POSIX

Portable **O**perating **S**ystem **I**nterface

Biblioteca C vs UNIX/POSIX

C stdio.h

- fopen - abre arquivo
- fscanf - entrada formatada
- fprintf - saida formatada
- fclose - fecha arquivo
- outros (fseek, rewind, …)

+ Buffered, poderoso, sempre disponível

- buffering pode ser confuso; adiciona *overheads*

UNIX / POSIX

- open
- read
- write
- close
- outros (mmap, lseek, poll, …)

+ simples, rápido, baixo-nível, preferiuel para interfaces com o hardware

- somente para UNIX

UNIX/POSIX I/O de Arquivos

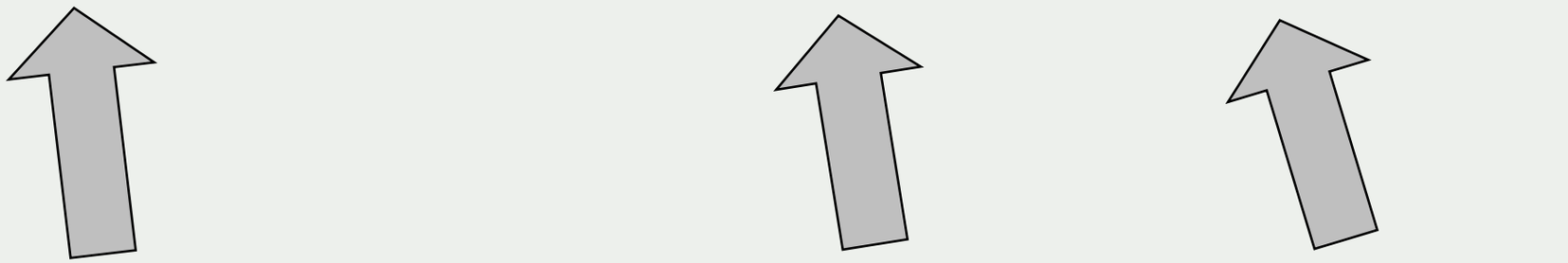
- Sem formatação - somente read/write *raw bytes*
- Preferido para trabalhar com *hardware*
- Não faz parte da Biblioteca padrão do C

POSIX open

abrir e opcionalmente caiar um arquivo para leitura e/ou escrita

- Protótipo:

```
int open(const char* path, int oflag, …)
```



"**descritor do arquivo**" (integer)

caminho do arquivo

Flag de Abertura

```
{O_WRONLY;
 O_RDONLY;
O_RDWR;...}
```

Exemplo:

```
int fd;
fd = open("/tmp/teste.txt", O_WRONLY);
```

POSIX open (Flags)

definidos em <fcntl.h>

Devem ser combinados com "OR"

O_RDONLY	open for reading only
O_WRONLY	open for writing only
O_RDWR	open for reading and writing
O_NONBLOCK	do not block on open or for data to become available
O_APPEND	append on each write
O_CREAT	create file if it does not exist
O_TRUNC	truncate size to 0
O_EXCL	error if O_CREAT and the file exists
O_SHLOCK	atomically obtain a shared lock
O_EXLOCK	atomically obtain an exclusive lock
O_NOFOLLOW	do not follow symlinks
O_SYMLINK	allow open of symlinks
O_EVTONLY	descriptor requested for event notifications only
O_CLOEXEC	mark as close-on-exec

Exemplo:(read/write, criar se não existir)

```
int fd;
fd = open("/tmp/teste.txt", O_RDWR | O_CREAT);
```

POSIX - Descritor de Arquivos

- Número inteiro que representa um "arquivo" **aberto**
- Cada processo (aplicação) possui uma tabela de descritores de arquivos (veja `/proc/<PID>/fid`). Alguns são padrão e abertos junto com a aplicação:
  - `0 - stdin`
  - `1 - stdout`
  - `2 - stderr`
- A função **open()** retorna este descritor
  - http://en.wikipedia.org/wiki/File_descriptor
- Os Tipos de "arquivo" podem ser:
  - arquivo em disco
  - diretório em disco
  - dispositivo de caracter (ex: porta serial)
  - dispositivo de bloco (ex: HDD; CDROM)
  - Tubo (***pipe***) (para comunicação interna)
  - ***socket*** (para rede)
  - outros…

Quase tudo no UNIX / Linux é representado por um arquivo

POSIX read

- Protótipo:

```
ssize_t read(int fildes, void *buf, size_t nbyte);
```

Namero de bytes lidos
ou EOF (0)
ou Error (-1)

"descritor do arquivo" (integer)

Endereço do buffer

Namero de bytes a serem lidos

```
Exemplo:
short siX16;
int res = read(fid, &siX16, sizeof(short) );
```

POSIX write

- Protótipo:

```
ssize_t write(int fildes, const void *buf, size_t nbyte);
```

Número de bytes escritos ou Error (-1)

"descritor de arquivo" (integer)

Endereço do buffer

Número de bytes a serem escritos

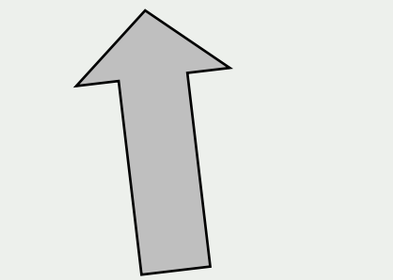
```
Exemplo:
short siX16=0x7FFF;
int res = write(fid, &siX16, sizeof(short) );
```

POSIX lseek

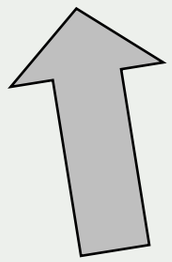
reposiciona o ponteiro do descritor do arquivo em um arquivo aberto

- Protótipo:

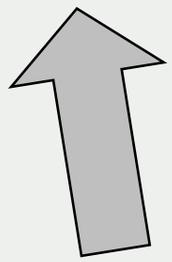
```
off_t lseek(int fd, off_t offset, int whence);
```



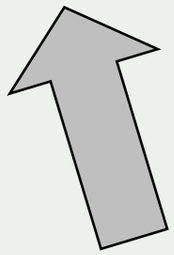
"**offset**"
(-1 se der erro)



Descritor de arquivo



Valor de *offset*



```
{SEEK_SET;
 SEEK_CUR;
 SEEK_END}
```

POSIX lseek

reposiciona o ponteiro do descritor do arquivo em um arquivo aberto

- Protótipo:

```
off_t lseek(int fd, off_t offset, int whence);
```

Exemplo:

O Arquivo teste.txt contém o texto
"Hello World"

```
int fd;
char c;
fd = open("/tmp/teste.txt", O_RDONLY);
lseek(fd, 6, SEEK_SET);
read(fd, &c, 1);
```

Qual caracter será gravado em **c**?

POSIX close

- Protótipo:

```
int close(int fildes)
```

OK (0)
Error (-1)

"descritor de arquivo"
(integer)

Exemplo:

```
int iStatus = close(fd);
```

Exemplo: UART

```c
#include <stdio.h>
#include <unistd.h>          //Used for UART
#include <fcntl.h>           //Used for UART
#include <termios.h>    //Used for UART

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

    int uart0_filestream = -1;

    uart0_filestream = open("/dev/ttyAMA0", O_RDWR | O_NOCTTY | O_NDELAY);     //Open in
non blocking read/write mode
    if (uart0_filestream == -1)
    {
    printf("Error - Unable to open UART.  Ensure it is not in use by another application
\n");
    }
    struct termios options;
    tcgetattr(uart0_filestream, &options);
    options.c_cflag = B9600 | CS8 | CLOCAL | CREAD;    //<Set baud rate
    options.c_iflag = IGNPAR;
    options.c_oflag = 0;
    options.c_lflag = 0;
    tcflush(uart0_filestream, TCIFLUSH);
    tcsetattr(uart0_filestream, TCSANOW, &options);
```

Exemplo: UART

```
unsigned char tx_buffer[20];
unsigned char *p_tx_buffer;

p_tx_buffer = &tx_buffer[0];
*p_tx_buffer++ = 'H';
*p_tx_buffer++ = 'e';
*p_tx_buffer++ = 'l';
*p_tx_buffer++ = 'l';
*p_tx_buffer++ = 'o';

if (uart0_filestream != -1)
{
 int count = write(uart0_filestream, &tx_buffer[0], (p_tx_buffer - &tx_buffer[0]));
 if (count < 0)
 {
      printf("UART TX error\n");
 }
 }

 if (uart0_filestream != -1)
 {
 unsigned char rx_buffer[256];
 int rx_length = read(uart0_filestream, (void*)rx_buffer, 255);
 if (rx_length < 0)
 {         }
 else if (rx_length == 0)
 { }
 else
 {
      rx_buffer[rx_length] = '\0';
      printf("%i bytes read : %s\n", rx_length, rx_buffer);
 }
 }
  close(uart0_filestream);
return 0;
}
```