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 foll

   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

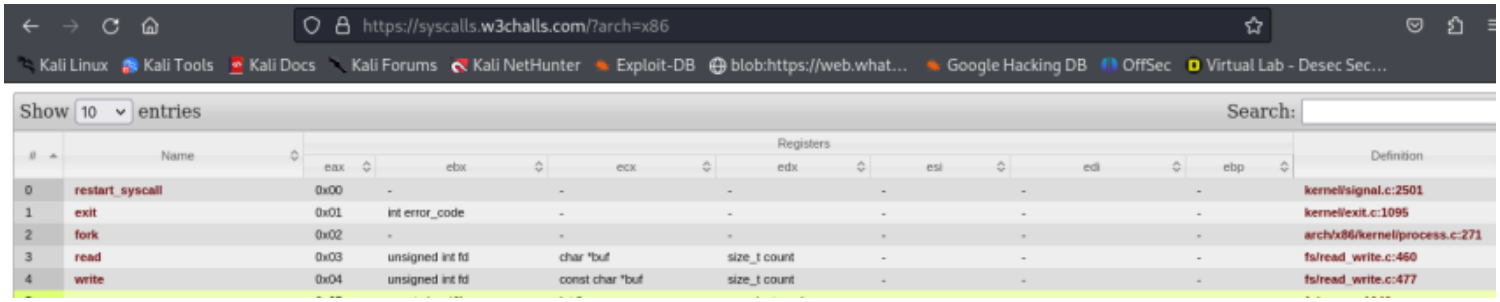
```
NAME
       write - write to a file descriptor

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 stderror(2))
→ Para ter uma descrição um pouco mais prática: https://syscalls.w3challs.com/?arch=x86



---
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
```