Tutorial - 1

Introduction to syscalls

Kannav Mehta and Mayank Musaddi

What do Operating Systems do?

- In layman terms it can be thought to be providing a safe reliable bed for applications to start, run, and exit.
- It can also be thought of as the Operating system manages the hardware and provides ability for software to use the hardware.

Kernel Mode and User Mode

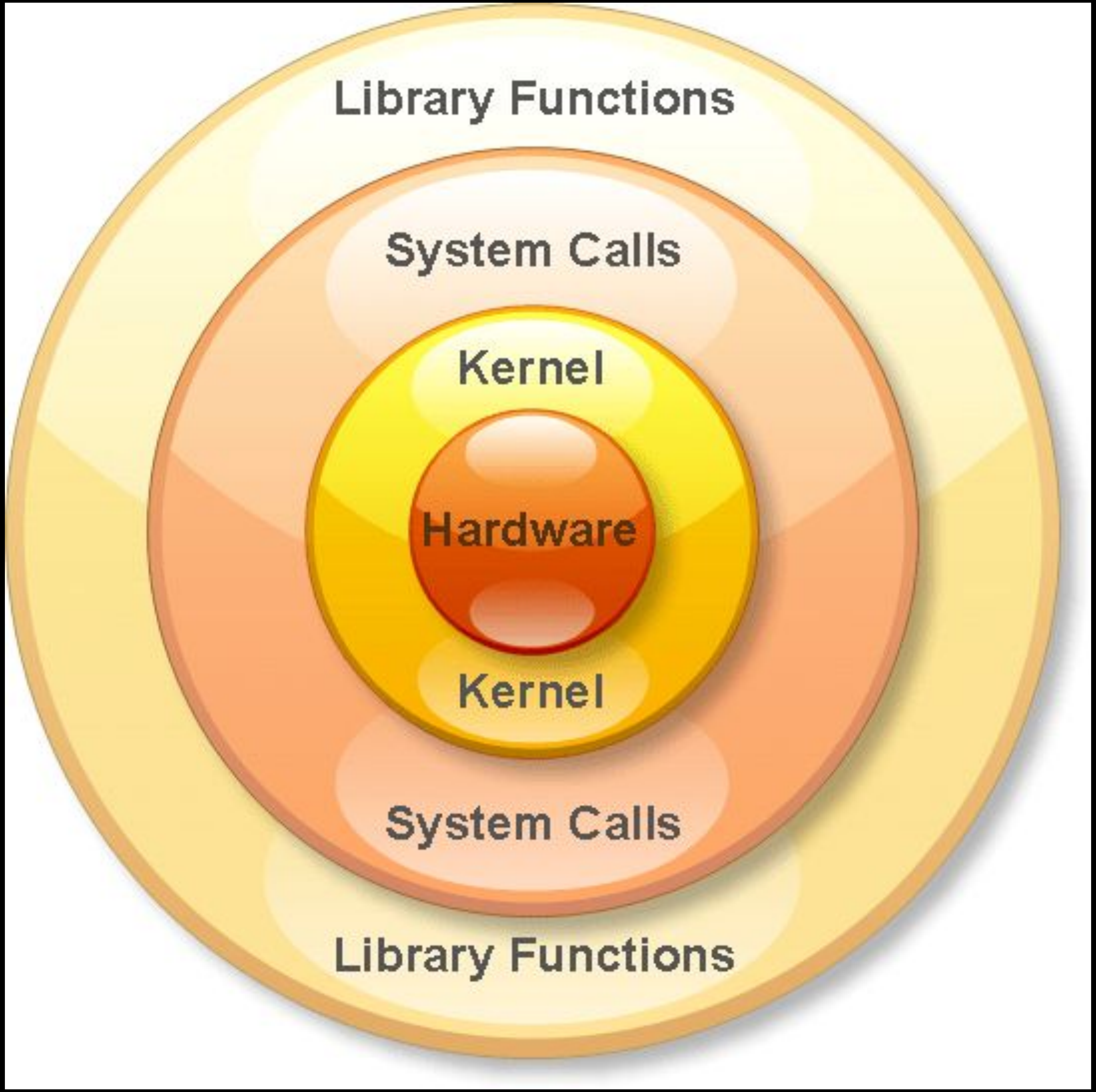
- A processor in a computer running an OS can be thought of having two different modes: **user mode** and **kernel mode**. The processor switches between the two modes depending on what type of code is running on the processor. Applications run in user mode, and core operating system components run in kernel mode.
- Initially on boot the processor starts in kernel mode, most ISA have special instructions that can only be run in kernel mode. Like setting some **hardware specific registers.** The kernel then runs user programs in user mode.
- This allows kernel control over the hardware but we have to allow user programs to use this hardware safely without disturbing other user programs.

Interrupts and ISR

- So user programs ask operating system to do things for them, how does the operating systems know some user program is asking stuff.
- The user program would call the operating system.
- But a call instruction won’t work as we have to switch modes, a user program can’t just go into kernel mode on a whim.
- We also cannot have the OS poll for events from time to time(Wasteful).
- We need to **interrupt** the CPU from executing user code and go to the kernel.
- We **need** hardware support.

- The way we do this is using hardware interrupts. You might have seem interrupts in arduino programming.
- Similar to arduinos, most CPUs have a INT pin that when sent a signal can be used to run routines in kernel mode.
- For legacy intel architectures this translates to the INT 0x80 instruction
- syscall instruction is the default way of entering kernel mode on x86-64.
- The kernel sets the code to be run during these routines when it first boots up, These are called the **interrupt handlers**.
- You can also pass arguments to interrupts via registers.

Syscalls


Library Functions
System Calls
Kernel
Hardware
Kernel
System Calls
Library Functions

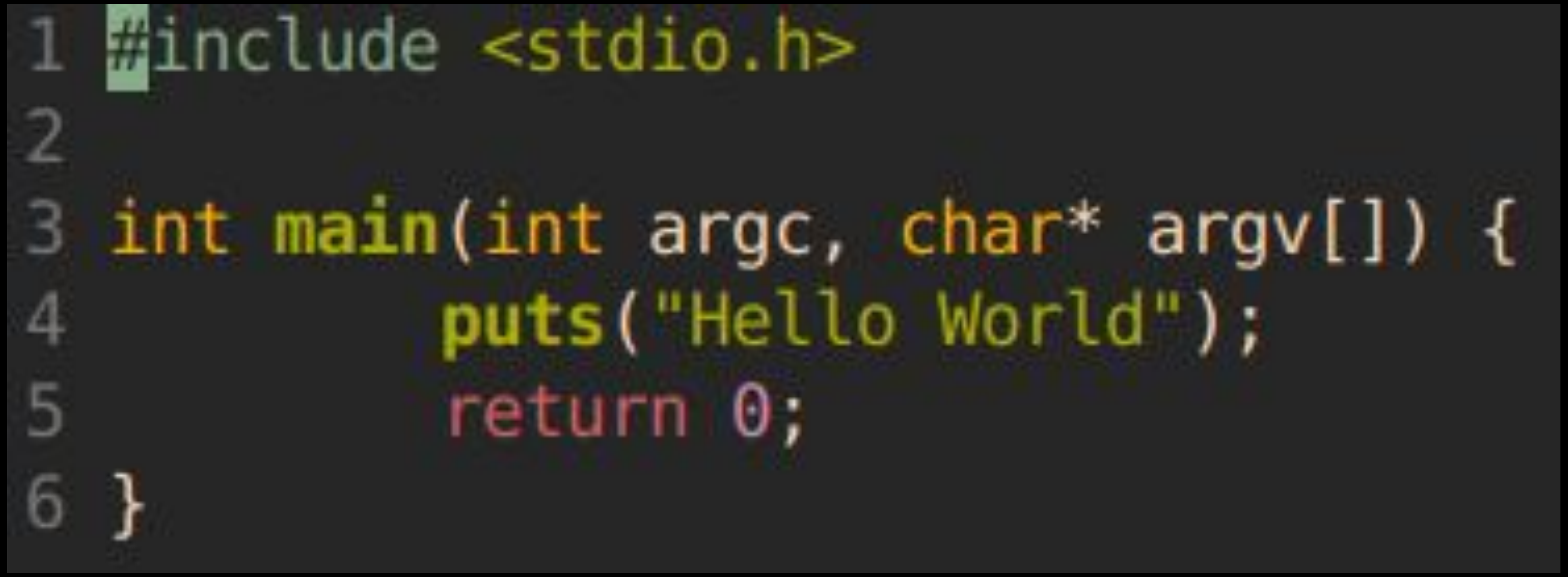
- Syscalls are the mechanism used modern operating systems to do this.
- That means the code that gets executed during syscalls is part of the **kernel**.
- In modern systems you will never do syscalls directly.
- In case of C this is through **glibc(The C standard library)**.

Types of System Calls : There are many different categories of system calls –

1. **Process control:** end, abort, create, terminate, allocate and free memory.
2. **File management:** create, open, close, delete, read file etc.
3. **Device management**
4. I**nformation maintenance**
5. **Communication**

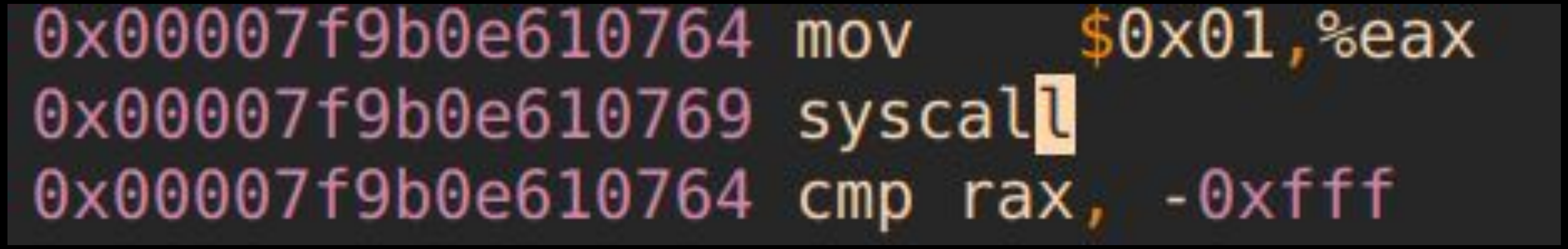
Example of syscall `write()`

A dive into linux kernel



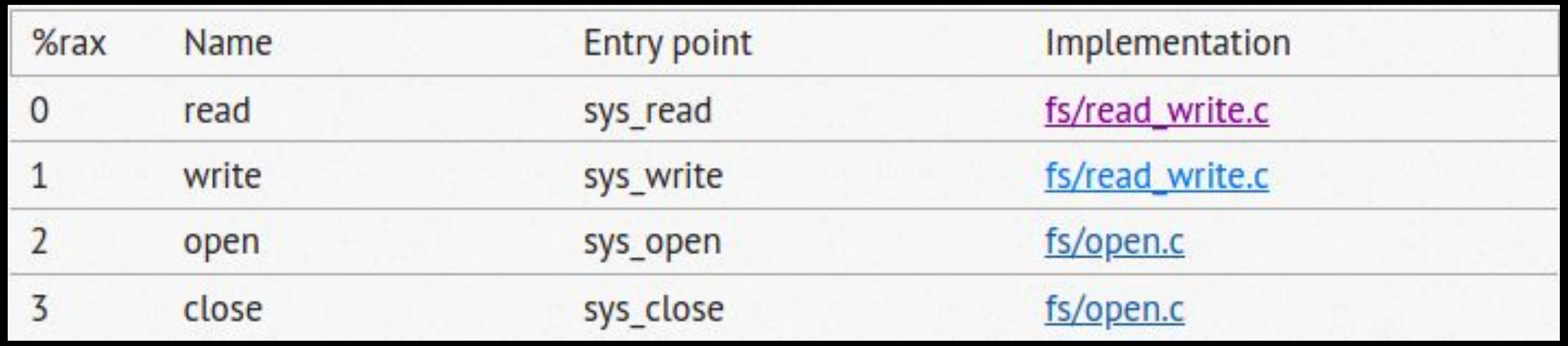
```c
#include <stdio.h>

int main(int argc, char* argv[]) {
        puts("Hello World");
        return 0;
}
```

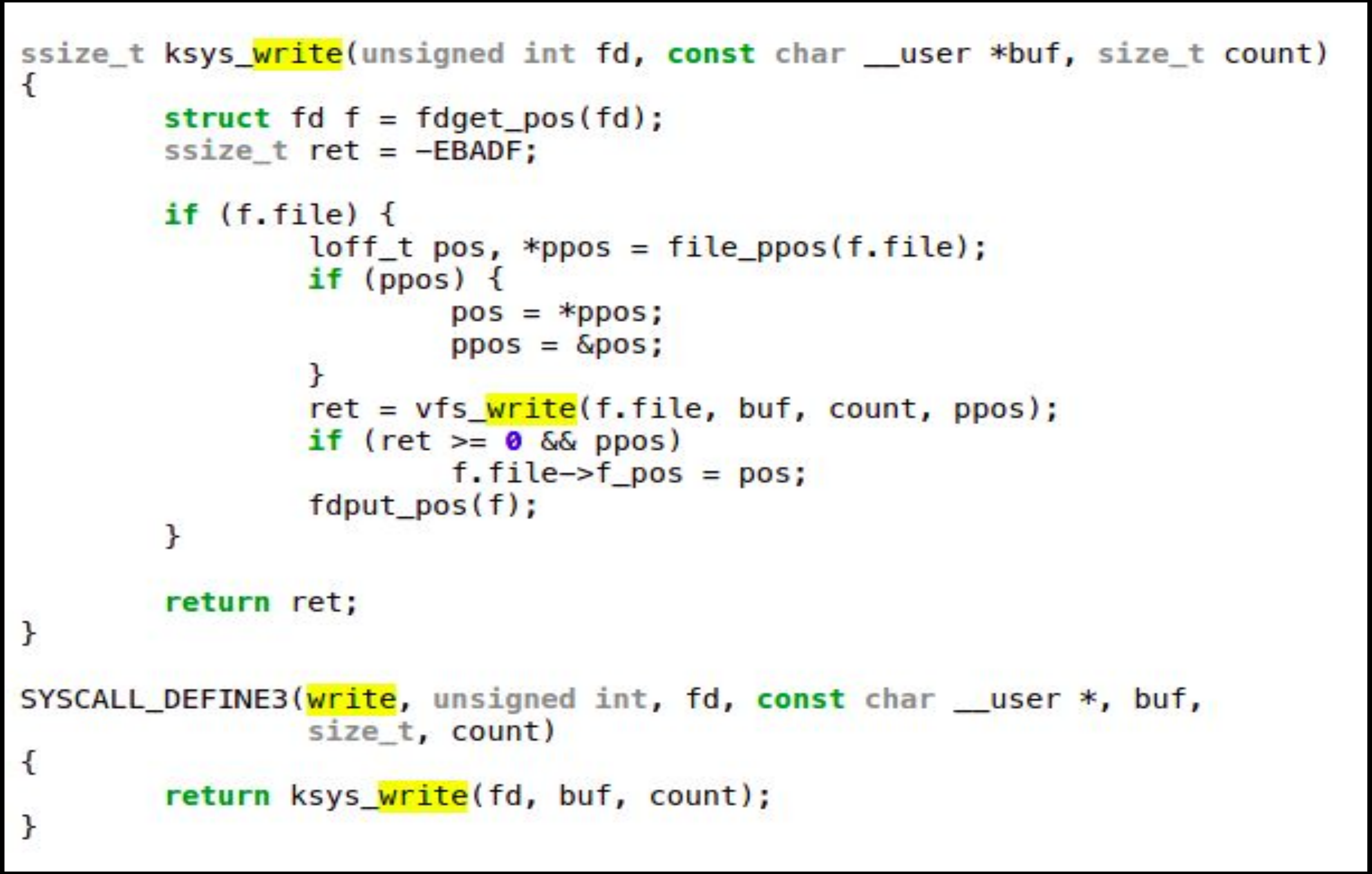


```
0x00007f9b0e610764 mov    $0x01,%eax
0x00007f9b0e610769 syscall
0x00007f9b0e610764 cmp rax, -0xfff
```

src: https://www.youtube.com/watch?v=fLS99zJDHOc



%rax	Name	Entry point	Implementation
0	read	sys_read	fs/read_write.c
1	write	sys_write	fs/read_write.c
2	open	sys_open	fs/open.c
3	close	sys_close	fs/open.c



```
ssize_t ksys_write(unsigned int fd, const char __user *buf, size_t count)
{
        struct fd f = fdget_pos(fd);
        ssize_t ret = -EBADF;

        if (f.file) {
                loff_t pos, *ppos = file_ppos(f.file);
                if (ppos) {
                        pos = *ppos;
                        ppos = &pos;
                }
                ret = vfs_write(f.file, buf, count, ppos);
                if (ret >= 0 && ppos)
                        f.file->f_pos = pos;
                fdput_pos(f);
        }

        return ret;
}

SYSCALL_DEFINE3(write, unsigned int, fd, const char __user *, buf,
                size_t, count)
{
        return ksys_write(fd, buf, count);
}
```

src: https://www.youtube.com/watch?v=fLS99zJDHOc

Code Demo