**NAME:** Dev Sandip Shah

**ROLL NO:** 200123074

**Exercise 1. Create a system call int sys\_draw(void \*buf, uint size), which copies an ASCII art image (Use a buffer of any picture, google it for more information) of a picture to a user-supplied buffer, provided that the buffer is large enough. You are welcome to use an ASCII art generator, or draw your own by hand. (One such example is given with the assignment) If the buffer is too small, or not valid, return a negative value. If the call succeeds, return the number of bytes copied. You may find it helpful to review how other system calls are implemented and compiled into the kernel, such as read.**

**Answer:**

An operating system supports two modes; the kernel mode and the user mode. When a program in user mode requires access to RAM or a hardware resource, it must ask the kernel to provide access to that particular resource. This is done via a system call. When a program makes a system call, the mode is switched from user mode to kernel mode. In order to define our own system, call in xv6, changes need to be made to 5 files. Namely, these files are as follows.

1. syscall.h
2. syscall.c
3. sysproc.c
4. usys.S
5. user.h

We would start the procedure by editing syscall.h in which a number is given to every system call. This file already contains 21 system calls. In order to add the custom system call, the following line needs to be added to this file.



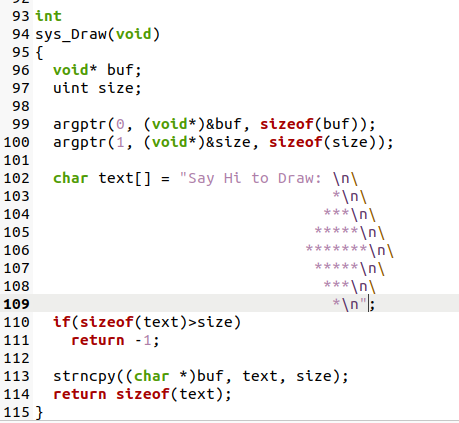
Next, we need to add a pointer to the system call in the syscall.c file. This file contains an array of function pointers which uses the above-defined numbers (indexes) as pointers to system calls which are defined in a different location. In order to add our custom system call, add the following line to this file.



When the system call with number 22 is called by a user program, the function pointer sys\_wolfie which has the index SYS\_wolfie or 22 will call the system call function. Hence, our next objective is to implement a system call function. But we do not implement the system call function in syscall.c. We only add a prototype as shown below.



We implement the system call function in sysproc.c.



Now, to add an interface for a user program to call the system call, we add



to usys.S and

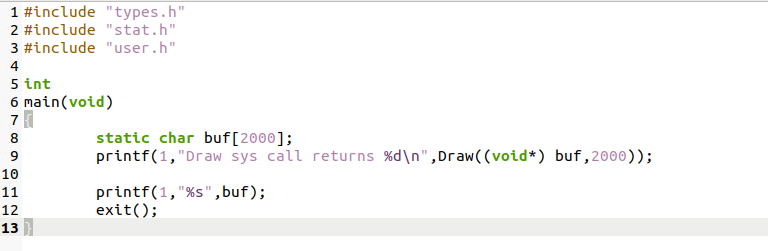


to user.h

**Exercise 2. Write a user-level application, called Drawtest.c, that gets the image from the kernel, and prints it to the console. When the OS runs, your program's binary should be included in fs.img and listed if someone runs ls at the xv6 shell's command prompt. Study Makefile to figure out how to compile a usermode program and add it to fs.img**

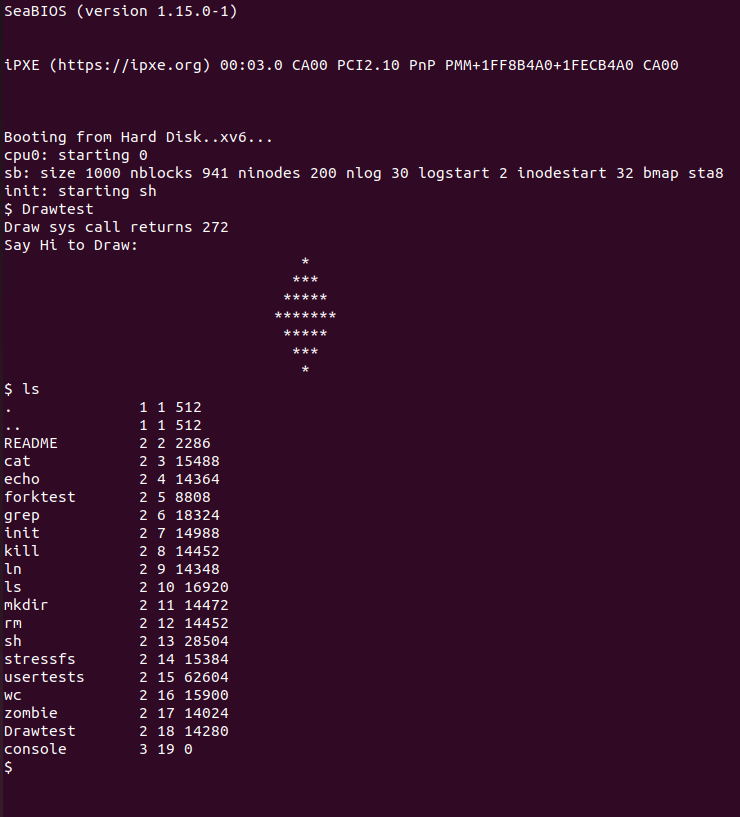
**Answer:**

Now the only task left is to add a user program to call the system call that we just made above. For this, I made a file Drawtest.c inside xv6 folder and wrote the following code in it.

****

After this, I added this file to the Makefile under UPROGS and EXTRA.

Then I used “make clean”, “make”, “make qemu-nox” and then entered Drawtest to get the following output. Also, my program is also listed when I use the ls command. Screenshot is attached below

****