

Sistemas Embarcados

Acesso a Arquivos com POSIX

POSIX

- **Portable Operating System Interface**

- Norma da *IEEE Computer Society* para compatibilidade entre sistemas operacionais
- Baseada em Unix
- Iniciada em 1988
- Última versão é de 2017
- Padroniza uma série de ferramentas de sistemas operacionais, tais como a linguagem de script shell, interfaces de sistema, operações sobre arquivos etc.

stdio.h “vs” POSIX

Operações sobre arquivos	stdio.h	POSIX
Abrir	<code>fopen()</code>	<code>open()</code>
Fechar	<code>fclose()</code>	<code>close()</code>
Ler	<code>fread()</code> <code>fscanf()</code> <code>getc()</code>	<code>read()</code>
Escrever	<code>fwrite()</code> <code>fprintf()</code> <code>putc()</code>	<code>write()</code>
Outros	<code>fseek()</code> <code>rewind()</code>	<code>mmap()</code> <code>lseek()</code> <code>poll()</code>
Vantagens	<i>Bufferizado</i> Poderoso Sempre disponível	Simples Rápido Baixo nível Preferível para interfaces com o hardware
Desvantagens	<i>Buffering</i> pode ser confuso, pois adiciona <i>overheads</i>	Somente para UNIX

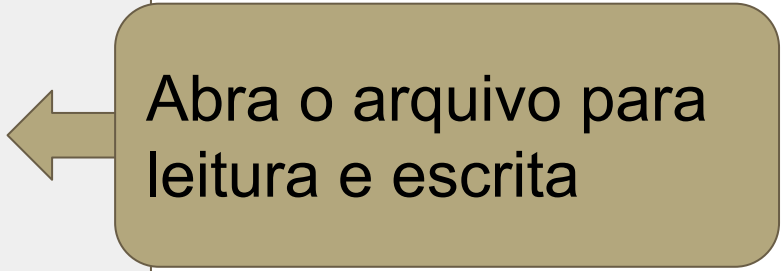
```
#include <stdio.h> // Para a funcao printf()
#include <stdlib.h> // Para a função exit()
#include <fcntl.h> // Para a funcao open()
#include <unistd.h> // Para a funcao close()

int main(int argc, const char * argv[])
{
    int fd = open("exemplo.bin", O_RDWR);
    if(fd==-1)
    {
        printf("Erro na abertura"
               " do arquivo.\n");
        exit(-1);
    }
    close(fd);
    return 0;
}
```

Code/05_File_POSIX/Ex1.c

```
#include <stdio.h> // Para a funcao printf()
#include <stdlib.h> // Para a função exit()
#include <fcntl.h> // Para a funcao open()
#include <unistd.h> // Para a funcao close()
```

```
int main(int argc, const char * argv[])
{
    int fd = open("exemplo.bin", O_RDWR);
    if(fd==-1)
    {
        printf("Erro na abertura"
               " do arquivo.\n");
        exit(-1);
    }
    close(fd);
    return 0;
}
```



Abra o arquivo para
leitura e escrita

Code/05_File_POSIX/Ex1.c

```
#include <stdio.h> // Para a funcao printf()
```

```
#in
```

```
#in
```

```
#in
```

```
int
```

```
{
```

```
}
```

Modo	Significado
O_RDONLY	Abre o arquivo somente leitura
O_WRONLY	Abre o arquivo somente para escrita
O_RDWR	Abre o arquivo para leitura e escrita
O_APPEND	Abre o arquivo para escrita ao final do arquivo
O_CREAT	Cria o arquivo se ele não existe
O_NONBLOCK O_EXCL O_SHLOCK O_EXLOCK O_NOFOLLOW O_SYMLINK O_EVTONLY O_CLOEXEC	Confira em http://man7.org/linux/man-pages/man2/open.2.html

para

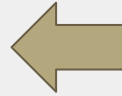
a

Code/05_File_POSIX/Ex1.c

```
#include <stdio.h> // Para a funcao printf()
#include <stdlib.h> // Para a função exit()
#include <fcntl.h> // Para a funcao open()
#include <unistd.h> // Para a funcao close()
```

```
int main(int argc, const char * argv[])
{
    int fd = open("exemplo.bin", O_RDWR);
    if(fd==-1)
    {
        printf("Erro na abertura"
               " do arquivo.\n");
        exit(-1);
    }
    close(fd);
    return 0;
}
```

O valor inteiro
retornado é
chamado de
descritor de arquivo



Code/05_File_POSIX/Ex1.c

```
#include
```

```
#include
```

```
#include
```

```
#include
```

```
int
```

```
{
```

```
}
```

Descritor de arquivo é um número inteiro que representa um arquivo aberto

Cada programa em execução possui um número único de identificação (PID) e uma tabela de descritores de arquivo, disponíveis em “/proc/PID/fd”

Alguns descritores de arquivo padronizados são

- 0: /dev/stdin
- 1: /dev/stdout
- 2: /dev/stderr

Quase tudo no UNIX é arquivo

Code/00_1110_10011/21110

ivo


```
#include <stdio.h> // Para a funcao printf()
#include <stdlib.h> // Para a função exit()
#include <fcntl.h> // Para a funcao open()
#include <unistd.h> // Para a funcao close()
```

```
int main(int argc, const char * argv[])
{
    int fd = open("exemplo.bin", O_RDONLY);
    if(fd == -1)
    {
        printf("Erro na abertura"
               " do arquivo.\n");
        exit(-1);
    }
    close(fd);
    return 0;
}
```

Se for retornado o valor “-1”, houve um erro na abertura do arquivo

```
#include <stdio.h> // Para a funcao printf()
#include <stdlib.h> // Para a função exit()
#include <fcntl.h> // Para a funcao open()
#include <unistd.h> // Para a funcao close()
```

```
int main(int argc, const char * argv[])
{
    int fd = open("exemplo.bin", O_RDWR);
    if(fd==-1)
    {
        printf("Erro na abertura"
               " do arquivo.\n");
        exit(-1);
    }
    close(fd);
    return 0;
}
```



Fecha o arquivo

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

int main(int argc, const char * argv[])
{
    int fd = open("exercicio2.txt",
        O_RDWR | O_CREAT);
    if(fd== -1)
    {
        printf("Erro na abertura"
            " do arquivo.\n");
        exit(-1);
    }
    printf("Abertura OK\n");
    close(fd);
    return 0;
}
```

Code/05_File_POSIX/Ex2.c

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

```
int main(int argc, const char * argv[])
{
    int fd = open("exercicio2.txt",
        O_RDWR | O_CREAT);
    if(fd == -1)
    {
        printf("Erro na abertura"
            " do arquivo.\n");
        exit(-1);
    }
    printf("Abertura OK\n");
    close(fd);
    return 0;
}
```

Definidos em <fcntl.h>, O_RDWR e O_CREAT não são excludentes. Aqui, queremos abrir um arquivo para leitura e escrita, criando-o se ele não existe

Code/05_File_POSIX/Ex2.c

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

int main(int argc, const char * argv[])
{
    int fd = open("exercicio3.txt",
        O_RDWR | O_CREAT);
    if(fd== -1)
    {
        printf("Erro na abertura"
            " do arquivo.\n");
        exit(-1);
    }
    printf("Arquivo 'exercicio3.txt'"
        " criado. Detalhes:\n");
    system("ls -l exercicio3.txt");
    close(fd);
    return 0;
}
```

Code/05_File_POSIX/Ex3.c

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

int main(int argc, const char * argv[])
{
    int fd = open("exercicio3.txt",
        O_RDWR | O_CREAT);
    if(fd== -1)
    {
        printf("Erro na abertura"
            " do arquivo.\n");
        exit(-1);
    }
    printf("Arquivo 'exercicio3.txt'"
        " criado. Detalhes:\n");
    system("ls -l exercicio3.txt");
    close(fd);
    return 0;
}
```

A função “system()”
executa o comando
indicado por sua string
de entrada

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

int main(int argc, const char * argv[])
{
    int fd = open("exercicio3.txt",
        O_RDWR | O_CREAT);
    if(fd== -1)
    {
        printf("Erro na abertura"
            " do arquivo.\n");
        exit(-1);
    }
    printf("Arquivo 'exercicio3.txt'"
        " criado. Detalhes:\n");
    system("ls -l exercicio3.txt");
    close(fd);
    return 0;
}
```

Ela não faz parte do POSIX, mas foi apresentada aqui por ser bastante importante ao longo do curso

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


int main(int argc, const char * argv[])
{
    int fd = open("exercicio4.txt",
        O_RDWR | O_CREAT, S_IRWXU);
    if(fd==-1)
    {
        printf("Erro na abertura"
            " do arquivo.\n");
        exit(-1);
    }
    system("ls -l exercicio3.txt "
        "exercicio4.txt");
    close(fd);
    return 0;
}
```



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

```
int main(int argc, const char * argv[])
{
    int fd = open("exercicio4.txt",
                  O_RDWR | O_CREAT, S_IRWXU);
    if(fd==-1)
    {
        printf("Erro na abertura"
              " do arquivo.\n");
        exit(-1);
    }
    system("ls -l exercicio3.txt "
          "exercicio4.txt");
    close(fd);
    return 0;
}
```

A função “open()”
aceita um terceiro
argumento
indicando as
permissões do
arquivo a ser
criado



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

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

Modo	Significado
S_IRWXU	00700 - Usuário pode ler, escrever e executar
S_IRUSR	00400 - Usuário pode ler
S_IWUSR	00200 - Usuário pode escrever
S_IXUSR	00100 - Usuário pode executar
S_IRWXG	00070 - Outros do mesmo grupo podem ler, escrever e executar
S_IRGRP	00040 - Outros do mesmo grupo podem ler
S_IWGRP	00020 - Outros do mesmo grupo podem escrever
S_IXGRP	00010 - Outros do mesmo grupo podem executar
S_IRWXO	00007 - Outros podem ler, escrever e executar
S_IROTH	00004 - Outros podem ler
S_IWOTH	00002 - Outros podem escrever
S_IXOTH	00001 - Outros podem executar

Code/05_File_POSIX/Ex4.c

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

int main(int argc, const char * argv[])
{
    int fd = open("exercicio4.txt",
        O_RDWR | O_CREAT, S_IRWXU);
    if(fd== -1)
    {
        printf("Erro na abertura"
            " do arquivo.\n");
        exit(-1);
    }
    system("ls -l exercicio3.txt "
        "exercicio4.txt");
    close(fd);
    return 0;
}
```

Confira a diferença
entre os resultados
deste exemplo e
do exemplo
anterior

```

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
int main(int argc, const char * argv[])
{
    char string[100];
    int i, fd = open ("exercicio5.txt", O_WRONLY | O_CREAT, S_IRWXU);
    if(fd==-1)
    {
        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++)
        write(fd, &(string[i]), 1);
    write(fd, "\n", 1);
    close(fd);
    return 0;
}

```

```

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
int main(int argc, const char * argv[])
{
    char string[100];
    int i, fd = open ("exercicio5.txt", O_WRONLY | O_CREAT, S_IRWXU);
    if(fd==-1)
    {
        printf("Erro na abertura"
               " do arquivo.\n");
        exit(-1);
    }
    printf("Entre com a string a ser gravada");
    gets(string);
    for(i=0; string[i] != 0; i++)
        write(fd, &(string[i]), 1);
    write(fd, "\n", 1);
    close(fd);
    return 0;
}

```

Depois de abrir o arquivo “exercicio5.txt”, o programa pede uma string para ser inserida no arquivo

(Ignore o perigo no uso da função “gets()” neste exemplo...)

```

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
int main(int argc, const char * argv[])
{
    char string[100];
    int i, fd = open ("exercicio5.txt", O_WRONLY | O_CREAT, S_IRWXU);
    if(fd==-1)
    {
        printf("Erro na abertura"
               " do arquivo.\n");
        exit(-1);
    }
    printf("Entre com a string a ser gravada: ");
    gets(string);
    for(i=0; string[i] != 0; i++)
        write(fd, &(string[i]), 1);
    write(fd, "\n", 1);
    close(fd);
    return 0;
}

```


Enquanto não se encontra o fim da string (caractere '\0')...

```

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
int main(int argc, const char * argv[])
{
    char string[100];
    int i, fd = open ("exercicio5.txt", O_WRONLY | O_CREAT, S_IRWXU);
    if(fd==-1)
    {
        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++)
        write(fd, &(string[i]), 1);
    write(fd, "\n", 1);
    close(fd);
    return 0;
}

```

... o programa escreve o
caractere correspondente
no arquivo



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

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

```
{
```

```
    size_t write(int fd, const void *buf, size_t count);
```

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

- “fd” é o descritor de arquivo do arquivo aberto
- “buf” indica o endereço da variável a ser escrita, ou o começo do vetor
- “count” indica a quantidade de bytes no vetor a ser escrita
- “write()” retorna a quantidade de bytes que puderam ser escritos no arquivo

```
    write(fd, "\n", 1);
    close(fd);
    return 0;
```

```
}
```

RWXU);

escreve o
respondente

no arquivo

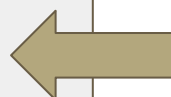

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>

int main(int argc, const char * argv[])
{
    char c;
    int fd = open("exercicio5.txt", O_RDONLY);
    if(fd== -1)
    {
        printf("Erro na abertura"
               " do arquivo.\n");
        exit(1);
    }
    while(read(fd, &c, 1) != 0)
        printf("%c", c);
    close(fd);
    return 0;
}
```

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

```
int main(int argc, const char * argv[])
{
    char c;
    int fd = open("exercicio5.txt", O_RDONLY);
    if(fd == -1)
    {
        printf("Erro na abertura"
               " do arquivo.\n");
        exit(1);
    }
    while(read(fd, &c, 1) != 0)
        printf("%c", c);
    close(fd);
    return 0;
}
```


Abrindo o
arquivo do
exemplo anterior
para leitura



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

```
int main(int argc, const char * argv[])
{
    char c;
    int fd = open("exercicio5.txt", O_RDONLY);
    if(fd==-1)
    {
        printf("Erro na abertura"
               " do arquivo.\n");
        exit(1);
    }
    while(read(fd, &c, 1) != 0)
        printf("%c", c);
    close(fd);
    return 0;
}
```

Enquanto o programa
consegue ler um byte do
arquivo aberto...



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

```
int main(int argc, const char * argv[])
{
    char c;
    int fd = open("exercicio5.txt", O_RDONLY);
    if(fd== -1)
    {
        printf("Erro na abertura"
               " do arquivo.\n");
        exit(1);
    }
    while(read(fd, &c, 1) != 0)
        printf("%c", c);
    close(fd);
    return 0;
}
```

... ele escreve o byte lido
na tela.



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

```
int
{
    size_t read(int fd, const void *buf, size_t count);
```

“read()” lê o conteúdo de um arquivo e o coloca em uma variável ou um vetor

- “fd” é o descritor de arquivo do arquivo aberto
- “buf” indica o endereço da variável aonde escrever, ou o começo do vetor
- “count” indica a quantidade de bytes no vetor a ser lida
- “read()” retorna a quantidade de bytes que puderam ser lidos do arquivo

```
close(fd),
return 0;
}
```

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

int main(int argc, const char * argv[])
{
    float pi = 3.1415;
    float pilido;
    int fd = open ("exercicio7.txt", O_RDWR | O_CREAT, S_IRWXU);
    if(fd==-1){ printf("Erro na abertura do arquivo.\n"); exit(-1); }
    write(fd, &pi, sizeof(float));
    close(fd);

    fd = open("exercicio7.txt", O_RDONLY);
    read(fd, &pilido, sizeof(float));
    close(fd);
    printf("\nO valor de PI (lido do arquivo) eh: %f\n\n", pilido);
    return(0);
}
```

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

```
int main(int argc, const char * argv[])
{
    float pi = 3.1415;
    float pilido;
    int fd = open ("exercicio7.txt", O_RDWR | O_CREAT, 0666);
    if(fd==-1){ printf("Erro na abertura do arquivo\n");
    write(fd, &pi, sizeof(float));
    close(fd);

    fd = open("exercicio7.txt", O_RDONLY);
    read(fd, &pilido, sizeof(float));
    close(fd);
    printf("\nO valor de PI (lido do arquivo) eh: %f\n", pilido);
    return(0);
}
```

Aqui, o programa está escrevendo o conteúdo da variável “pi” no arquivo aberto

Como “pi” é do tipo “float”, serão escritos 4 bytes no arquivo

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

```
int main(int argc, const char * argv[])
{
    float pi = 3.1415;
    float pilido;
    int fd = open ("exercicio7.txt", O_RDWR | O_CREAT, 0644);
    if(fd==-1){ printf("Erro na abertura do arquivo\n");
    write(fd, &pi, sizeof(float));
    close(fd);

    fd = open("exercicio7.txt", O_RDONLY);
    read(fd, &pilido, sizeof(float));
    close(fd);
    printf("\nO valor de PI (lido do arquivo) eh: %f\n", pilido);
    return(0);
}
```

Depois de fechar o arquivo para escrita, o programa abre o arquivo para leitura, lê 4 bytes do arquivo e os coloca na variável “pilido”


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

int main(int argc, const char * argv[])
{
    char str[100] = {'\0'};
    float pi = 3.1415;
    float pilido;
    int fd = open ("exercicio8.txt", O_RDWR | O_CREAT, S_IRWXU);
    sprintf(str, "pi = %f\n", pi);
    write(fd, str, 100);
    close(fd);

    fd = open("exercicio8.txt", O_RDONLY);
    read(fd, str, 100);
    sscanf(str, "pi = %f\n", &pilido);
    close(fd);
    printf("\nO valor de PI (lido do arquivo) eh: %f\n\n", pilido);
    return(0);
}
```

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

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

```
    char str[100] = {'\0'};
```

```
    float pi = 3.1415;
```

```
    float pilido;
```

```
    int fd = open ("exercicio8.txt", O_RDWR | O_CREAT, S_IRWXU);
```

```
    sprintf(str, "pi = %f\n", pi);
```

```
    write(fd, str, 100);
```

```
    close(fd);
```

```
    fd = open("exercicio8.txt", O_RDONLY
```

```
    read(fd, str, 100);
```


```
    sscanf(str, "pi = %f\n", &pilido);
```

```
    close(fd);
```

```
    printf("\nO valor de PI (lido do arquivo) eh: %f\n\n", pilido);
```

```
    return(0);
```

```
}
```

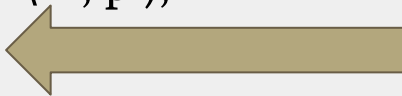


Agora, o programa está escrevendo o texto “pi = 3.141500\n” na string “str”...

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

```
int main(int argc, const char * argv[])
{
    char str[100] = {'\0'};
    float pi = 3.1415;
    float pilido;
    int fd = open ("exercicio8.txt", O_RDWR | O_CREAT, S_IRWXU);
    sprintf(str, "pi = %f\n", pi);
    write(fd, str, 100);
    close(fd);

    fd = open("exercicio8.txt", O_RDONLY);
    read(fd, str, 100);
    sscanf(str, "pi = %f\n", &pilido);
    close(fd);
    printf("\nO valor de PI (lido do arquivo) eh: %f\n\n", pilido);
    return(0);
}
```



... e escrevendo a string “str” INTEIRA (100 bytes) no arquivo aberto.

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

```
int main(int argc, const char * argv[])
{
    char str[100] = {'\0'};
    float pi = 3.1415;
    float pilido;
    int fd = open ("exercicio8.txt", O_RDWR | O_CREAT, S_IRWXU);
    sprintf(str, "pi = %f\n", pi);
    write(fd, str, 100);
    close(fd);

    fd = open("exercicio8.txt", O_RDONLY);
    read(fd, str, 100);
    sscanf(str, "pi = %f\n", &pilido);
    close(fd);
    printf("\nO valor de PI (lido do arquivo) e\n");
    return(0);
}
```

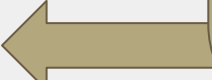
Depois, o programa fecha o arquivo para escrita, o abre para leitura, e lê 100 bytes do arquivo

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

```
int main(int argc, const char * argv[])
{
    char str[100] = {'\0'};
    float pi = 3.1415;
    float pilido;
    int fd = open ("exercicio8.txt", O_RDWR | O_CREAT, 0666);
    sprintf(str, "pi = %f\n", pi);
    write(fd, str, 100);
    close(fd);

    fd = open("exercicio8.txt", O_RDONLY);
    read(fd, str, 100);
    sscanf(str, "pi = %f\n", &pilido);
    close(fd);
    printf("\nO valor de PI (lido do arquivo) eh: %f\n\n", pilido);
    return(0);
}
```

A função “sscanf()” procura pelo padrão “pi = “%f\n” na string “str”, e guarda o resultado na variável “pilido”



Confirmam também

- Reposicionamento do ponteiro para o arquivo:
`int lseek (int fd, long int offset, int whence);`
 `whence = 0 = SEEK_SET`: início do arquivo
 `whence = 1 = SEEK_CUR`: posição atual do arquivo
 `whence = 2 = SEEK_END`: fim do arquivo