrada e saída

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

```
int main(int argc, const char * argv[])  
{
```

```
    FILE *fp;
```

```
    fp = fopen("exemplo.bin", "wb");
```

```
    if(fp==NULL)
```

```
        printf("Erro na abertura do arquivo");
```

```
    fclose(fp);
```

```
    return 0;
```

```
}
```


Devemos liberar o acesso  
ao arquivo quando  
terminamos

Code/04\_File\_stdio/Ex1.c

```
#include <stdio.h>
#include <stdlib.h>
void erro_fopen(FILE *fp)
{
    if(fp==NULL)
    {
        printf("Erro na abertura do arquivo.\n");
        printf(" Fim de programa.\n");
        exit(-1);
    }
}
int main(int argc, const char * argv[])
{
    FILE *fp;
    fp = fopen("exemplo.bin","wb");
    erro_fopen(fp);
    printf("Arquivo aberto com sucesso.\n");
    fclose(fp);
    return 0;
}
```

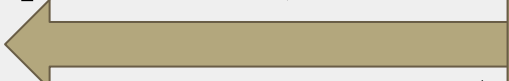
```
#include <stdio.h>
#include <stdlib.h>
void erro_fopen(FILE *fp)
{
    if(fp==NULL)
    {
        printf("Erro na abertura do arquivo.\n");
        printf(" Fim de programa.\n");
        exit(-1);
    }
}
int main(int argc, const char * argv[])
{
    FILE *fp;
    fp = fopen("exemplo.bin","wb");
    erro_fopen(fp);
    printf("Arquivo aberto com sucesso.\n");
    fclose(fp);
    return 0;
}
```

Novamente, abrimos o  
arquivo “exemplo.bin”  
para escrita em formato  
binário



```
#include <stdio.h>
#include <stdlib.h>
void erro_fopen(FILE *fp)
{
    if(fp==NULL)
    {
        printf("Erro na abertura do arquivo.\n");
        printf(" Fim de programa.\n");
        exit(-1);
    }
}
int main(int argc, const char * argv[])
{
    FILE *fp;
    fp = fopen("exemplo.bin","wb");
    erro_fopen(fp);
    printf("Arquivo aberto com sucesso.\n");
    fclose(fp);
    return 0;
}
```

Vamos conferir se o  
arquivo foi aberto  
corretamente na função  
“erro\_fopen( )”



```
#include <stdio.h>
#include <stdlib.h>
void erro_fopen(FILE *fp)
{
    if(fp==NULL)
    {
        printf("Erro na abertura do arquivo.\n");
        printf(" Fim de programa.\n");
        exit(-1);
    }
}
int main(int argc, const char * argv[])
{
    FILE *fp;
    fp = fopen("exemplo.bin","wb");
    erro_fopen(fp);
    printf("Arquivo aberto com sucesso.\n");
    fclose(fp);
    return 0;
}
```

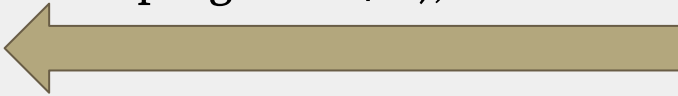
Se for retornado o ponteiro nulo ("NULL"), houve algum problema na criação ou abertura do arquivo

```
#include <stdio.h>
#include <stdlib.h>
void erro_fopen(FILE *fp)
{
    if(fp==NULL)
    {
        printf("Erro na abertura do arquivo.\n");
        printf(" Fim de programa.\n");
        exit(-1);
    }
}
int main(int argc, const char * argv[])
{
    FILE *fp;
    fp = fopen("exemplo.bin","wb");
    erro_fopen(fp);
    printf("Arquivo aberto com sucesso.\n");
    fclose(fp);
    return 0;
}
```

Se executássemos um “return” nesse ponto, somente retornaríamos para este ponto no código

```
#include <stdio.h>
#include <stdlib.h>
void erro_fopen(FILE *fp)
{
    if(fp==NULL)
    {
        printf("Erro na abertura do arquivo.\n");
        printf(" Fim de programa.\n");
        exit(-1);
    }
}
int main(int argc, const char * argv[])
{
    FILE *fp;
    fp = fopen("exemplo.bin","wb");
    erro_fopen(fp);
    printf("Arquivo aberto com sucesso.\n");
    fclose(fp);
    return 0;
}
```


A função “exit( )”  
termina a execução  
do programa a partir  
de qualquer ponto  
do código





```
#include <stdio.h>
#include <stdlib.h>
void erro_fopen(FILE *fp)
{
    if(fp==NULL)
    {
        printf("Erro na abertura do arquivo.\n");
        printf(" Fim de programa.\n");
        exit(-1);
    }
}
int main(int argc, const char * argv[])
{
    FILE *fp;
    fp = fopen("exemplo.bin","wb");
    erro_fopen(fp);
    printf("Arquivo aberto com sucesso.\n");
    fclose(fp);
    return 0;
}
```

A função “exit( )” está definida na biblioteca “<stdlib.h>”




```
#include <stdio.h>
#include <stdlib.h>
void erro_fopen(FILE *fp)
{
    if(fp==NULL)
    {
        printf("Erro na abertura do arquivo.\n");
        printf(" Fim de programa.\n");
        exit(-1);
    }
}
int main(int argc, const char * argv[])
{
    FILE *fp;
    fp = fopen("exemplo.bin","wb");
    erro_fopen(fp);
    printf("Arquivo aberto com sucesso.\n");
    fclose(fp);
    return 0;
}
```

Se o arquivo  
foi aberto  
corretamente,  
continuamos a  
execução do  
código

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, const char * argv[])
{
    FILE *fp;
    char string[100];
    int i;
    fp = fopen("arquivo.txt", "w");
    if(!fp)
    {
        printf( "Erro na abertura do arquivo");
        exit(-1);
    }
    printf("Entre com a string a"
        " ser gravada no arquivo: ");
    gets(string);
    for(i=0; string[i]!='\0'; i++)
        putc(string[i], fp);
    putc('\n', fp);
    fclose(fp);
    return 0;
}
```


```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, const char * argv[])
{
    FILE *fp;
    char string[100];
    int i;
    fp = fopen("arquivo.txt", "w");
    if(!fp)
    {
        printf( "Erro na abertura do arquivo");
        exit(-1);
    }
    printf("Entre com a string a"
        " ser gravada no arquivo: ");
    gets(string);
    for(i=0; string[i]!='\0'; i++)
        putc(string[i], fp);
    putc('\n', fp);
    fclose(fp);
    return 0;
}
```

Abriremos um  
arquivo em  
modo texto



```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, const char * argv[])
{
    FILE *fp;
    char string[100];
    int i;
    fp = fopen("arquivo.txt", "w");
    if(!fp)
    {
        printf( "Erro na abertura do arquivo\n");
        exit(-1);
    }
    printf("Entre com a string a"
           " ser gravada no arquivo: ");
    gets(string);
    for(i=0; string[i]!='\0'; i++)
        putc(string[i], fp);
    putc('\n', fp);
    fclose(fp);
    return 0;
}
```

A extensão “.txt” não importa. Ela só ajuda o sistema operacional a saber qual programa usar para abri-lo



```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, const char * argv[])
{
    FILE *fp;
    char string[100];
    int i;
    fp = fopen("arquivo.txt", "w");
    if(!fp)
    {
        printf( "Erro na abertura do arquivo");
        exit(-1);
    }
    printf("Entre com a string a"
        " ser gravada no arquivo: ");
    gets(string);
    for(i=0; string[i]!='\0'; i++)
        putc(string[i], fp);
    putc('\n', fp);
    fclose(fp);
    return 0;
}
```

A função “gets( )”  
escreve no vetor  
“string” o que o usuário  
digitar no terminal até  
apertar ENTER

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, const char * argv[])
{
    FILE *fp;
    char string[100];
    int i;
    fp = fopen("arquivo.txt", "w");
    if(!fp)
    {
        printf( "Erro na abertura do arquivo\n");
        exit(-1);
    }
    printf("Entre com a string a ser gravada no arquivo: ");
    gets(string);
    for(i=0; string[i]!='\0'; i++)
        putc(string[i], fp);
    putc('\n', fp);
    fclose(fp);
    return 0;
}
```

Ela é bastante perigosa, e foi utilizada aqui a título de exemplo.

O vetor “string” ocupa 100 bytes na memória. Se o usuário digitar mais de 100 caracteres, a função “gets( )” irá sobrescrever memória

<https://www.youtube.com/watch?v=1S0aBV-Waao>

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, const char * argv[])
{
    FILE *fp;
    char string[100];
    int i;
    fp = fopen("arquivo.txt", "w");
    if(!fp)
    {
        printf( "Erro na abertura do arquivo");
        exit(-1);
    }
    printf("Entre com a string a"
        " ser gravada no arquivo: ");
    gets(string);
    for(i=0; string[i]!='\0'; i++)
        putc(string[i], fp);
    putc('\n', fp);
    fclose(fp);
    return 0;
}
```

*Ignorando este problema...*  
o final da string é indicada  
pelo caractere '\0'

Vamos escrever cada  
caractere escrito pelo  
usuário no arquivo aberto



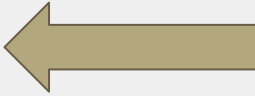
```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, const char * argv[])
{
    FILE *fp;
    char string[100];
    int i;
    fp = fopen("arquivo.txt", "w");
    if(!fp)
    {
        printf( "Erro na abertura do arquivo");
        exit(-1);
    }
    printf("Entre com a string a"
        " ser gravada no arquivo: ");
    gets(string);
    for(i=0; string[i]!='\0'; i++)
        putc(string[i], fp);
    putc('\n', fp);
    fclose(fp);
    return 0;
}
```

Vamos colocar uma última  
quebra de linha no arquivo e  
fecha-lo

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, const char * argv[])
{
    FILE *fp;
    char c;
    fp = fopen("arquivo.txt", "r");
    if(!fp)
    {
        printf( "Erro na abertura do
arquivo");
        exit(-1);
    }
    c = getc(fp);
    while( c != EOF )
    {
        printf("%c", c);
        c = getc(fp);
    }
    fclose(fp);
    return 0;
}
```

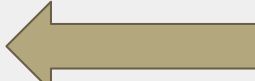
```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, const char * argv[])
{
    FILE *fp;
    char c;
    fp = fopen("arquivo.txt", "r");
    if(!fp)
    {
        printf( "Erro na abertura do
arquivo");
        exit(-1);
    }
    c = getc(fp);
    while( c != EOF )
    {
        printf("%c", c);
        c = getc(fp);
    }
    fclose(fp);
    return 0;
}
```

Vamos ler o  
arquivo criado no  
exemplo anterior



```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, const char * argv[])
{
    FILE *fp;
    char c;
    fp = fopen("arquivo.txt", "r");
    if(!fp)
    {
        printf( "Erro na abertura do
arquivo");
        exit(-1);
    }
    c = getc(fp);
    while( c != EOF )
    {
        printf("%c", c);
        c = getc(fp);
    }
    fclose(fp);
    return 0;
}
```

Vamos ler o  
primeiro caractere  
do arquivo




```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, const char * argv[])
{
    FILE *fp;
    char c;
    fp = fopen("arquivo.txt", "r");
    if(!fp)
    {
        printf( "Erro na abertura do
arquivo");
        exit(-1);
    }
    c = getc(fp);
    while( c != EOF )
    {
        printf("%c", c);
        c = getc(fp);
    }
    fclose(fp);
    return 0;
}
```

EOF = *end of file*



```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, const char * argv[])
{
    FILE *fp;
    char c;
    fp = fopen("arquivo.txt", "r");
    if(!fp)
    {
        printf( "Erro na abertura do
arquivo");
        exit(-1);
    }
    c = getc(fp);
    while( c != EOF )
    {
        printf("%c", c);
        c = getc(fp);
    }
    fclose(fp);
    return 0;
}
```

Enquanto não  
chegamos ao final  
do arquivo, cada  
caractere é lido e  
escrito na tela



```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
FILE * abre_arq(char* arquivo, char *modo);
void le_arq(FILE *p);
int main(int argc, const char * argv[])
{
    FILE *p;
    char str[100], c;
    char frase[120] = "Este e um arquivo chamado: ";
    printf("Entre com um nome para o arquivo:\n");
    gets(str);
    p = abre_arq(str, "w");
    strcat(frase, str);
    fputs(frase, p);
    putc('\n', p);
    fclose(p);
    p = abre_arq(str, "r");
    le_arq(p);
    fclose(p);
    return 0;
}

```

Code/04\_File\_stdio/Ex5.c

```

FILE * abre_arq(char* arquivo,
char *modo)
{
    FILE *p = fopen(arquivo,
modo);
    if(p==NULL)
    {
        printf("Erro!
Impossivel abrir o
arquivo!\n");
        exit(-1);
    }
    return p;
}
void le_arq(FILE *p)
{
    char c = getc(p);
    while( c != EOF )
    {
        printf("%c", c);
        c = getc(p);
    }
}

```

(continuação)

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

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

```
FILE * abre_arq(char* arquivo,
```

```
void le_arq(FILE *p)
```

```
int main()
```

```
{
```

```
FILE *p;
```

```
char *arquivo;
```

```
char *modo;
```

```
printf("Digite o nome do arquivo: ");
```

```
gets(arquivo);
```

```
p = abre_arq(arquivo, "r");
```

```
if (p == NULL)
```

```
strcat(arquivo, ".txt");
```

```
printf("Arquivo não encontrado. Criando...\n");
```

```
fputs("Conteúdo do arquivo", p);
```

```
putc(' ', p);
```

```
fclose(p);
```

```
p = abre_arq(arquivo, "r");
```

```
le_arq(p);
```

```
fclose(p);
```

```
return 0;
```

```
}
```

Foram criadas duas funções para facilitar o entendimento do código.

“`abre_arq( )`” abre o arquivo indicado, e termina a execução do código em caso de erro

“`le_arq( )`” lê um arquivo já aberto e escreve na tela seu conteúdo

do):

chamado: "  
arquivo:\n");

```
FILE * abre_arq(char* arquivo,
char *modo)
{
```

```
FILE *p = fopen(arquivo,
modo);
```

```
if(p==NULL)
{
```

```
printf("Erro!
Impossivel abrir o
arquivo!\n");
exit(-1);
```

```
}
```

```
return p;
```

```
}
```

```
void le_arq(FILE *p)
{
```

```
char c = getc(p);
while( c != EOF )
{
```

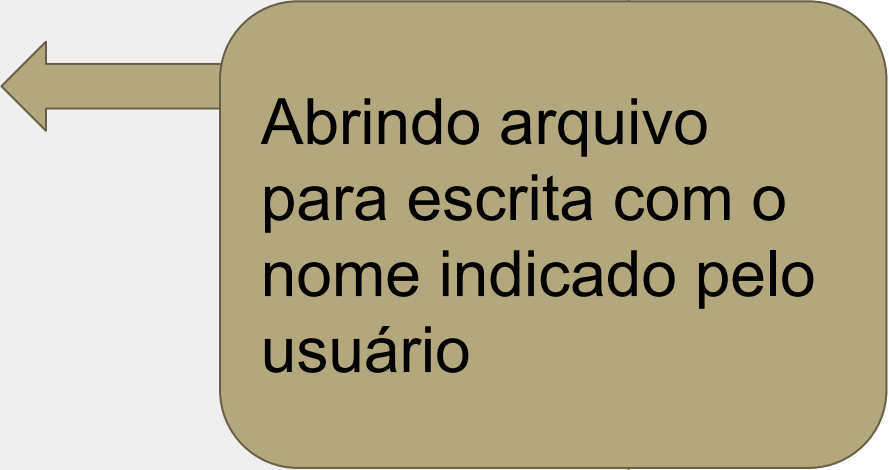
```
printf("%c", c);
c = getc(p);
```

```
}
```

```
}
```



```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
FILE * abre_arq(char* arquivo, char *modo);
void le_arq(FILE *p);
int main(int argc, const char * argv[])
{
    FILE *p;
    char str[100], c;
    char frase[120] = "Este e um arquivo chamado: ";
    printf("Entre com um nome para o arquivo:\n");
    gets(str);
    p = abre_arq(str, "w");
    strcat(frase, str);
    fputs(frase, p);
    putc('\n', p);
    fclose(p);
    p = abre_arq(str, "r");
    le_arq(p);
    fclose(p);
    return 0;
}
```



Abrindo arquivo  
para escrita com o  
nome indicado pelo  
usuário

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
FILE * abre_arq(char* arquivo, char *modo);
void le_arq(FILE *p);
int main(int argc, const char * argv[])
{
    FILE *p;
    char str[100], c;
    char frase[120] = "Este e um arquivo chamado: ";
    printf("Entre com um nome para o arquivo:\n");
    gets(str);
    p = abre_arq(str, "w");
    strcat(frase, str);
    fputs(frase, p);
    putc('\n', p);
    fclose(p);
    p = abre_arq(str, "r");
    le_arq(p);
    fclose(p);
    return 0;
}
```

Concatenando o conteúdo de “frase” com o conteúdo em “str”

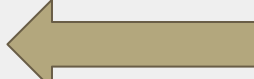
A função “strcat( )” é definida na biblioteca <string.h>

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
FILE * abre_arq(char* arquivo, char *modo);
void le_arq(FILE *p);
int main(int argc, const char * argv[])
{
    FILE *p;
    char str[100], c;
    char frase[120] = "Este e um arquivo chamado: ";
    printf("Entre com um nome para o arquivo:\n");
    gets(str);
    p = abre_arq(str, "w");
    strcat(frase, str);
    fputs(frase, p);
    putc('\n', p);
    fclose(p);
    p = abre_arq(str, "r");
    le_arq(p);
    fclose(p);
    return 0;
}
```

Escrevendo o conteúdo  
de “frase” no arquivo  
aberto

A função “fputs( )” sabe a  
hora de parar a escrita  
detectando o ‘\0’ no final  
da string

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
FILE * abre_arq(char* arquivo, char *modo);
void le_arq(FILE *p);
int main(int argc, const char * argv[])
{
    FILE *p;
    char str[100], c;
    char frase[120] = "Este e um arquivo chamado: ";
    printf("Entre com um nome para o arquivo:\n");
    gets(str);
    p = abre_arq(str, "w");
    strcat(frase, str);
    fputs(frase, p);
    putc('\n', p);
    fclose(p);
    p = abre_arq(str, "r");
    le_arq(p);
    fclose(p);
    return 0;
}
```



Vamos abrir o arquivo  
escrito e ler seu conteúdo

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
FILE * abre_arq(char* arquivo, char *modo);

int main()
{
    FILE *fp;
    char string[100];
    fp = abre_arq("arquivo.txt", "w");
    do
    {
        printf("\nDigite uma nova string."
            " Para terminar, digite <enter>: ");
        gets(string);
        fputs(string, fp);
        putc('\n', fp);
    } while(strlen(string) > 0);
    fclose(fp);
}

```

Code/04\_File\_stdio/Ex6.c

```

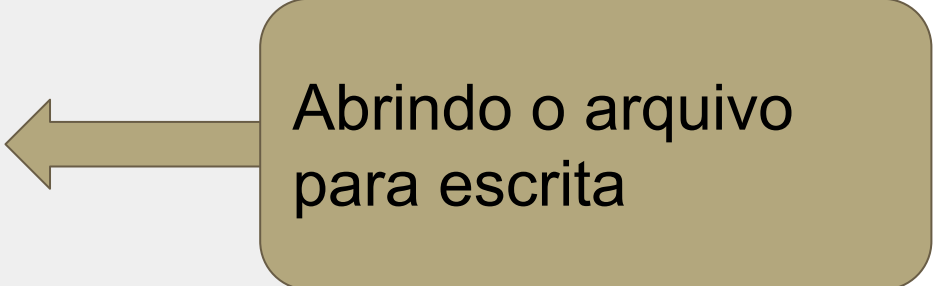
FILE * abre_arq(char* arquivo,
char *modo)
{
    FILE *p = fopen(arquivo,
modo);
    if(p==NULL)
    {
        printf("Erro!
Impossível abrir o
arquivo!\n");
        exit(-1);
    }
    return p;
}

```

(continuação)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
FILE * abre_arq(char* arquivo, char *modo);
```

```
int main()
{
    FILE *fp;
    char string[100];
    fp = abre_arq("arquivo.txt", "w");
    do
    {
        printf("\nDigite uma nova string."
            " Para terminar, digite <enter>: ");
        gets(string);
        fputs(string, fp);
        putc('\n', fp);
    } while(strlen(string) > 0);
    fclose(fp);
}
```

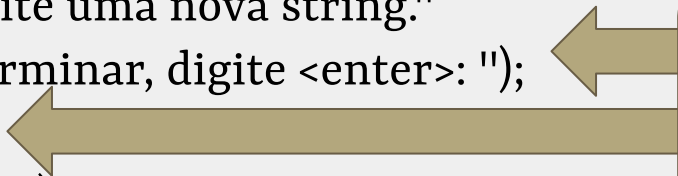


Abrindo o arquivo  
para escrita

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
FILE * abre_arq(char* arquivo, char *modo);
```

```
int main()
{
    FILE *fp;
    char string[100];
    fp = abre_arq("arquivo.txt", "w");
    do
    {
        printf("\nDigite uma nova string."
            " Para terminar, digite <enter>: ");
        gets(string);
        fputs(string, fp);
        putc('\n', fp);
    } while(strlen(string) > 0);
    fclose(fp);
}
```


Obtendo texto do usuário

A diagram consisting of two horizontal arrows pointing to the left. The top arrow originates from the text box and points to the first parameter of the printf statement. The bottom arrow originates from the text box and points to the gets(string) function call.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
FILE * abre_arq(char* arquivo, char *modo);

int main()
{
    FILE *fp;
    char string[100];
    fp = abre_arq("arquivo.txt", "w");
    do
    {
        printf("\nDigite uma nova string."
            " Para terminar, digite <enter>: ");
        gets(string);
        fputs(string, fp);
        putc('\n', fp);
    } while(strlen(string) > 0);
    fclose(fp);
}
```

Escrevendo o texto  
obtido no arquivo  
aberto





```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
FILE * abre_arq(char* arquivo, char *modo);
```

```
int main()
{
    FILE *fp;
    char string[100];
    fp = abre_arq("arquivo.txt", "w");
    do
    {
        printf("\nDigite uma nova string."
               " Para terminar, digite <enter>: ");
        gets(string);
        fputs(string, fp);
        putc('\n', fp);
    } while(strlen(string) > 0);
    fclose(fp);
}
```

“strlen( )” obtém o tamanho da string digitada pelo usuário

Se o tamanho for 0, o usuário só apertou o ENTER. Nesse caso, o código para de escrever no arquivo


```
#include <stdio.h>
#include <stdlib.h>
FILE * abre_arq(char* arquivo, char *modo);

int main()
{
    FILE *fp;
    float pi = 3.1415;
    float pilido;
    fp = abre_arq("arquivo.bin", "wb");
    fwrite(&pi, sizeof(float), 1, fp);
    fclose(fp);
    fp = abre_arq("arquivo.bin", "rb");
    fread(&pilido, sizeof(float), 1, fp);
    printf("O valor de PI "
           "(lido do arquivo) eh: %f\n",
           pilido);
    fclose(fp);
    return(0);
}
```

```
#include <stdio.h>
#include <stdlib.h>
FILE * abre_arq(char* arquivo, char *modo);

int main()
{
    FILE *fp;
    float pi = 3.1415;
    float pilido;
    fp = abre_arq("arquivo.bin", "wb");
    fwrite(&pi, sizeof(float), 1, fp);
    fclose(fp);
    fp = abre_arq("arquivo.bin", "rb");
    fread(&pilido, sizeof(float), 1, fp);
    printf("O valor de PI "
           "(lido do arquivo) eh: %f\n",
           pilido);
    fclose(fp);
    return(0);
}
```

Escrevendo em modo binário o valor da variável “pi” em “arquivo.bin”



```
#include <stdio.h>
#include <stdlib.h>
FILE * abre_arq(char* arquivo, char *modo);

int main()
{
    FILE *fp;
    float pi = 3.1415;
    float pilido;
    fp = abre_arq("arquivo.bin", "wb");
    fwrite(&pi, sizeof(float), 1, fp);
```

```
size_t fwrite ( const void * ptr, size_t size, size_t count, FILE * stream );
```

“fwrite( )” escreve em arquivo o conteúdo de uma variável ou de um vetor

“ptr” indica o endereço da variável a ser escrita, ou o começo do vetor

“size” indica o tamanho em bytes da variável

“count” indica a quantidade de posições no vetor a ser escrita


“stream” é o ponteiro para o arquivo aberto

A função “fwrite( )” retorna a quantidade de elementos que puderam ser escritos no arquivo

```
#include <stdio.h>
#include <stdlib.h>
FILE * abre_arq(char* arquivo, char *modo);

int main()
{
    FILE *fp;
    float pi = 3.1415;
    float pilido;
    fp = abre_arq("arquivo.bin", "wb");
    fwrite(&pi, sizeof(float), 1, fp);
    fclose(fp);
    fp = abre_arq("arquivo.bin", "rb");
    fread(&pilido, sizeof(float), 1, fp);
    printf("O valor de PI "
           "(lido do arquivo) eh: %f\n",
           pilido);
    fclose(fp);
    return(0);
}
```

Abrindo o arquivo em modo binário, e lendo o valor salvo no arquivo na variável “pilido”



```
#include <stdio.h>
#include <stdlib.h>
FILE * abre_arq(char* arquivo, char* modo);
```

```
size_t fread ( const void * ptr, size_t size, size_t count, FILE * stream );
```

“fread( )” escreve em uma variável ou vetor o conteúdo de um arquivo

“ptr” indica o endereço da variável aonde escrever, ou o começo do vetor

“size” indica o tamanho em bytes da variável

“count” indica a quantidade de posições no vetor a ser escrita

“stream” é o ponteiro para o arquivo aberto

A função “fread( )” retorna a quantidade de elementos que puderam ser lidos do arquivo

```
fread(&pilido, sizeof(float), 1, fp);
printf("O valor de PI "
      "(lido do arquivo) eh: %f\n",
      pilido);
fclose(fp);
return(0);
}
```

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

```
#  
~/Code/04_File_stdio $ ./Ex7.out  
O valor de PI (lido do arquivo) eh: 3.141500  
~/Code/04_File_stdio $ ls -l arquivo.bin  
-rw-r--r-- 1 diogo diogo 4 mar 20 15:01 arquivo.bin  
{  
~/Code/04_File_stdio $ cat arquivo.bin  
VI@
```

```
float pi = 3.1415;  
float pilido;  
fp = abre_arq("arquivo.bin", "wb");  
fwrite(&pi, sizeof(float), 1, fp);  
fclose(fp);  
fp = abre_arq("arquivo.bin", "rb");  
fread(&pilido, sizeof(float), 1, fp);  
printf("O valor de PI "  
      "(lido do arquivo) eh: %f\n",  
      pilido);  
fclose(fp);  
return(0);  
}
```

Code/04\_File\_stdio/Ex7.c

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

```
# ./Code/04_File_stdio $ ./Ex7.out  
O valor de PI (lido do arquivo) eh: 3.141500  
~/Code/04_File_stdio $ ls -l arquivo.bin  
-rw-r--r-- 1 diogo diogo 4 mar 20 15:01 arquivo.bin  
~/Code/04_File_stdio $ cat arquivo.bin  
VI@
```

```
float pi = 3.1415;
```

3.1415 não pode ser representado perfeitamente em “float”,  
sendo aproximado por 3.141499996185302734375

Em binário, ele é representado por

0 10000000 10010010000111001010110  
Sinal Expoente Mantissa

Em hexadecimal, ele é representado por  
0x40 0x49 0x0e 0x56

Em ASCII, esses bytes são interpretados como  
[@] [I] [SHIFT OUT] [V],  
que aparecem no terminal como “VI@”

<https://www.exploringbinary.com/floating-point-converter/>  
<https://pt.wikipedia.org/wiki/ASCII>



```
#include <stdio.h>
#include <stdlib.h>
FILE * abre_arq(char* arquivo, char *modo);
int main()
{
    FILE *fp;
    float pi = 3.1415;
    float pilido;

    fp = abre_arq("arquivo.txt", "w");
    fprintf(fp, "pi = %f\n", pi);
    fclose(fp);

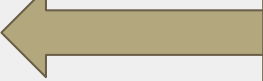
    fp = abre_arq("arquivo.txt", "r");
    fscanf(fp, "pi = %f\n", &pilido);
    printf("O valor de PI "
           "(lido do arquivo) eh: %f\n",
           pilido);
    fclose(fp);
    return(0);
}
```

```
#include <stdio.h>
#include <stdlib.h>
FILE * abre_arq(char* arquivo, char *modo);
int main()
{
    FILE *fp;
    float pi = 3.1415;
    float pilido;

    fp = abre_arq("arquivo.txt", "w");
    fprintf(fp, "pi = %f\n", pi);
    fclose(fp);

    fp = abre_arq("arquivo.txt", "r");
    fscanf(fp, "pi = %f\n", &pilido);
    printf("O valor de PI "
           "(lido do arquivo) eh: %f\n",
           pilido);
    fclose(fp);
    return(0);
}
```

Escrevendo em modo texto o valor da variável “pi” em “arquivo.txt”

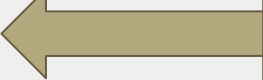


```
#include <stdio.h>
#include <stdlib.h>
FILE * abre_arq(char* arquivo, char *modo);
int main()
{
    FILE *fp;
    float pi = 3.1415;
    float pilido;

    fp = abre_arq("arquivo.txt", "w");
    fprintf(fp, "pi = %f\n", pi);
    fclose(fp);

    fp = abre_arq("arquivo.txt", "r");
    fscanf(fp, "pi = %f\n", &pilido);
    printf("O valor de PI "
           "(lido do arquivo) eh: %f\n",
           pilido);
    fclose(fp);
    return(0);
}
```

Lendo em modo texto  
o valor da variável “pi”  
em “arquivo.txt”



```

#include <stdio.h>
#include <stdlib.h>
FILE * abre_arq(char* arquivo, char *modo);
int main()
{
    FILE *fp;
    float pi = 3.1415;

    printf("O valor de PI\n");
    printf("O valor de PI\n");
    printf("(lido do arquivo) eh: %f\n",
        pilido);
    fclose(fp);
    return(0);
}

```

```

~/Code/04_File_stdio $ ./Ex8.out
O valor de PI (lido do arquivo) eh: 3.141500
~/Code/04_File_stdio $ ls -l arquivo.bin arquivo.txt
-rw-r--r-- 1 diogo diogo  4 mar 20 15:01 arquivo.bin
-rw-r--r-- 1 diogo diogo 14 mar 20 15:45 arquivo.txt
~/Code/04_File_stdio $ cat arquivo.bin
VI@
~/Code/04_File_stdio $ cat arquivo.txt
pi = 3.141500

```

Code/04\_File\_stdio/Ex8.c

# Confirmam também

- Reposicionamento do ponteiro para o arquivo:  
`int fseek (FILE *fp, long int offset, int origin);`  
    `origin = 0 = SEEK_SET`: início do arquivo  
    `origin = 1 = SEEK_CUR`: posição atual do arquivo  
    `origin = 2 = SEEK_END`: fim do arquivo
- Reposicionamento do ponteiro para o início do arquivo:  
`void rewind ( FILE *fp);`
- Apagar um arquivo:  
`int remove(char *nome_do_arquivo);`