Operating Systems-Fall2024
Homework 1
Submission: 11:55pm 15th September, 2024

Q1. Write an xv6 runnable program uses xv6 system calls to “ping-pong” a byte between two processes over a pair of pipes, one for each direction.

Q2. Write an xv6 runnable program that counts the number of times system calls are invoked in another program.

Q3. (a) What is the benefit of pipes over other file redirection method. Read the book.

(b) The following code implements

/bin/ls -al / | /usr/bin/tr a-z A-Z

The first command generates a long-format directory listing of the root (/) directory and the second command takes that listing and translates all lowercase characters to uppercase.

(c) Try to implement this code in xv6. Give reasons why it cannot be done

```
#include <stdlib.h>
#include <stdio.h>

void runpipe();

int
main(int argc, char **argv)
{
	int pid, status;
	int fd[2];

	pipe(fd);

	switch (pid = fork()) {

	case 0:
		runpipe(fd);
```

```
                exit(0);

        default:
                while ((pid = wait(&status)) != -1)
                fprintf(stderr, "process %d exits with %d\n", pid, WEXITSTATUS(status));
                break;

        case -1:
                perror("fork");
                exit(1);
        }
        exit(0);
}

char *cmd1[] = { "/bin/ls", "-al", "/", 0 };
char *cmd2[] = { "/usr/bin/tr", "a-z", "A-Z", 0 };

void
runpipe(int pfd[])
{
        int pid;

        switch (pid = fork()) {

        case 0:
                dup2(pfd[0], 0);
                close(pfd[1]);
                execvp(cmd2[0], cmd2);
                perror(cmd2[0]);

        default:
                dup2(pfd[1], 1);
                close(pfd[0]);
                execvp(cmd1[0], cmd1);
                perror(cmd1[0]);

        case -1:
                perror("fork");
                exit(1);
        }
}
```