Wei Fei (2538810)
EECS678 Lab3 Report
Prasanth Vivekanandan
02/15/15

1. Briefly describe the design of the program. How many processes and how many pipes are created with your solution? How does data flow from process to process?

The program needs us to write a program named “finder.c” that produces the same output as the shell file. Also, the program is a kind of skeleton for a four stage pipeline. So we have three pipes to connect four children processes.

The data flow is from process to process smartly. For example, when “pid_1 == 0”, the data is accepted by the command line, and the first children process will write data through “pip1”. Secondly, when “pid_2 == 0”, the second children process will write the data that is read by the first children. In other word, the data is passed from the read end of a pipe to write end of the pipe. Similarly, the third children and fourth children will also read and write data through pipelines.

2. Did you have any problems writing or debugging this program? How did you test and debug your solution?

Yes. I have one problem is that when I was doing “dup2” function, I put it after “execl” function originally. The output showed me a lot of pipeline information. When I used the debugger to find the problem, I move the “dup2” function prior of the “execl” function. The output turned to correct.

Meanwhile, when I was debugging, the debugger showed me the line number of some possible errors. Ans the output gave me an “0” for the error. I was thinking the process might enter the if statement and run the error function. So I commented each functions one by one, and find the error finally. I think the problem is because we need to create pipe firstly, and then pass data through pipe. If we do not have pipes, we may not have any useful tools to pass data.

3. When he was head of Bell Labs, Doug McIlroy summarized the "Unix philosophy" as follows. How do pipes contribute to the Unix philosophy? Also, do you think this is a "good" philosophy for software engineering? Why or why not?

By using pipes in the unix or linux program, not only we can write programs that do one thing and do it well, but also we can write programs to handle text streams and let each program work together. The pipes are kind of universal interfaces.

I think this is a good philosophy for software engineering. For example, if two process needs to work together and work at the same time, we could pass the command from the previous process to the next process by using pipes. When we use “fork()” function to create or invoke a lot of processes, pipe file descriptor can be used by all these processes. It is conveniently to connect them and pass data between them.