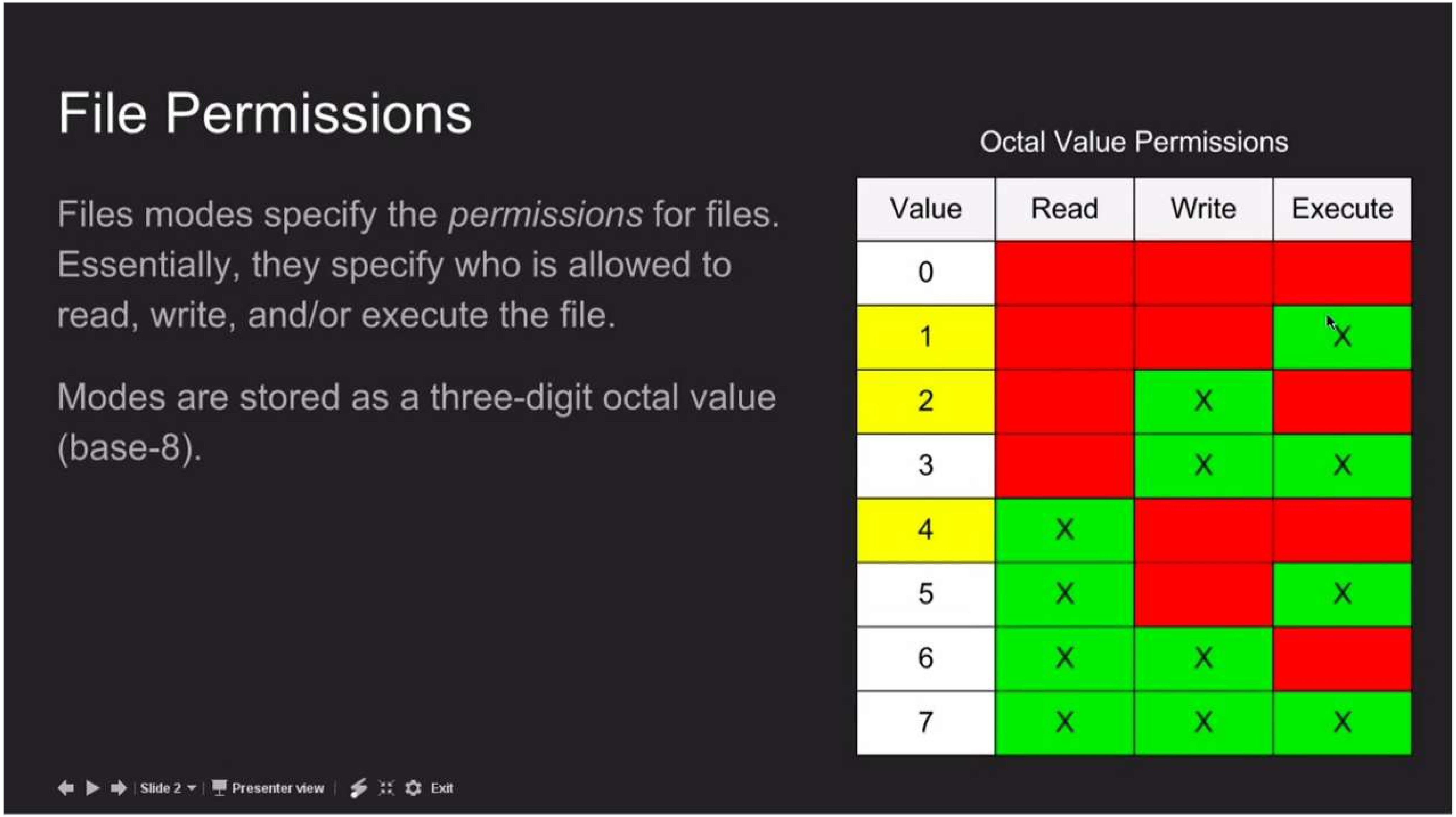


File Permissions

Files modes specify the *permissions* for files. Essentially, they specify who is allowed to read, write, and/or execute the file.

Modes are stored as a three-digit octal value (base-8).

Octal Value Permissions

Value	Read	Write	Execute
0			
1			X
2		X	
3		X	X
4	X		
5	X		X
6	X	X	
7	X	X	X

File Permissions

File permissions in Linux are set with four octal values. The least 3 significant octal values are for the file owner's permissions, the group's permissions, and the "other's" permissions (those outside the group and not the file owner).

The most significant octal value is reserved for special permissions.

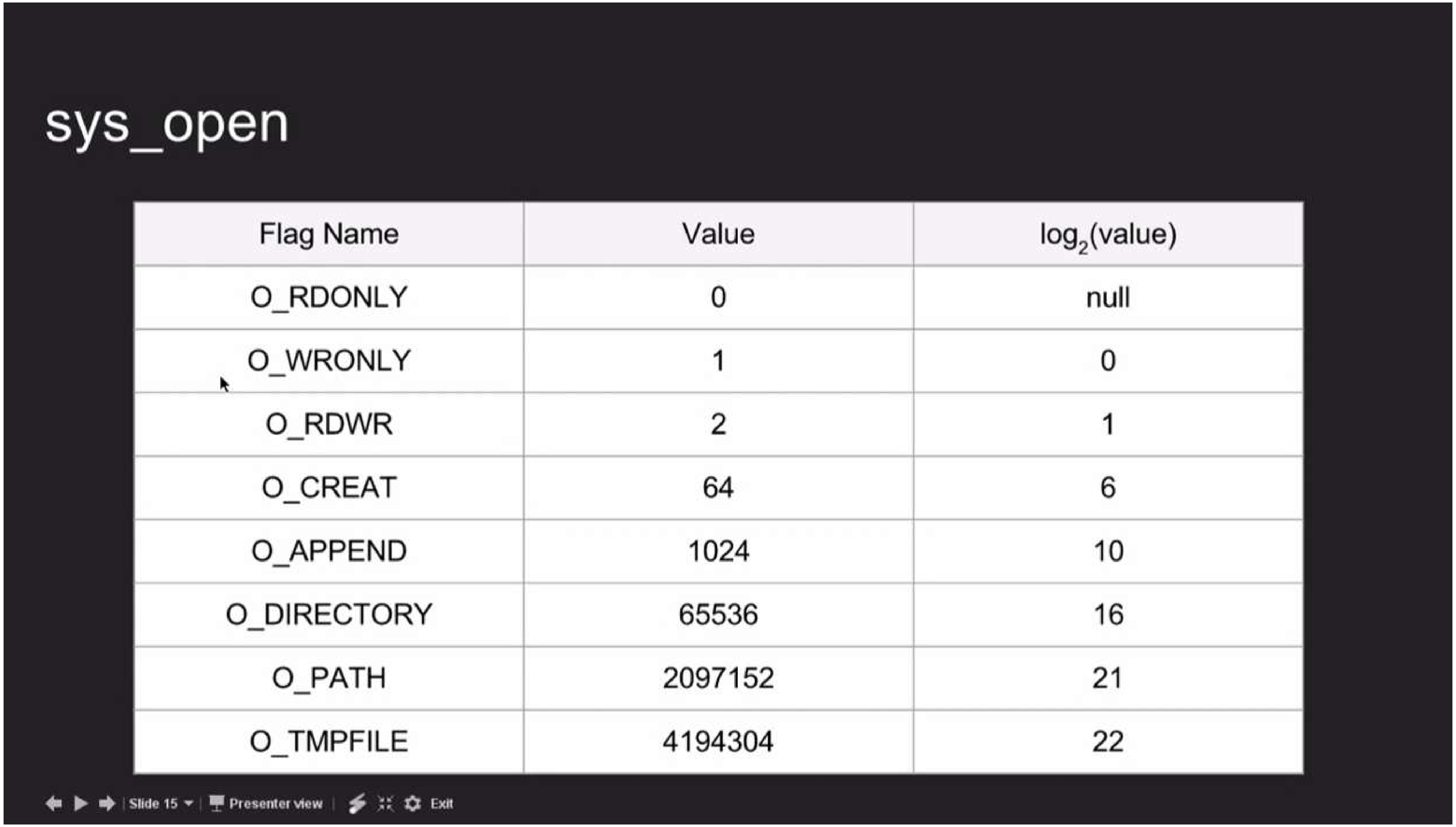
Usually we do not need to use these.

	Special	Owner	Group	Other
1	sticky bit	execute	execute	execute
2	setgid	write	write	write
4	setuid	read	read	read

System Calls

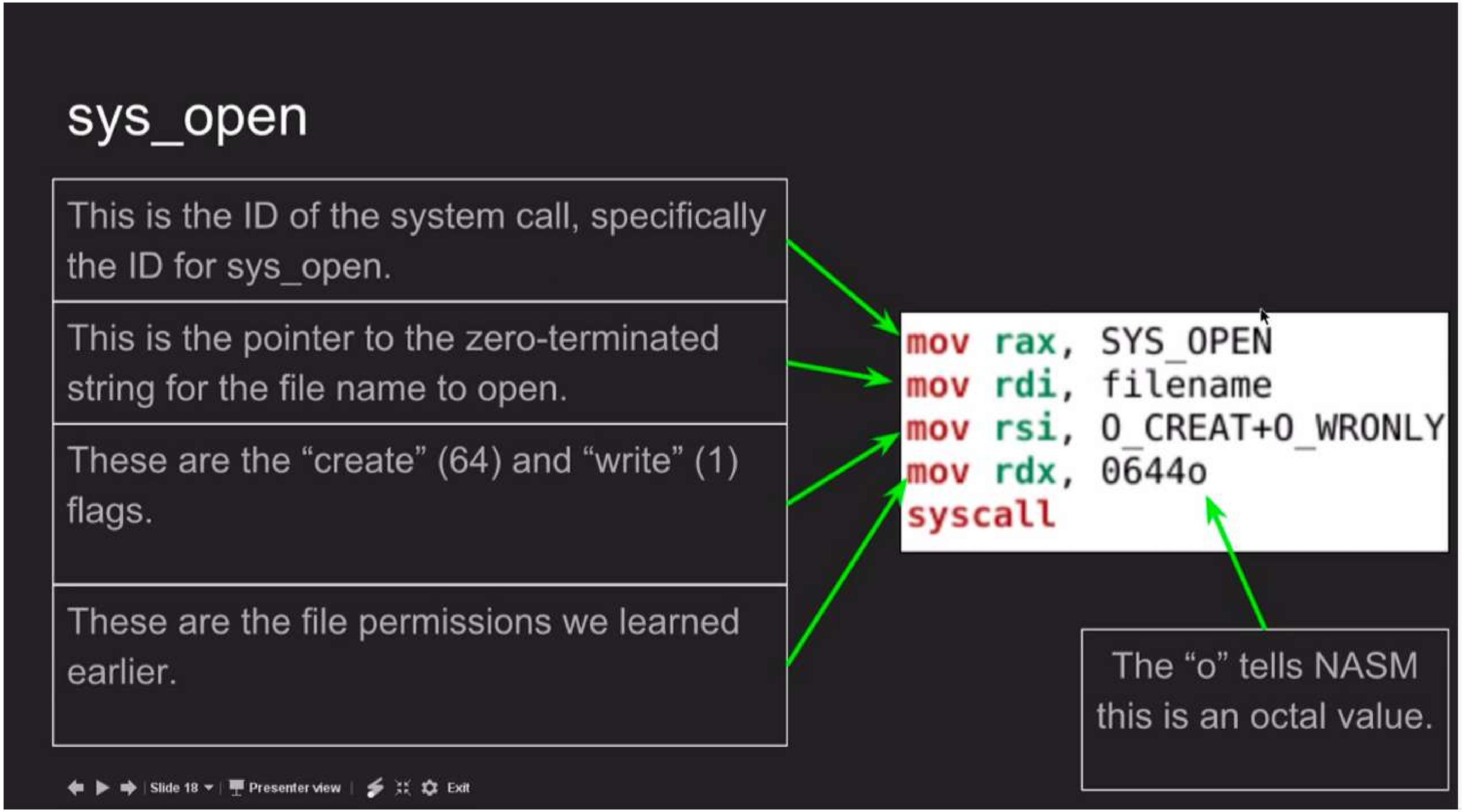
Recall the table of system calls.

syscall	ID	ARG1	ARG2	ARG3	ARG4	ARG5	ARG6
sys_read	0	#filedescriptor	$buffer	#count			
sys_write	1	#filedescriptor	$buffer	#count			
sys_open	2	$filename	#flags	#mode			
sys_close	3	#filedescriptor					
...	...	...	...	...	...	...	...
pwritev2	328	...	...	...	...	...	...



sys_open

Flag Name	Value	$\log_2$(value)
O_RDONLY	0	null
O_WRONLY	1	0
O_RDWR	2	1
O_CREAT	64	6
O_APPEND	1024	10
O_DIRECTORY	65536	16
O_PATH	2097152	21
O_TMPFILE	4194304	22



sys_open

This is the ID of the system call, specifically the ID for sys_open.

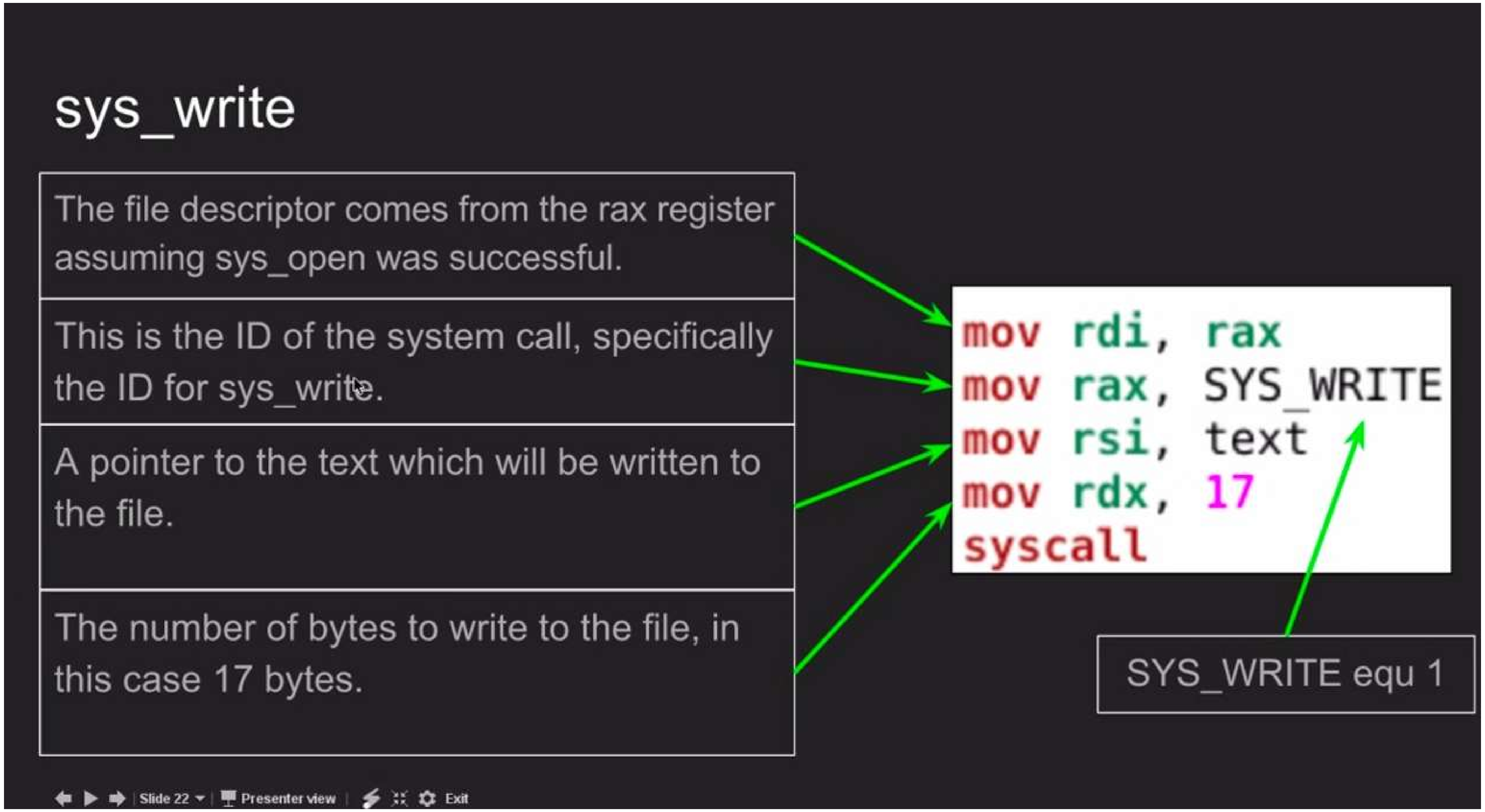
This is the pointer to the zero-terminated string for the file name to open.

These are the "create" (64) and "write" (1) flags.

These are the file permissions we learned earlier.

```
mov rax, SYS_OPEN
mov rdi, filename
mov rsi, O_CREAT+O_WRONLY
mov rdx, 0644o
syscall
```

The "o" tells NASM this is an octal value.



sys_write

The file descriptor comes from the rax register assuming sys_open was successful.

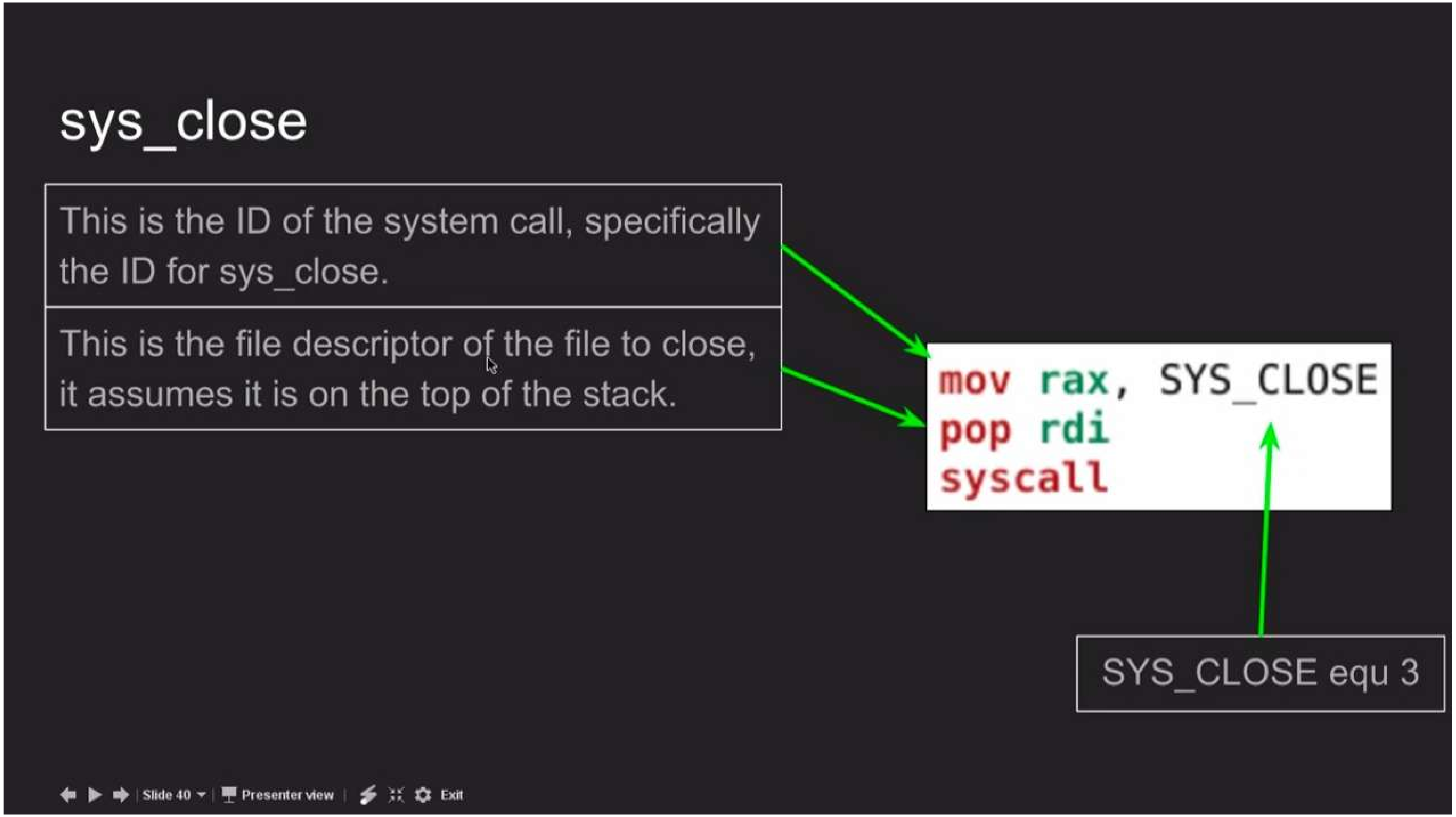
This is the ID of the system call, specifically the ID for sys_write.

A pointer to the text which will be written to the file.

The number of bytes to write to the file, in this case 17 bytes.

```
mov rdi, rax
mov rax, SYS_WRITE
mov rsi, text
mov rdx, 17
syscall
```

SYS_WRITE equ 1



sys_close

This is the ID of the system call, specifically the ID for sys_close.

This is the file descriptor of the file to close, it assumes it is on the top of the stack.

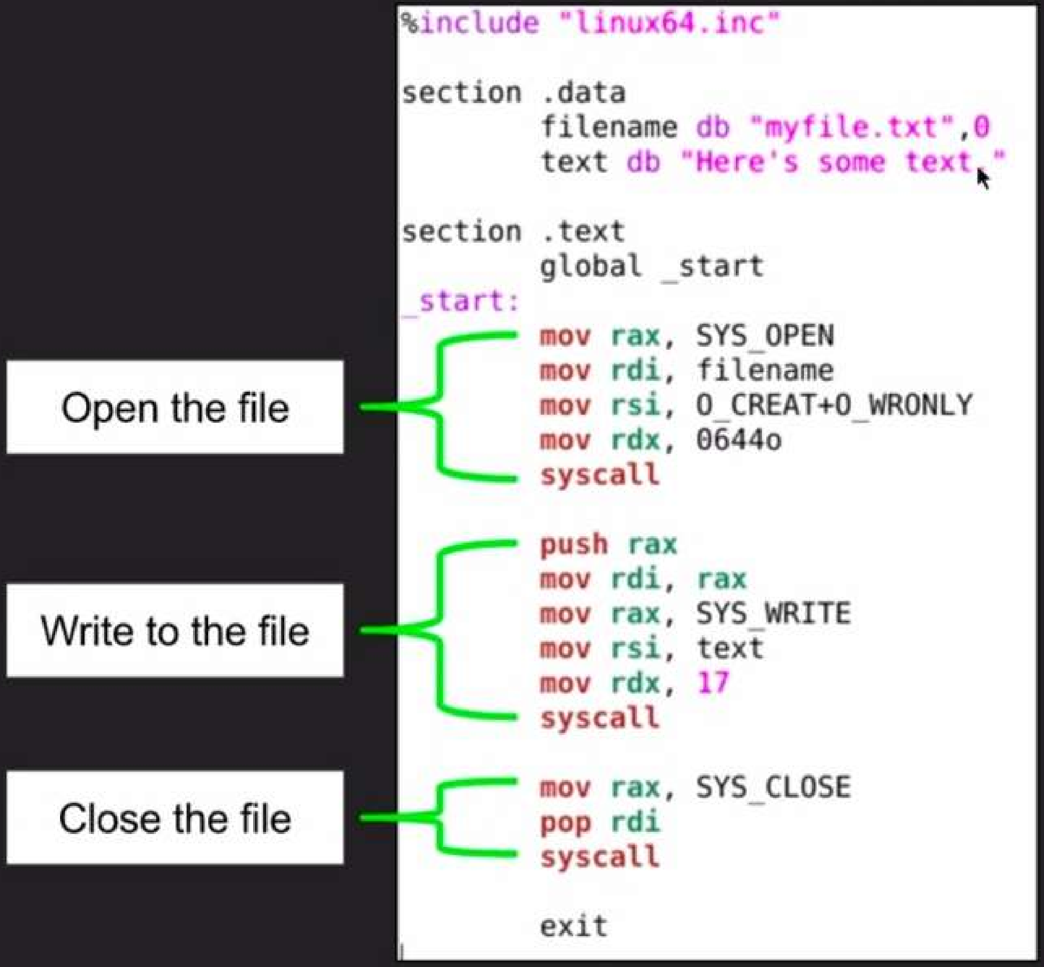
```
mov rax, SYS_CLOSE
pop rdi
syscall
```

SYS_CLOSE equ 3

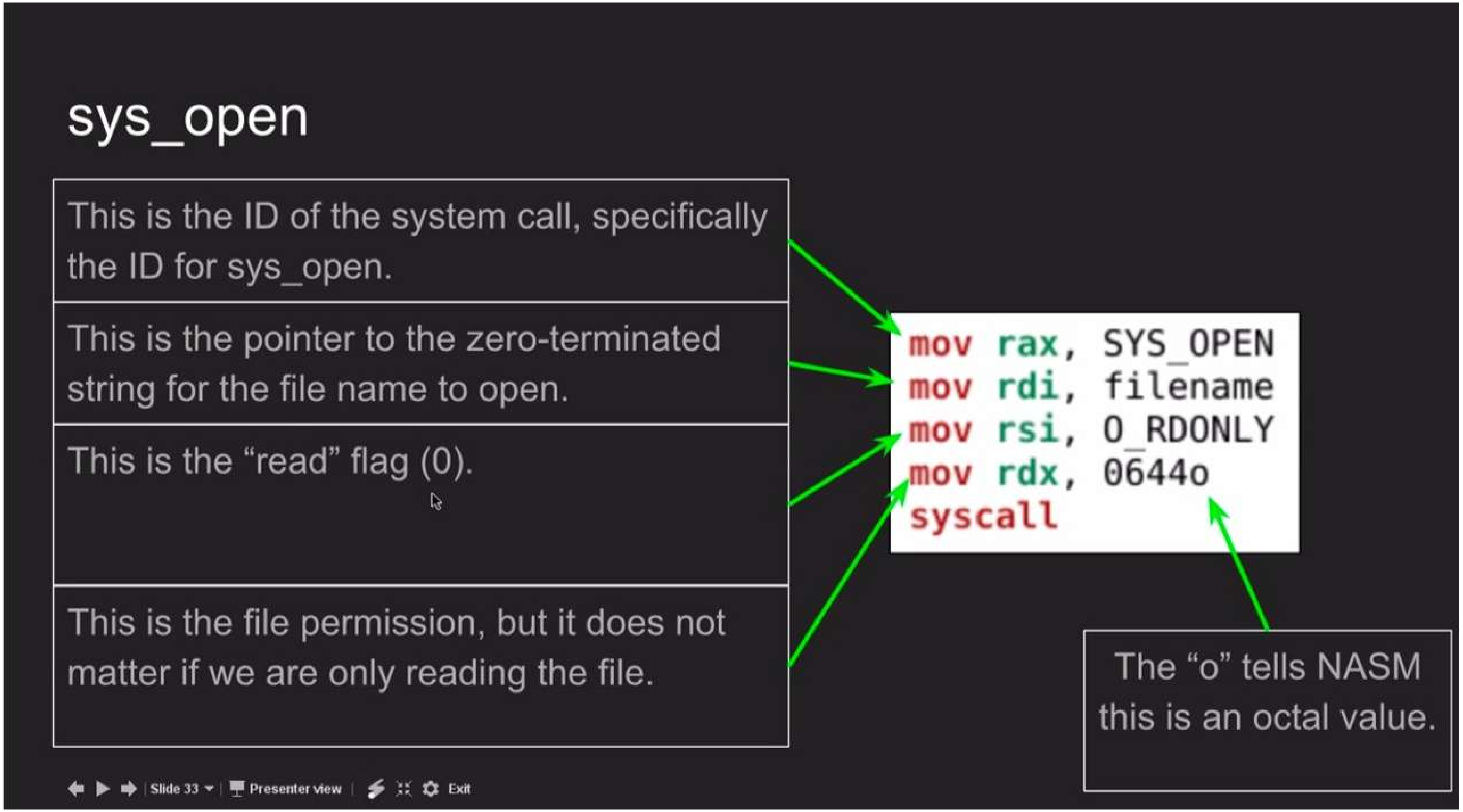
Code to Write to a File

Find this code here:

http://pastebin.com/b9TUxfzg



```
%include "linux64.inc"

section .data
        filename db "myfile.txt",0
        text db "Here's some text."

section .text
        global _start
_start:
        mov rax, SYS_OPEN
        mov rdi, filename
        mov rsi, O_CREAT+O_WRONLY
        mov rdx, 0644o
        syscall

        push rax
        mov rdi, rax
        mov rax, SYS_WRITE
        mov rsi, text
        mov rdx, 17
        syscall

        mov rax, SYS_CLOSE
        pop rdi
        syscall

        exit
```



sys_open

This is the ID of the system call, specifically the ID for sys_open.

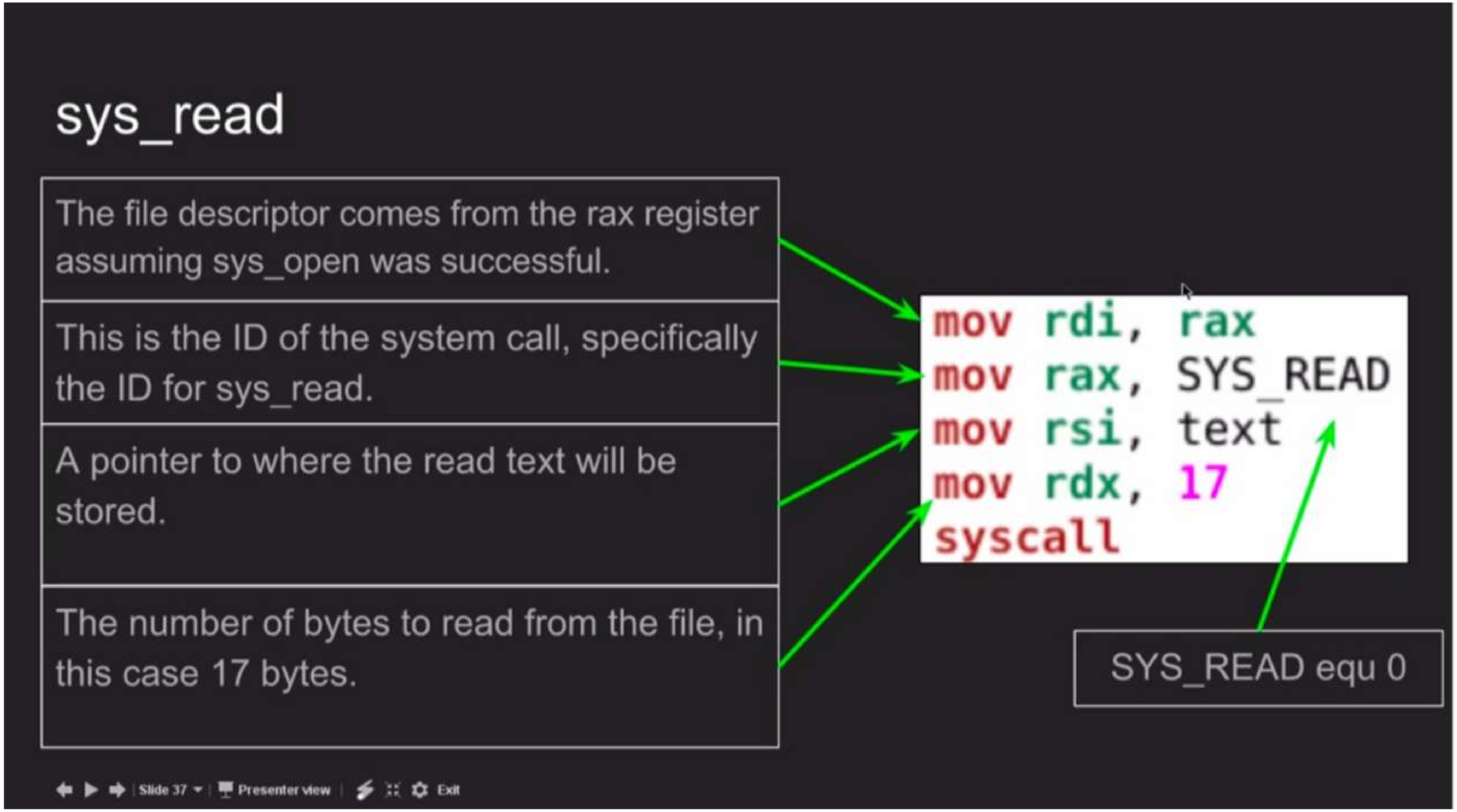
This is the pointer to the zero-terminated string for the file name to open.

This is the "read" flag (0).

This is the file permission, but it does not matter if we are only reading the file.

```
mov rax, SYS_OPEN
mov rdi, filename
mov rsi, O_RDONLY
mov rdx, 0644o
syscall
```

The "o" tells NASM this is an octal value.



sys_read

The file descriptor comes from the rax register assuming sys_open was successful.

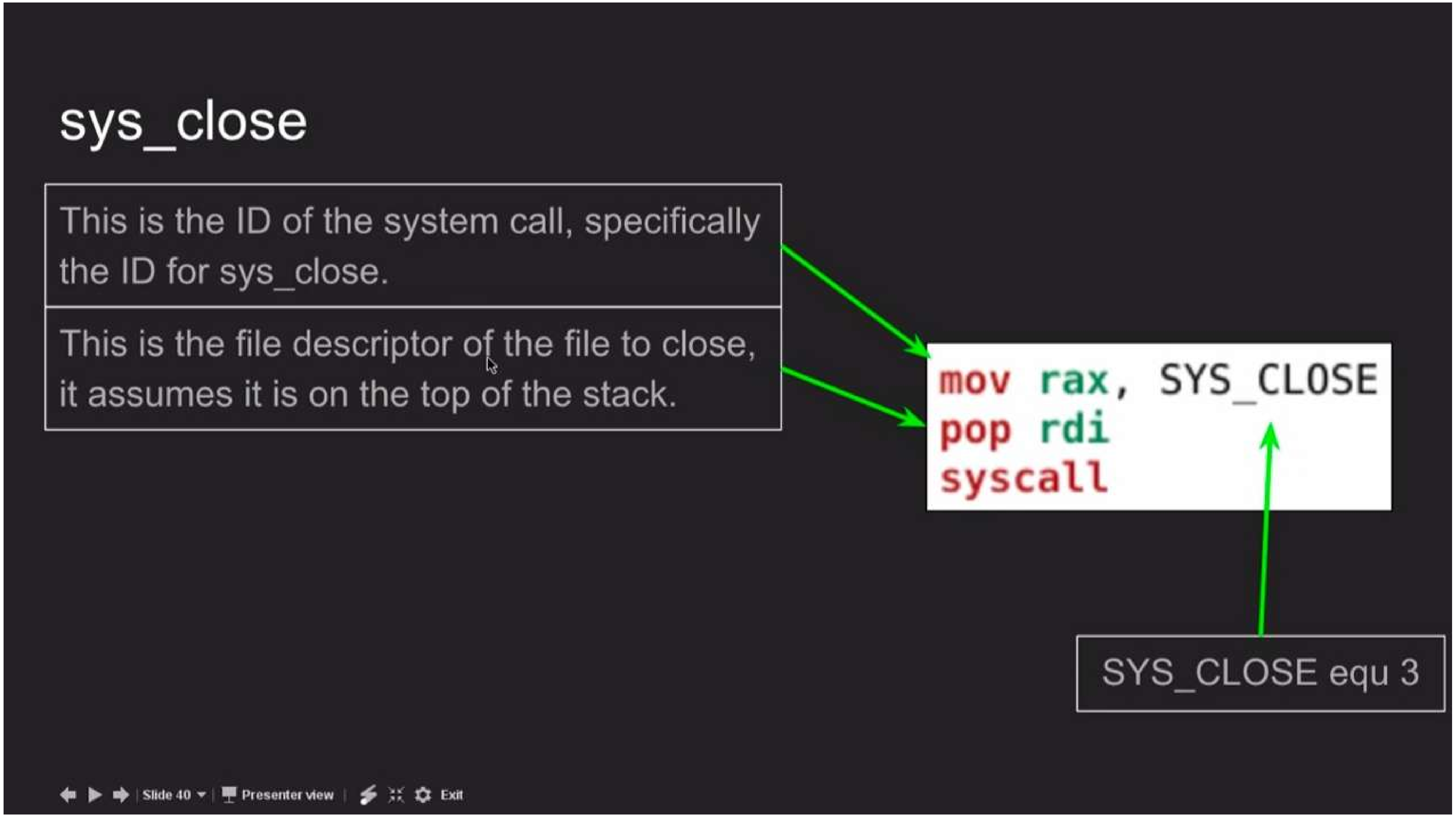
This is the ID of the system call, specifically the ID for sys_read.

A pointer to where the read text will be stored.

The number of bytes to read from the file, in this case 17 bytes.

```
mov rdi, rax
mov rax, SYS_READ
mov rsi, text
mov rdx, 17
syscall
```

SYS_READ equ 0



sys_close

This is the ID of the system call, specifically the ID for sys_close.

This is the file descriptor of the file to close, it assumes it is on the top of the stack.

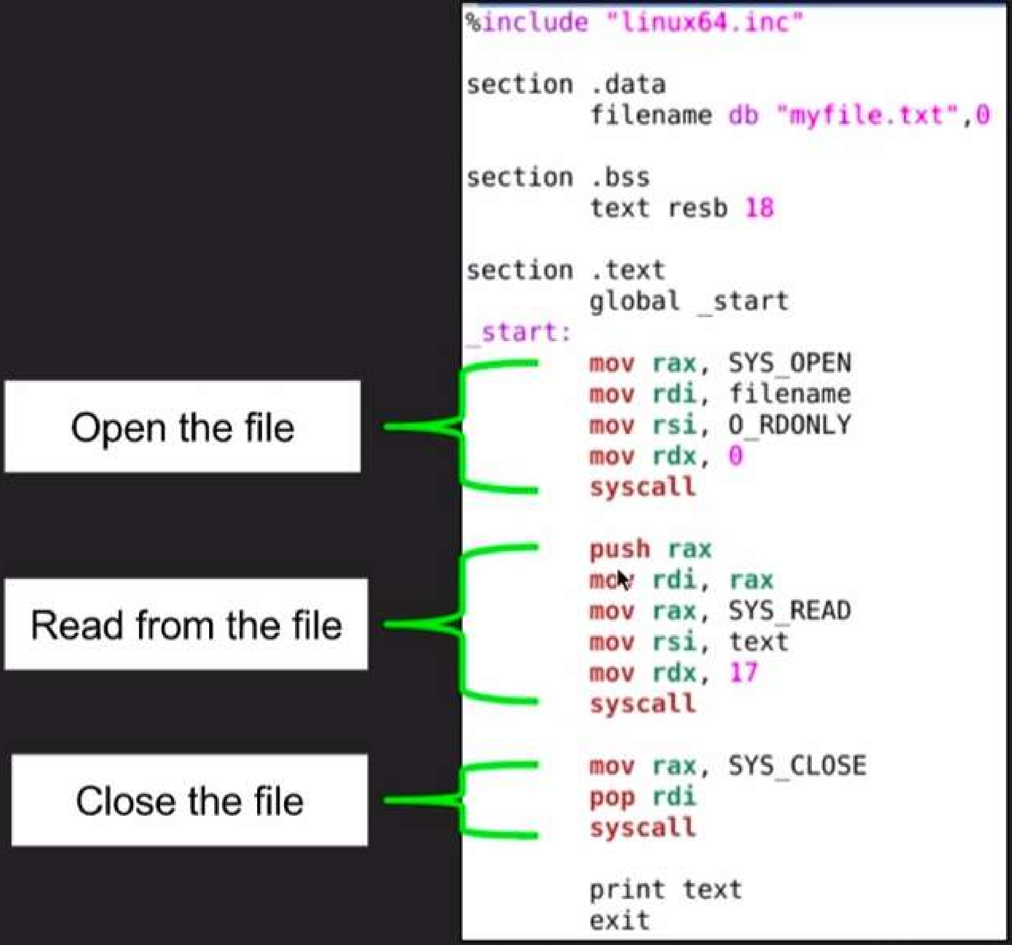
```
mov rax, SYS_CLOSE
pop rdi
syscall
```

SYS_CLOSE equ 3

Code to Write to a File

Find this code here:

http://pastebin.com/xcFtXk3t



```
%include "linux64.inc"

section .data
        filename db "myfile.txt",0

section .bss
        text resb 18

section .text
        global _start
_start:
        mov rax, SYS_OPEN
        mov rdi, filename
        mov rsi, O_RDONLY
        mov rdx, 0
        syscall

        push rax
        mov rdi, rax
        mov rax, SYS_READ
        mov rsi, text
        mov rdx, 17
        syscall

        mov rax, SYS_CLOSE
        pop rdi
        syscall

        print text
        exit
```