

# Programando Assembly no Linux x86

+ Se quisermos, por exemplo, escrever alguma coisa na tela, devemos chamar a `write`

→ Para achar essa syscall e seu nº:

```
nano /usr/include/x86_64-linux-gnu/asm/unistd_32.h
```

```
GNU nano 7.2
#ifndef _ASM_UNISTD_32_H
#define _ASM_UNISTD_32_H

#define __NR_restart_syscall 0
#define __NR_exit 1
#define __NR_fork 2
#define __NR_read 3
#define __NR_write 4
#define __NR_open 5
#define __NR_close 6
#define __NR_waitpid 7
#define __NR_creat 8
#define __NR_link 9
#define __NR_unlink 10
```

→ O número da write é 4, então moveremos isso pra `eax`

→ Se dermos um `man write`, ele não vai falar da funcionalidade da write, então

daremos um `man man` para ver a funcionalidade da man

```
The table below shows the section numbers of the manual follo

1 Executable programs or shell commands
2 System calls (functions provided by the kernel)
3 Library calls (functions within program libraries)
4 Special files (usually found in /dev)
5 File formats and conventions, e.g. /etc/passwd
```

→ Para ver a funcionalidade de funções do tipo system calls, devemos por o

parâmetro 2 antes do nome da função: `man 2 write`

```
NAME
    write - write to a file descriptor

LIBRARY
    Standard C library (libc, -lc)

SYNOPSIS
    #include <unistd.h>

    ssize_t write(int fd, const void buf[.count], size_t count);
```

→ `fd` = file descriptor (`stdin` (0), `stdout`(1) e `stderr`(2))

→ Para ter uma descrição um pouco mais prática: <https://syscalls.w3challs.com/?arch=x86>

<a href="#">←</a> <a href="#">→</a> <a href="#">↺</a> <a href="#">🏠</a> <a href="#">🔒</a> <a href="https://syscalls.w3challs.com/7arch=x86">https://syscalls.w3challs.com/7arch=x86</a> <a href="#">☆</a> <a href="#">🔒</a> <a href="#">🔖</a>									
<a href="#">Kali Linux</a> <a href="#">Kali Tools</a> <a href="#">Kali Docs</a> <a href="#">Kali Forums</a> <a href="#">Kali NetHunter</a> <a href="#">Exploit-DB</a> <a href="#">blob:https://web.what... </a> <a href="#">Google Hacking DB</a> <a href="#">OffSec</a> <a href="#">Virtual Lab - Desec Sec...</a>									
Show 10 entries <span style="float: right;">Search: <input type="text"/></span>									
#	Name	Registers							Definition
		eax	ebx	ecx	edx	esi	edi	ebp	
0	restart_syscall	0x00	-	-	-	-	-	-	kernel/signal.c:2501
1	exit	0x01	int error_code	-	-	-	-	-	kernel/exit.c:1095
2	fork	0x02	-	-	-	-	-	-	arch/x86/kernel/process.c:271
3	read	0x03	unsigned int fd	char *buf	size_t count	-	-	-	fs/read_write.c:460
4	write	0x04	unsigned int fd	const char *buf	size_t count	-	-	-	fs/read_write.c:477

```
-----
ass.asm-----
global _main
```

```
section .data
```

```
    curso: db "Desec Security",0xa ; o 0xa representa o /n, que quebra linha
```

```
section .text
```

```
_main:
```

```
    mov eax, 4 ; chama a write
    mov ebx, 1 ; apresenta o stdout
    mov ecx, curso
    mov edx, 15 ; conta a qtd de caracteres
    int 0x80 ; chama a syscall
```

```
    mov eax, 1 ; 1 é a syscall do exit, saída
    mov ebx, 0
    int 0x80
```

```
nasm -f elf32 ass.asm
```

```
ld --entry _main -m elf_i386 ass.o -o ass
```