# ASSIGNMENT ON TRACING SYSTEM CALLS IN XV6

Q1

Part One: **System call tracing**

Your first task is to modify the xv6 kernel to print out a line for each system call invocation. It is enough to print
the name of the system call and the return value; you don't need to print the system call arguments.

When you're done, you should see output like this when booting xv6:

```
...
fork -> 2
exec -> 0
open -> 3
close -> 0
$write -> 1
write -> 1
```

That's init forking and executing sh, sh making sure only two file descriptors are open, and sh writing the $ prompt.

(Note: the output of the shell and the system call trace are intermixed, because the shell uses the write syscall to
print its output.)

Hint: modify the syscall() function in syscall.c

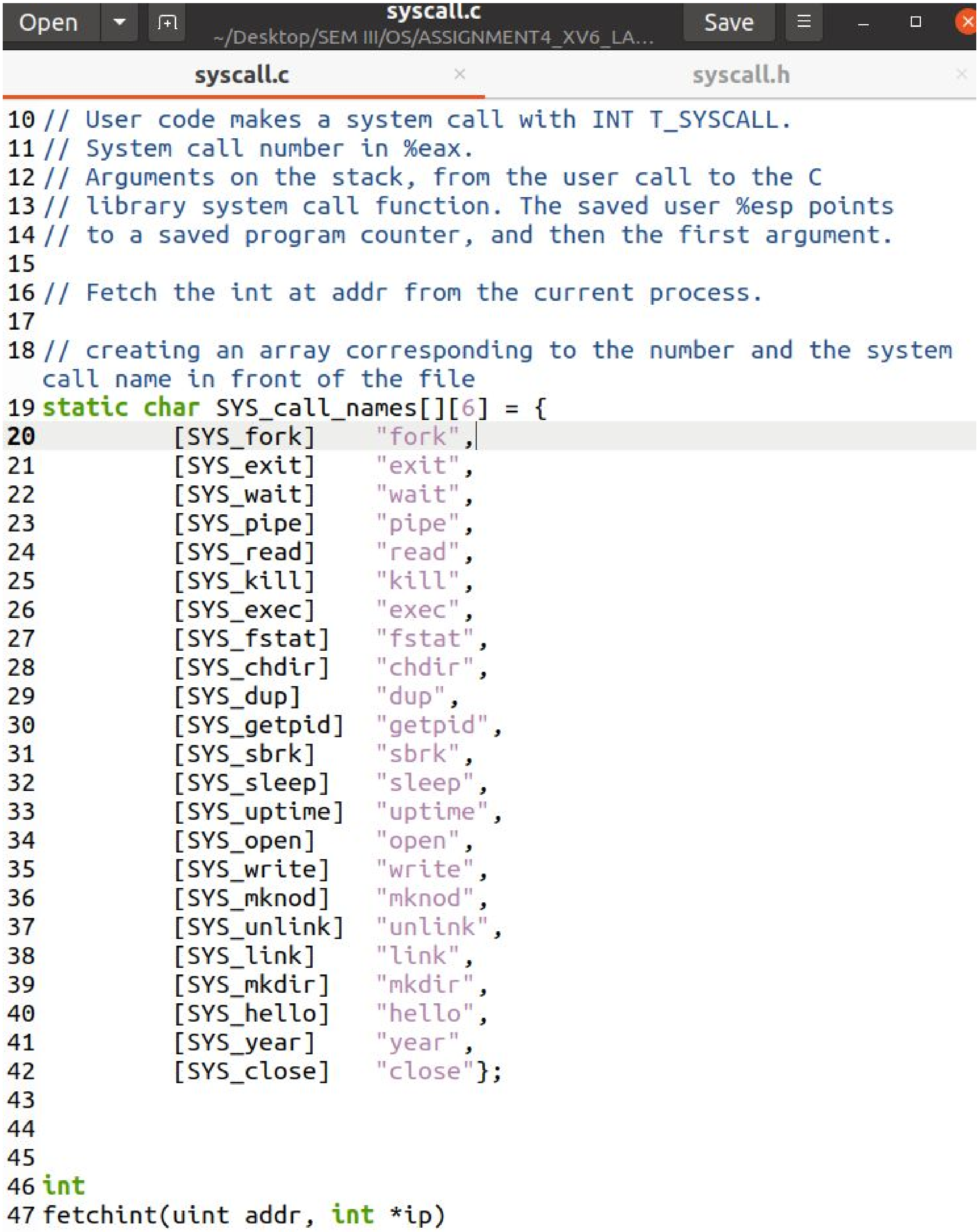
STEPS:

In file syscall.h,

the name of the system call and the corresponding serial number. What we want to display on the terminal is the corresponding name and number.

- **We need to modify the file , hence to do that open syscall.c and add an array corresponding to the number and the system call name in front of the file .**

- **Then in the syscall.c file , we need to edit the code to the following attached**



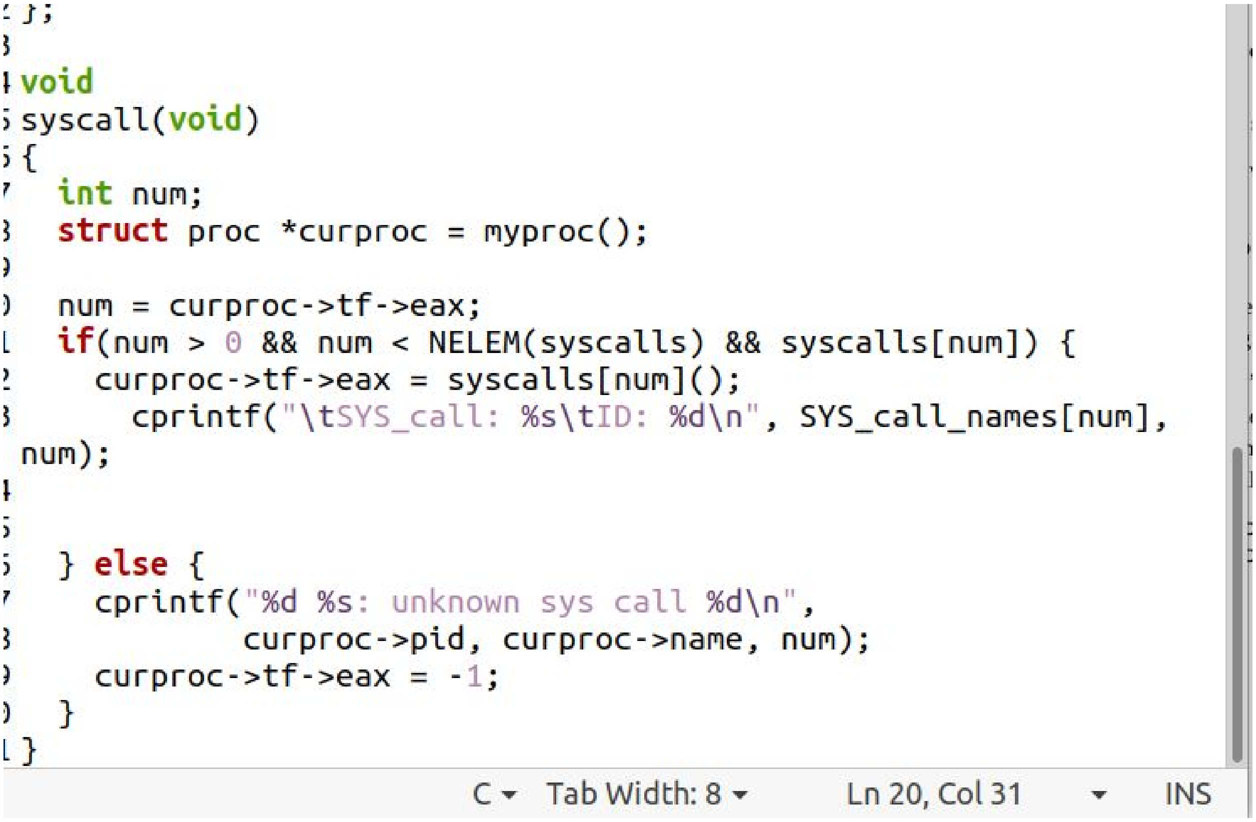
```
// User code makes a system call with INT T_SYSCALL.
// System call number in %eax.
// Arguments on the stack, from the user call to the C
// library system call function. The saved user %esp points
// to a saved program counter, and then the first argument.

// Fetch the int at addr from the current process.

// creating an array corresponding to the number and the system call name in front of the file
static char SYS_call_names[][6] = {
         [SYS_fork]    "fork",
         [SYS_exit]    "exit",
         [SYS_wait]    "wait",
         [SYS_pipe]    "pipe",
         [SYS_read]    "read",
         [SYS_kill]    "kill",
         [SYS_exec]    "exec",
         [SYS_fstat]   "fstat",
         [SYS_chdir]   "chdir",
         [SYS_dup]     "dup",
         [SYS_getpid]  "getpid",
         [SYS_sbrk]    "sbrk",
         [SYS_sleep]   "sleep",
         [SYS_uptime]  "uptime",
         [SYS_open]    "open",
         [SYS_write]   "write",
         [SYS_mknod]   "mknod",
         [SYS_unlink]  "unlink",
         [SYS_link]    "link",
         [SYS_mkdir]   "mkdir",
         [SYS_hello]   "hello",
         [SYS_year]    "year",
         [SYS_close]   "close"};



int
fetchint(uint addr, int *ip)
```



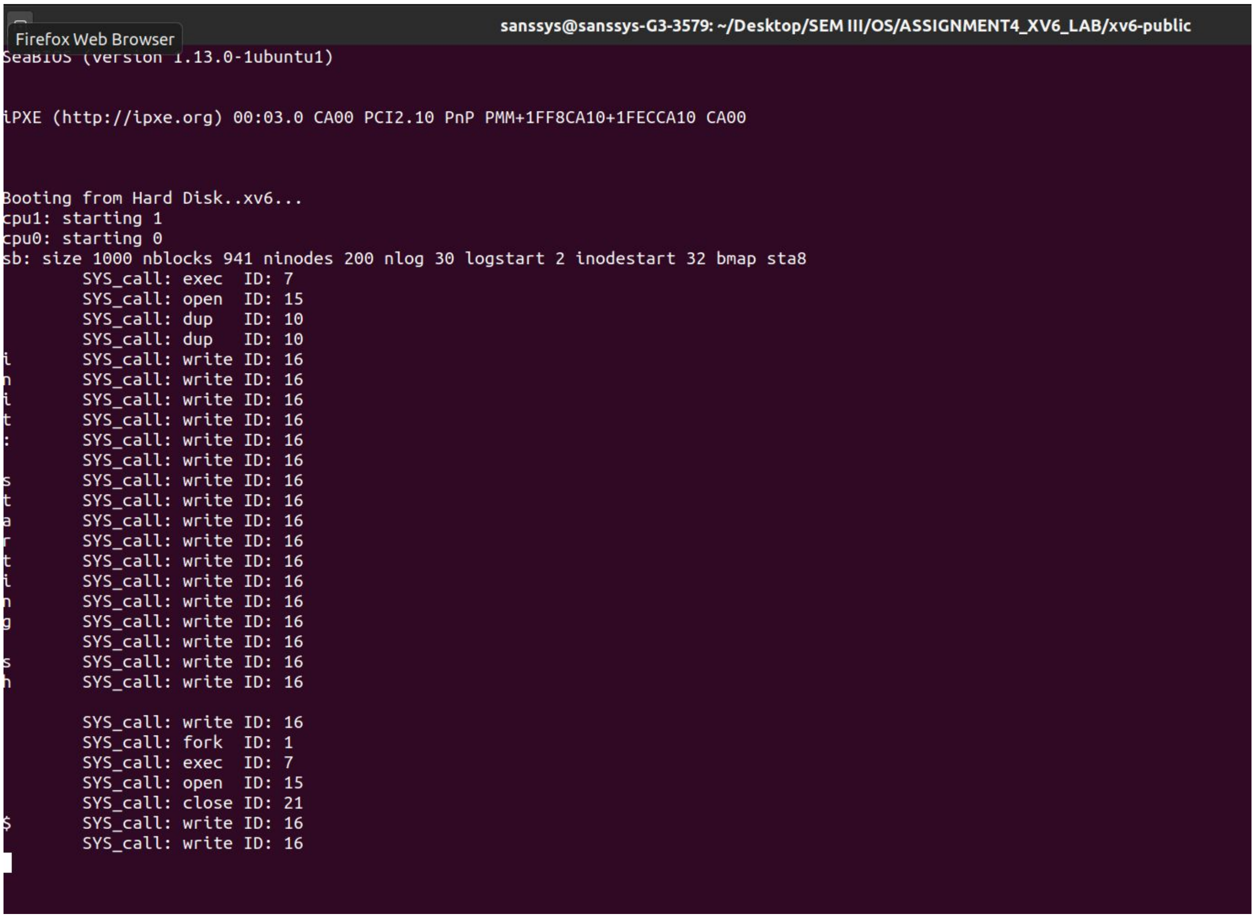
```
};

void
syscall(void)
{
  int num;
  struct proc *curproc = myproc();

  num = curproc->tf->eax;
  if(num > 0 && num < NELEM(syscalls) && syscalls[num]) {
    curproc->tf->eax = syscalls[num]();
      cprintf("\tSYS_call: %s\tID: %d\n", SYS_call_names[num],
num);


  } else {
    cprintf("%d %s: unknown sys call %d\n",
            curproc->pid, curproc->name, num);
    curproc->tf->eax = -1;
  }
}
```

- **Then using make qemu-nox command , and then we finally get this output**


Firefox Web Browser
sanssys@sanssys-G3-3579: ~/Desktop/SEM III/OS/ASSIGNMENT4_XV6_LAB/xv6-public

```
SeaBIOS (version 1.13.0-1ubuntu1)


iPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8CA10+1FECCA10 CA00



Booting from Hard Disk..xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap sta8
        SYS_call: exec  ID: 7
        SYS_call: open  ID: 15
        SYS_call: dup   ID: 10
        SYS_call: dup   ID: 10
i       SYS_call: write ID: 16
n       SYS_call: write ID: 16
i       SYS_call: write ID: 16
t       SYS_call: write ID: 16
:       SYS_call: write ID: 16
        SYS_call: write ID: 16
s       SYS_call: write ID: 16
t       SYS_call: write ID: 16
a       SYS_call: write ID: 16
r       SYS_call: write ID: 16
t       SYS_call: write ID: 16
i       SYS_call: write ID: 16
n       SYS_call: write ID: 16
g       SYS_call: write ID: 16
        SYS_call: write ID: 16
s       SYS_call: write ID: 16
h       SYS_call: write ID: 16

        SYS_call: write ID: 16
        SYS_call: fork  ID: 1
        SYS_call: exec  ID: 7
        SYS_call: open  ID: 15
        SYS_call: close ID: 21
$       SYS_call: write ID: 16
        SYS_call: write ID: 16
```