Report - OS Lab 4 - 150050031

How to compile xv6

- Using the given Makefile, we can just run `make` to compile everything. `make qemu` runs the OS in qemu and we can interact with it.
- Inside of Makefile:
  - The OS consists of bootblock, the kernel and the filesystem
  - For the bootblock, the file bootblock.c is compiled into a binary and signed.
  - For the kernel, we have various files (.c counterpart of files listed under OBJS), these contain all the syscalls and other kernel functionality. These are linked with the initcode and entryother binaries.
  - The filesystem contains user level programs and files. All the user level binaries are compiled and then the mkfs programs makes an image of the filesystem.
- All the above becomes a part of the xv6 image which is loaded into qemu.

How to implement syscalls in xv6

- The function exposed to the user is in `user.h` (say `my_syscall`).
- `usys.S` tells the OS which function calls to make into a system call, so this has to be the same as the function exposed to the user (Add `SYSCALL(my_syscall)` to `usys.S`)
- When compiling, the OS looks in `syscall.h` for getting the syscall number for it. It looks for the value of `SYS_my_syscall` which is `#defined`. The `SYS` appended is probably for convention.
- In `syscall.c`, this number is mapped to a function which will be used to extract arguments to the syscall from the stack and perform checks on them. We can call this function anything (say `getargs_my_syscall`). If we followed convention, we would've named it `sys_my_syscall`.
- We'll define the function in a separate file `sysfile.c`, so we'll `extern` the function `getargs_my_syscall`. Note that since we don't know the arguments yet, this function will take no arguments. Also the return type is int, usually -1 is returned on error. The linker links the object files of `sysfile.c` and `syscall.c` (We tell it to do that in the `Makefile`), so everything works out.
- `getargs_my_syscall` will get the arguments and call the implementation of this syscall (say `implement_my_syscall`). Again we could've named it anything! This function will take the same arguments we gave `my_syscall`.

- The function definition of implement_my_syscall is to be placed in defs.h so that sysfile.c can find it.
- Implement implement_my_syscall in a file(say my_syscall.c)
- Make proper changes to Makefile for proper compilation.