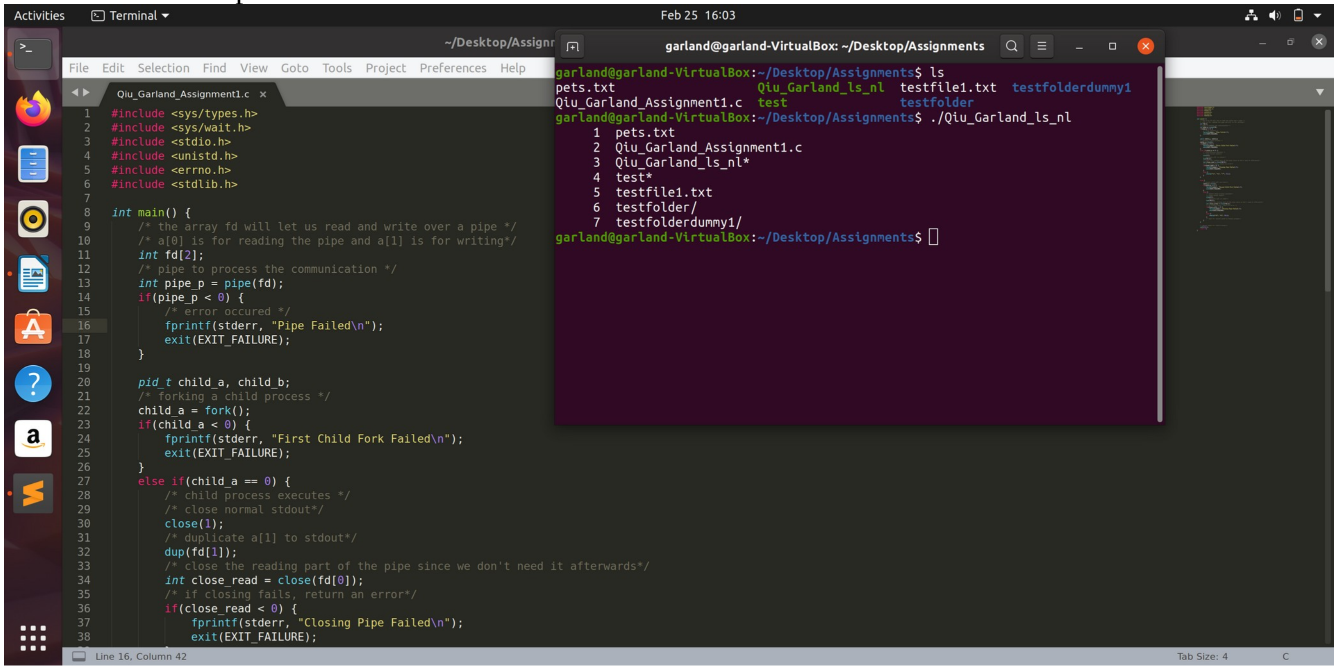
Garland Qiu
Assignment 1
Operating Systems Section T

Screenshot of Output



```c
#include <sys/types.h>
#include <sys/wait.h>
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <stdlib.h>

int main() {
    /* the array fd will let us read and write over a pipe */
    /* a[0] is for reading the pipe and a[1] is for writing*/
    int fd[2];
    /* pipe to process the communication */
    int pipe_p = pipe(fd);
    if(pipe_p < 0) {
        /* error occured */
        fprintf(stderr, "Pipe Failed\n");
        exit(EXIT_FAILURE);
    }

    pid_t child_a, child_b;
    /* forking a child process */
    child_a = fork();
    if(child_a < 0) {
        fprintf(stderr, "First Child Fork Failed\n");
        exit(EXIT_FAILURE);
    }
    else if(child_a == 0) {
        /* child process executes */
        /* close normal stdout*/
        close(1);
        /* duplicate a[1] to stdout*/
        dup(fd[1]);
        /* close the reading part of the pipe since we don't need it afterwards*/
        int close_read = close(fd[0]);
        /* if closing fails, return an error*/
        if(close_read < 0) {
            fprintf(stderr, "Closing Pipe Failed\n");
            exit(EXIT_FAILURE);
```

```
garland@garland-VirtualBox:~/Desktop/Assignments$ ls
pets.txt                   Qiu_Garland_ls_nl  testfile1.txt  testfolderdummy1
Qiu_Garland_Assignment1.c  test               testfolder
garland@garland-VirtualBox:~/Desktop/Assignments$ ./Qiu_Garland_ls_nl
     1  pets.txt
     2  Qiu_Garland_Assignment1.c
     3  Qiu_Garland_ls_nl*
     4  test*
     5  testfile1.txt
     6  testfolder/
     7  testfolderdummy1/
garland@garland-VirtualBox:~/Desktop/Assignments$ 
```

Questions:

1. What form of exec() did you use? Why?

I used execlp() because instead of having to specify the full path “/bin/ls”, I can simply let the function find it in my PATH variables and run it, and I would not need the full path to run the system call. Linux generally has the functions “ls” and “nl” built in to the PATH variables, so it feels unnecessary to locate them again.

2. How many times you used fork()? Why?

I used fork() twice because we have two child processes in this case: “ls -F” and “nl”. I first fork “ls -F” before “nl” because “nl” requires an input, which will be our output from “ls -F”.

3. How many pipes this assignment requires? Why?

We only need one pipe for this task because we only have two processes to work with. The system call “ls -F” reads all the files in the directory and write all of it to our system call “nl”.

4. What form of wait() you used? How many times?

I used wait(NULL) one time only because wait(NULL) will wait for one child to finish. In my code, wait(NULL) comes into effect after “nl” fork()’s value becomes 1.