# Linux Assembly

Armine Hayrapetyan

#### Outline

- Introduction to Files
- Open Files
- Writing Files
- Reading Files

### 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			Ą
1			Х
2		Х	
3		Х	Х
4	Х		
5	Х		Х
6	Х	Х	
7	Х	Х	Х

#### 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

We have previously used sys\_read and sys\_write to read text from and write text to the standard input and output respectively.

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						

## System Calls

sys\_close is used when a file is no longer in use.

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						

The first argument sys\_open takes is a pointer to the filename (zero terminated).

The second argument are the flags.

The third argument is the file mode, being the 4-digit octal number that we learned from earlier.

syscall	ID	ARG1	ARG2	ARG3	ARG4	ARG5	ARG6
sys_open	2	\$filename	#flags	#mode			

Here is the code to open a file with the "create" and "write" flag.

The "create" flag creates the file if it does not exist.

```
mov rax, SYS_OPEN
mov rdi, filename
mov rsi, O_CREAT+O_WRONLY
mov rdx, 06440
syscall
```

This is the ID of the system call, specifically the ID for sys\_open. This is the pointer to the zero-terminated mov rax, SYS OPEN mov rdi, filename string for the file name to open. mov rsi, 0 CREAT+0 WRONLY These are the "create" (64) and "write" (1) mov rdx, 06440 flags. syscall These are the file permissions we learned The "o" tells NASM earlier. this is an octal value.

This system call returns the file descriptor of the file opened within the rax register.

If there is an error, that error is returned in the rax register.

### sys\_write

This system call can be used to write text to a file.

It is used exactly like how we used it in the "Hello, World!" video to display text on the screen. The only difference is that the first argument is changed to the file descriptor returned from the sys open system call.

syscall	ID	ARG1	ARG2	ARG3	ARG4	ARG5	ARG6
sys_write	1	#filedescriptor	\$buffer	#count			

### sys\_write

Here is code to write text to a file opened.

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

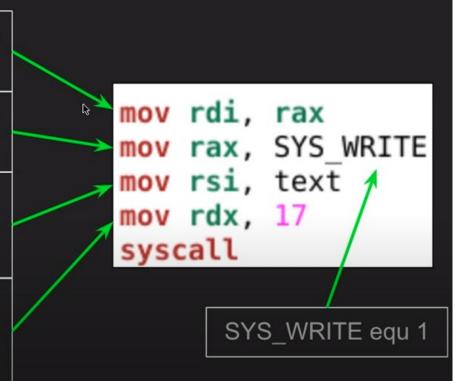
### sys\_write

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.



sys\_close only takes the file descriptor as its only argument.

syscall	ID	ARG1	ARG2	ARG3	ARG4	ARG5	ARG6
sys_close	3	#filedescriptor					

Here is code to close an opened file.

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

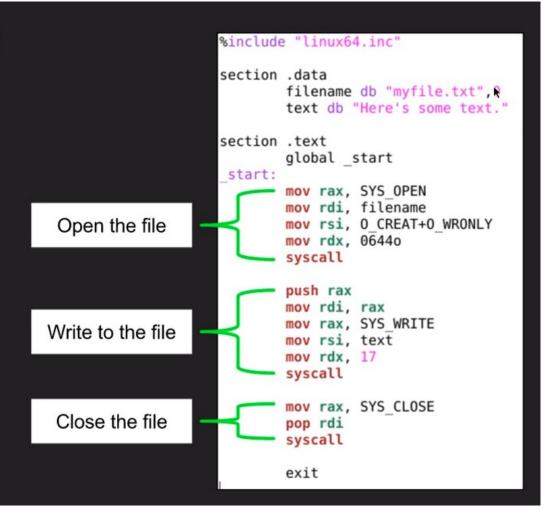
This is the ID of the system call, specifically the ID for sys\_close. This is the file descriptor of the file to close, mov rax, SYS CLOSE it assumes it is on the top of the stack. syscall

SYS CLOSE equ 3

#### Code to Write to a File

Find this code here:

http://pastebin.com/b9TUxfzg



The first argument sys\_open takes is a pointer to the filename (zero terminated).

The second argument are the flags.

The third argument is the file mode, being the 4-digit octal number that we learned from earlier.

syscall	ID	ARG1	ARG2	ARG3	ARG4	ARG5	ARG6
sys_open	2	\$filename	#flags	#mode			

Flag Name	Value	log <sub>2</sub> (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

Here is the code to open a file with the "read only" flag.

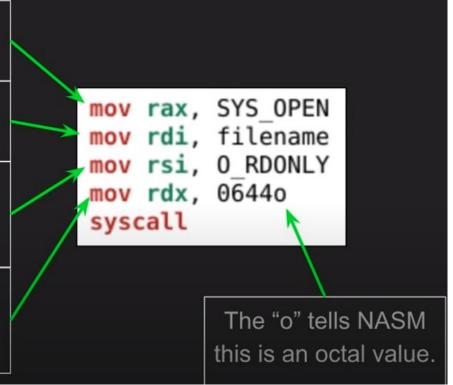
mov rax, SYS\_OPEN
mov rdi, filename
mov rsi, O\_RDONLY
mov rdx, 06440
syscall

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.



This system call returns the file descriptor of the file opened within the rax register.

If there is an error, that error is returned in the rax register.

### sys\_read

This system call can be used to read text from a file.

It is used exactly like how we used it in the tutorial on getting user input. The only difference is that the first argument is changed to the file descriptor returned from the sys\_open system call.

 syscall
 ID
 ARG1
 ARG2
 ARG3
 ARG4
 ARG5
 ARG6

 sys\_read
 0
 #filedescriptor
 \$buffer
 #count

### sys\_read

Here is code to read from an opened file.

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

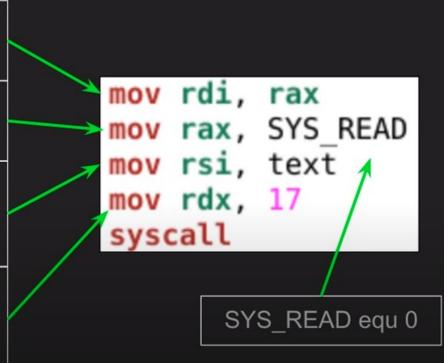
### sys\_read

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.



This is the ID of the system call, specifically the ID for sys close. This is the file descriptor of the file to close, mov rax, SYS CLOSE it assumes it is on the top of the stack. 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
  Open the file
                                    mov rsi, 0 RDONLY
                       (g
                                    mov rdx, 0
                                    syscall
                                    push rax
                                    mov rdi, rax
                                    mov rax, SYS READ
Read from the file
                                    mov rsi, text
                                    mov rdx, 17
                                    syscall
                                    mov rax, SYS CLOSE
   Close the file
                                    pop rdi
                                    syscall
                                    print text
                                    exit
```

#### References

- https://www.youtube.com/watch?v=AwmhZUATGYM&list=PLetF-YjXm-sCH6 FrTz4AQhfH6INDQvQSn&index=10
- https://www.youtube.com/watch?v=vXsUIX\_Ozgc&list=PLetF-YjXm-sCH6FrT z4AQhfH6INDQvQSn&index=11
- https://www.youtube.com/watch?v=vXsUIX\_Ozgc&list=PLetF-YjXm-sCH6FrTz4AQhfH6INDQvQSn&index=12