Level 0x09

Kernel Interfacing

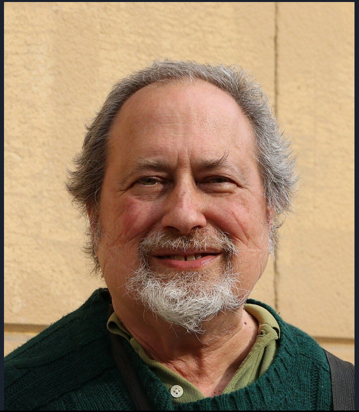
Topics

- Events
- Hacker History
- Linux syscalls

Upcoming Events

- December hacking contests
  - Advent of Cyber - TryHackMe.com
  - Sans Institute Holiday Hack Challenge - Less CTF, more game, probably more incident responder and defender focused
  - Advent of Code - Programming challenges. Used to be 25 2-parters, now reducing down to 12 days of challenges
  - Pwn.College is going to do an Advent of Pwn

Richard Stallman aka rms



- Harvard Physics graduate
- Worked at MIT AI Lab… where a printer triggered him…
- Started the GNU Project (GNUs Not Unix)
    - GNU Compiler Collection (GCC), GNU make
    - GNU Debugger (GDB)
    - GPL License (Open Source License)
    - Free Software Foundation (Freedom: Libre vs Gratis)

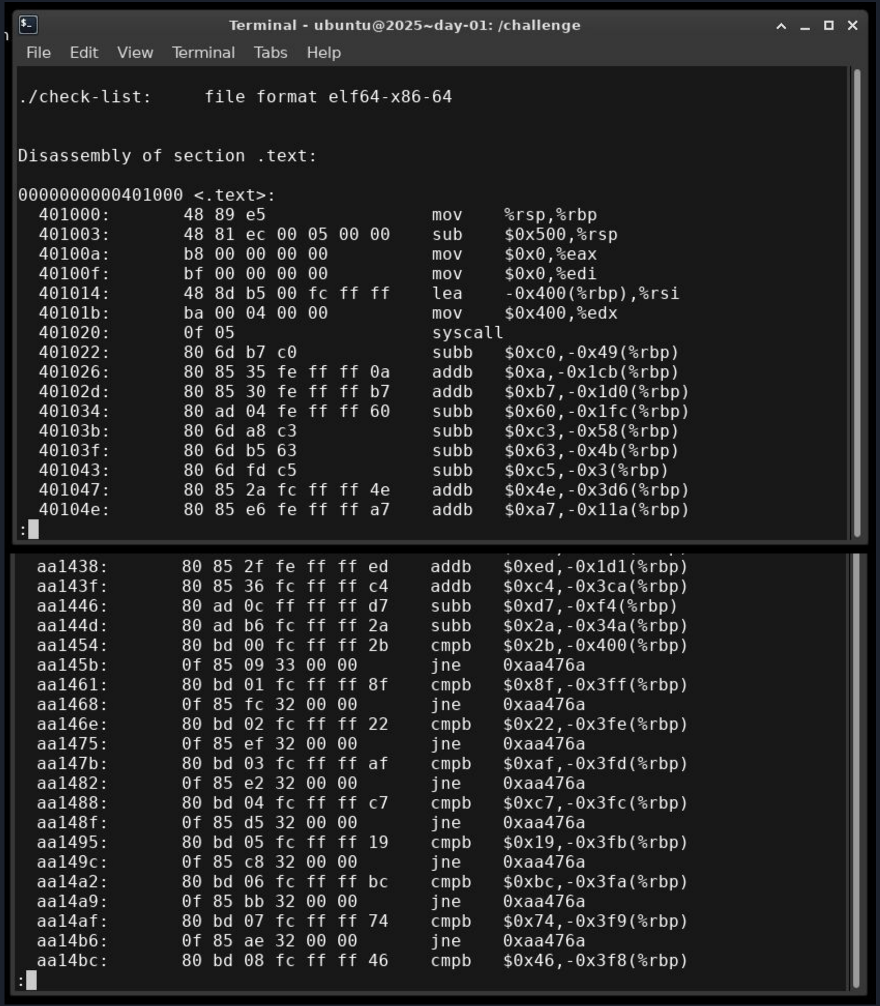
- MIT AI Lab had leadership involved with Jeffrey Epstein
- Has an insanely large speech rider
    - What pets are preferred at the place of stay
    - What codecs you are allowed to distribute speech in
    - Don’t even mention breakfast

Advent of Pwn Day 1

- Hint tells you to run objdump
  - -d is for disassemble

What we find in checklist:

- Some stuff
- syscall
- subb / addb instructions
  - Sooo many of these
  - Really, pages of them
- cmpb and jne instructions

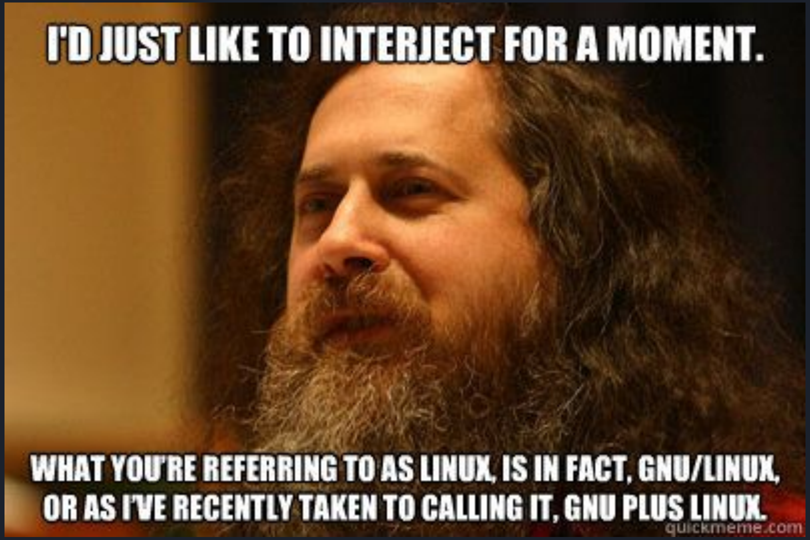


```
./check-list:     file format elf64-x86-64


Disassembly of section .text:

0000000000401000 <.text>:
  401000:       48 89 e5                mov    %rsp,%rbp
  401003:       48 81 ec 00 05 00 00    sub    $0x500,%rsp
  40100a:       b8 00 00 00 00          mov    $0x0,%eax
  40100f:       bf 00 00 00 00          mov    $0x0,%edi
  401014:       48 8d b5 00 fc ff ff    lea    -0x400(%rbp),%rsi
  40101b:       ba 00 04 00 00          mov    $0x400,%edx
  401020:       0f 05                   syscall
  401022:       80 6d b7 c0             subb   $0xc0,-0x49(%rbp)
  401026:       80 85 35 fe ff ff 0a    addb   $0xa,-0x1cb(%rbp)
  40102d:       80 85 30 fe ff ff b7    addb   $0xb7,-0x1d0(%rbp)
  401034:       80 ad 04 fe ff ff 60    subb   $0x60,-0x1fc(%rbp)
  40103b:       80 6d a8 c3             subb   $0xc3,-0x58(%rbp)
  40103f:       80 6d b5 63             subb   $0x63,-0x4b(%rbp)
  401043:       80 6d fd c5             subb   $0xc5,-0x3(%rbp)
  401047:       80 85 2a fc ff ff 4e    addb   $0x4e,-0x3d6(%rbp)
  40104e:       80 85 e6 fe ff ff a7    addb   $0xa7,-0x11a(%rbp)
:
```

```
  aa1438:       80 85 2f fe ff ff ed    addb   $0xed,-0x1d1(%rbp)
  aa143f:       80 85 36 fc ff ff c4    addb   $0xc4,-0x3ca(%rbp)
  aa1446:       80 ad 0c ff ff ff d7    subb   $0xd7,-0xf4(%rbp)
  aa144d:       80 ad b6 fc ff ff 2a    subb   $0x2a,-0x34a(%rbp)
  aa1454:       80 bd 00 fc ff ff 2b    cmpb   $0x2b,-0x400(%rbp)
  aa145b:       0f 85 09 33 00 00       jne    0xaa476a
  aa1461:       80 bd 01 fc ff ff 8f    cmpb   $0x8f,-0x3ff(%rbp)
  aa1468:       0f 85 fc 32 00 00       jne    0xaa476a
  aa146e:       80 bd 02 fc ff ff 22    cmpb   $0x22,-0x3fe(%rbp)
  aa1475:       0f 85 ef 32 00 00       jne    0xaa476a
  aa147b:       80 bd 03 fc ff ff af    cmpb   $0xaf,-0x3fd(%rbp)
  aa1482:       0f 85 e2 32 00 00       jne    0xaa476a
  aa1488:       80 bd 04 fc ff ff c7    cmpb   $0xc7,-0x3fc(%rbp)
  aa148f:       0f 85 d5 32 00 00       jne    0xaa476a
  aa1495:       80 bd 05 fc ff ff 19    cmpb   $0x19,-0x3fb(%rbp)
  aa149c:       0f 85 c8 32 00 00       jne    0xaa476a
  aa14a2:       80 bd 06 fc ff ff bc    cmpb   $0xbc,-0x3fa(%rbp)
  aa14a9:       0f 85 bb 32 00 00       jne    0xaa476a
  aa14af:       80 bd 07 fc ff ff 74    cmpb   $0x74,-0x3f9(%rbp)
  aa14b6:       0f 85 ae 32 00 00       jne    0xaa476a
  aa14bc:       80 bd 08 fc ff ff 46    cmpb   $0x46,-0x3f8(%rbp)
:
```

What is inside an OS kernel

- Hardware Interfacing
  - Memory Access
  - SSD / Hard Drive control
  - Graphics Drivers
  - Keyboard / mouse interfacing
  - Ethernet / Wi-Fi
- Process control (threads)
- Filesystems
- Memory access
- Networking
- Enforces privilege

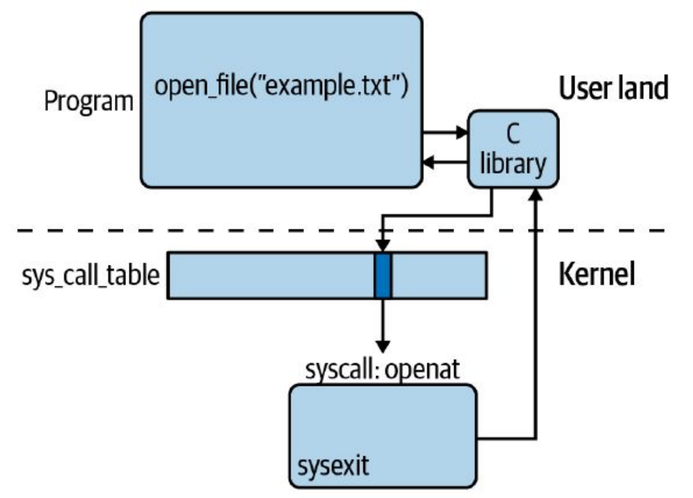

I'D JUST LIKE TO INTERJECT FOR A MOMENT.
WHAT YOU'RE REFERRING TO AS LINUX, IS IN FACT, GNU/LINUX, OR AS I'VE RECENTLY TAKEN TO CALLING IT, GNU PLUS LINUX.
quickmeme.com

Trap / Kernel syscall

- We program / use applications in userspace
  - Code written in Java, Python, C/C++
  - Code is converted to machine code for our CPU
- We call libraries (other functions) to help us…
  - Read / write data from user or terminal
  - Draw images on the screen
  - GUI interfaces
- Libraries talk to kernel to do all of the above
- Kernel interfaces to the hardware

Example of reading data from a file

Trap (syscall) is special instruction, tells CPU to change privilege level and run some code as the kernel



System Call Table

- How does the kernel know what we want it to do we say “kernel do stuff” / syscall?

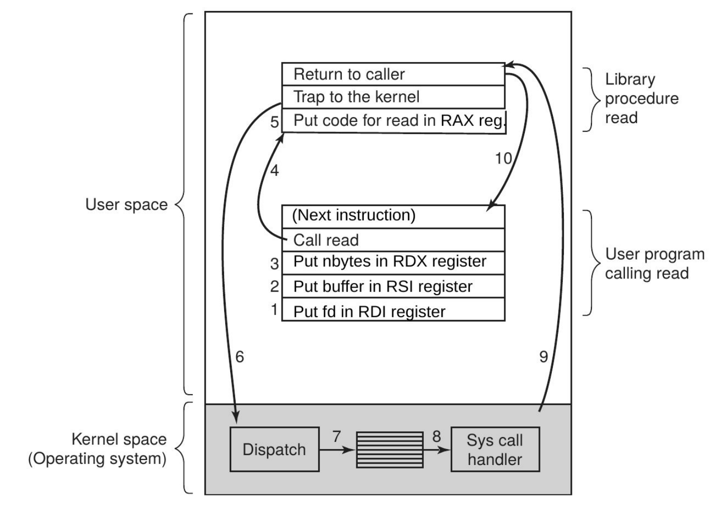
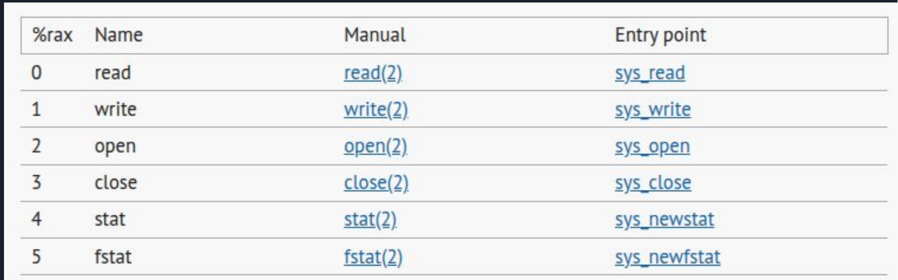


Figure 1-17. The 10 steps in making the system call read(fd, buffer, nbytes).

System Call Table

- How does the kernel know what we want it to do we say “kernel do stuff” / syscall?

- Pass arguments to kernel via CPU registers
  - RAX register is the sys call number



%rax	Name	Manual	Entry point
0	read	read(2)	sys_read
1	write	write(2)	sys_write
2	open	open(2)	sys_open
3	close	close(2)	sys_close
4	stat	stat(2)	sys_newstat
5	fstat	fstat(2)	sys_newfstat

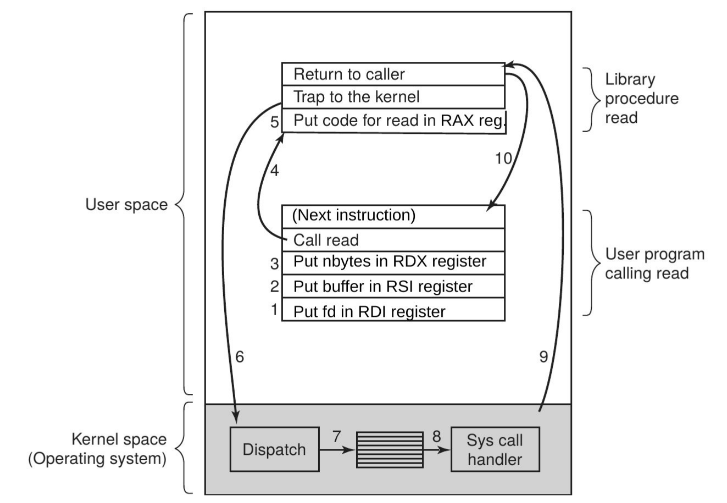
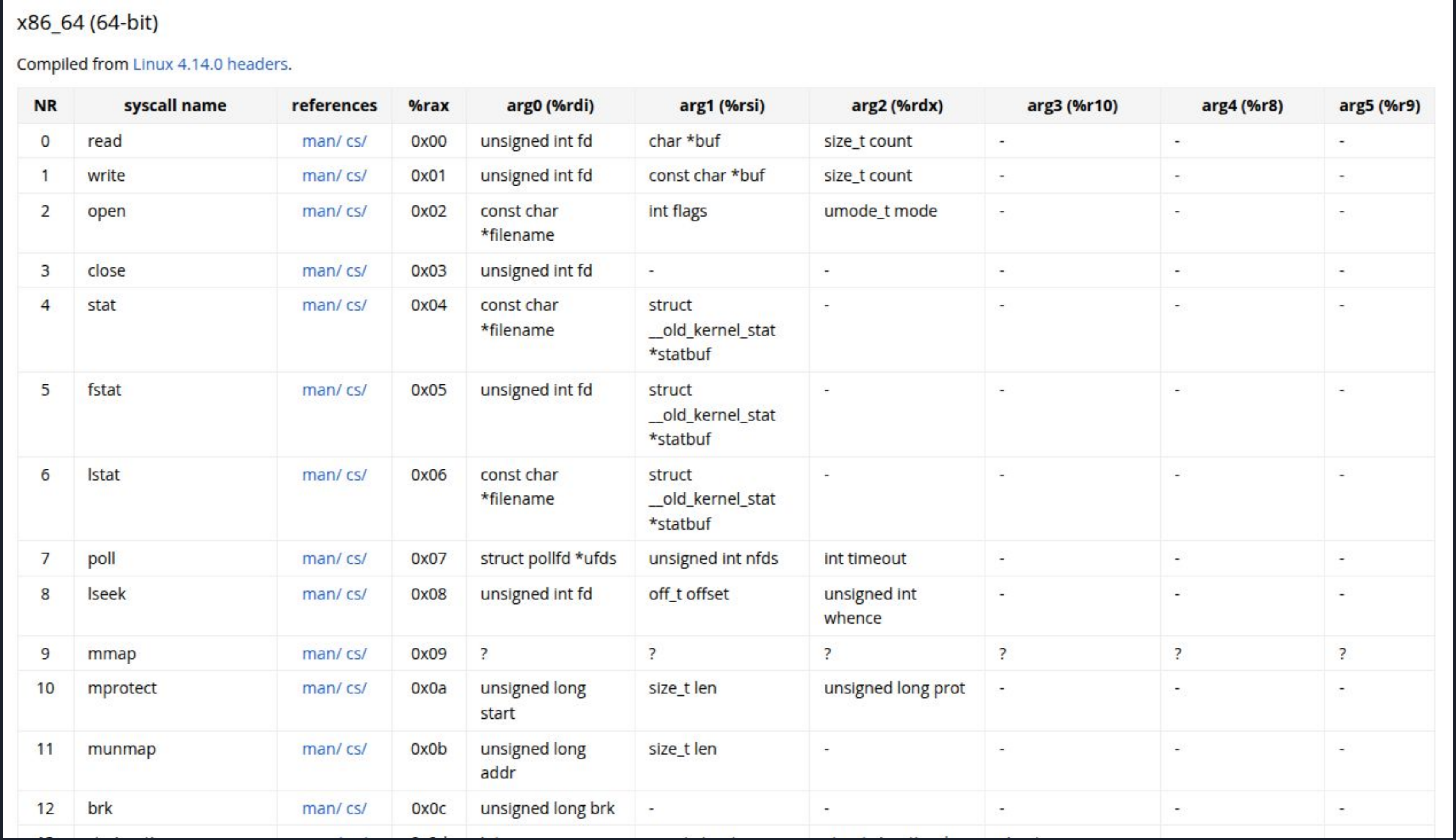


Figure 1-17. The 10 steps in making the system call read(fd, buffer, nbytes).

The syscall table is CPU arch specific. Many are similar, but each arch have differences



x86_64 (64-bit)

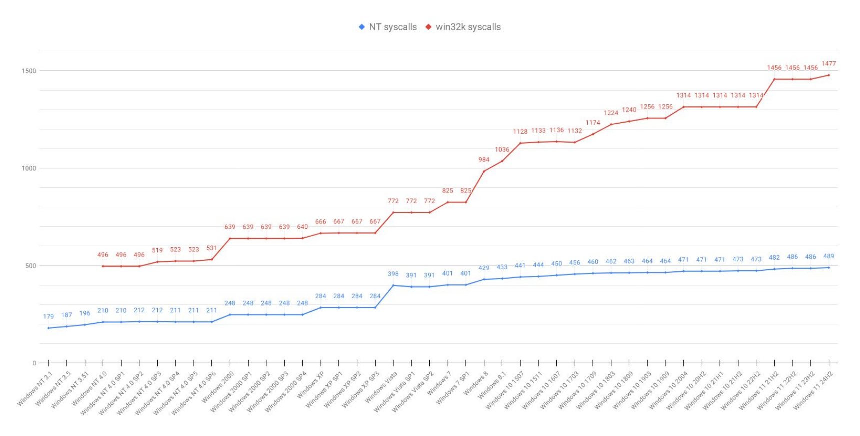
Compiled from Linux 4.14.0 headers.

NR	syscall name	references	%rax	arg0 (%rdi)	arg1 (%rsi)	arg2 (%rdx)	arg3 (%r10)	arg4 (%r8)	arg5 (%r9)
0	read	man/ cs/	0x00	unsigned int fd	char *buf	size_t count	-	-	-
1	write	man/ cs/	0x01	unsigned int fd	const char *buf	size_t count	-	-	-
2	open	man/ cs/	0x02	const char *filename	int flags	umode_t mode	-	-	-
3	close	man/ cs/	0x03	unsigned int fd	-	-	-	-	-
4	stat	man/ cs/	0x04	const char *filename	struct __old_kernel_stat *statbuf	-	-	-	-
5	fstat	man/ cs/	0x05	unsigned int fd	struct __old_kernel_stat *statbuf	-	-	-	-
6	lstat	man/ cs/	0x06	const char *filename	struct __old_kernel_stat *statbuf	-	-	-	-
7	poll	man/ cs/	0x07	struct pollfd *ufds	unsigned int nfds	int timeout	-	-	-
8	lseek	man/ cs/	0x08	unsigned int fd	off_t offset	unsigned int whence	-	-	-
9	mmap	man/ cs/	0x09	?	?	?	?	?	?
10	mprotect	man/ cs/	0x0a	unsigned long start	size_t len	unsigned long prot	-	-	-
11	munmap	man/ cs/	0x0b	unsigned long addr	size_t len	-	-	-	-
12	brk	man/ cs/	0x0c	unsigned long brk	-	-	-	-	-

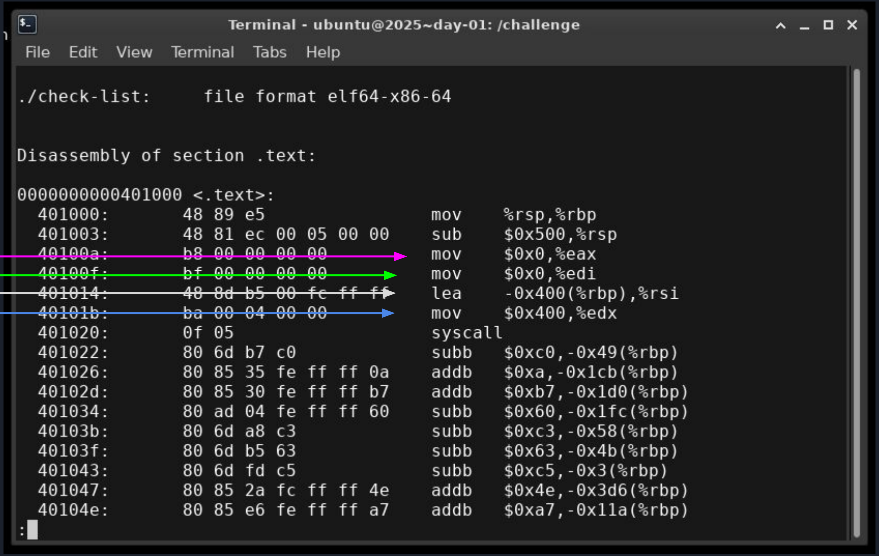
Linux vs Windows

- Linux
  - 300-400 syscalls
  - They rarely ever get changed

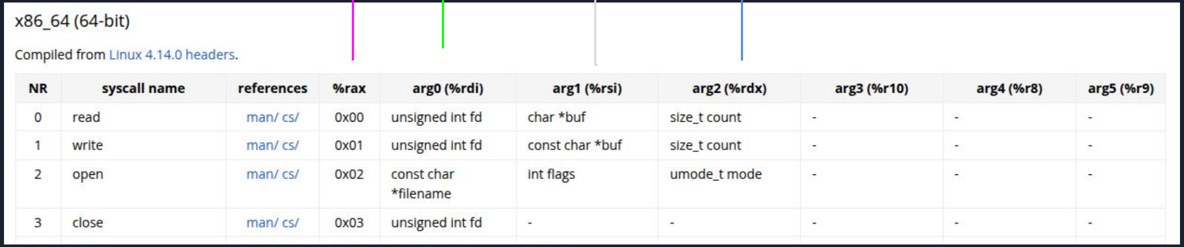
- WinNT kernel
  - 400+ syscalls
  - Change a lot / don’t use directly!
  - Closed source / not documented
- Win32 API / ntdll.dll
  - 1000s of functions
  - How developers should access the kernel



Back to Advent of Pwn...



```
./check-list:     file format elf64-x86-64


Disassembly of section .text:

0000000000401000 <.text>:
  401000:       48 89 e5                mov    %rsp,%rbp
  401003:       48 81 ec 00 05 00 00    sub    $0x500,%rsp
  40100a:       b8 00 00 00 00          mov    $0x0,%eax
  40100f:       bf 00 00 00 00          mov    $0x0,%edi
  401014:       48 8d b5 00 fc ff ff    lea    -0x400(%rbp),%rsi
  40101b:       ba 00 04 00 00          mov    $0x400,%edx
  401020:       0f 05                   syscall
  401022:       80 6d b7 c0             subb   $0xc0,-0x49(%rbp)
  401026:       80 85 35 fe ff ff 0a    addb   $0xa,-0x1cb(%rbp)
  40102d:       80 85 30 fe ff ff b7    addb   $0xb7,-0x1d0(%rbp)
  401034:       80 ad 04 fe ff ff 60    subb   $0x60,-0x1fc(%rbp)
  40103b:       80 6d a8 c3             subb   $0xc3,-0x58(%rbp)
  40103f:       80 6d b5 63             subb   $0x63,-0x4b(%rbp)
  401043:       80 6d fd c5             subb   $0xc5,-0x3(%rbp)
  401047:       80 85 2a fc ff ff 4e    addb   $0x4e,-0x3d6(%rbp)
  40104e:       80 85 e6 fe ff ff a7    addb   $0xa7,-0x11a(%rbp)
:
```



x86_64 (64-bit)

Compiled from Linux 4.14.0 headers.

NR	syscall name	references	%rax	arg0 (%rdi)	arg1 (%rsi)	arg2 (%rdx)	arg3 (%r10)	arg4 (%r8)	arg5 (%r9)
0	read	man/ cs/	0x00	unsigned int fd	char *buf	size_t count	-	-	-
1	write	man/ cs/	0x01	unsigned int fd	const char *buf	size_t count	-	-	-
2	open	man/ cs/	0x02	const char *filename	int flags	umode_t mode	-	-	-
3	close	man/ cs/	0x03	unsigned int fd	-	-	-	-	-

Links

- Tanenbaum and Bos, Modern Operating Systems (book)
- Hausenblas, Learning Modern Linux (book)
- https://filippo.io/linux-syscall-table/
- https://chromium.googlesource.com/chromiumos/docs/+/master/constants/syscalls.md
- https://j00ru.vexillium.org/syscalls/win32k/64/
-