

91.308 Operating Systems Proj #2 February 8, 2012

1. The **due date** for this assignment is **Friday, Feb 24**.
2. **All** of your submissions must include a minimum of **four** separate files:
 - **File 1:** A short **write-up** that first specifies what you think your **degree of success** with a project is (**from 0% to 100%**), followed by a brief discussion of your approach to the project along with a **detailed description** of any problems that you were **not** able to resolve for this project. **Failure to specifically provide this information will result in a 0 grade** on your assignment. If you do **not disclose** problems in your write-up and problems are detected when your program is tested, you will receive a grade of 0.
 - **File(s) 2(a, b, c, ...):** Your **complete source code**, in one or more **.c** and/or **.h** files
 - **File 3:** A **make file** to build your assignment. This file must be named **Makefile**.
 - **File 4:** A file that includes your **resulting output** run(s) from your project. This is a simple text file that shows your output, but make sure that you annotate it so that it is self descriptive and that all detailed output is well identified.
3. The files described above should be the **only files** placed in one of your **subdirectories**, and this subdirectory should be the **target of your submit command**. From a shell prompt use the submit command as follows:

```
$ submit csong 308hwN assignN_dir
```

where **308hwN** is the name of the assignment, and **assignN_dir** is the name of the directory with exactly your required submission files in it for assignment number **N** (i.e. **submit csong 308hw2 assign2_dir** would be the correct submit command for this assignment).

Chunyao Song (chunyao.song@qq.com) is the TA and assignment corrector for this course, and she will email you your grade and any details about points you may have lost on each assignment. **It is very important to make sure that you include your email address in the write-up portion of your submission, so Chunyao can return your grade to you.**

4. This problem will require you to write two different, but similar, programs that each spawn a child, let the child process load itself with a Linux/UNIX type command (the `grep` and `sort` commands will be used in this assignment), and then send data to and retrieve data from the child process. Each of your programs will need to:

- Create 2 nameless pipes
 - Create a child process
 - Perform the appropriate channel management with the processes' standard IO channels and the pipe channels to create a circular connection through the UNIX command
 - standard input to the command child will be sent as output from the parent
 - input to the parent will be sent by the child command as standard output
 - the parent will use the appropriate pipe channels to talk to the child
 - The parent will be responsible for managing any data files that the child command will need sent via the pipe
 - The parent will collect all output from the child command sent through the pipe, process it as required and write required results to the standard output
 - The parent will terminate with success when the pipe being read from the child command shows EOF
5. You must implement your programs to work with the `sort` and `grep` commands as follows:
- For the `sort` command: from your parent process, open the file `~bill/cs308/cs308a2_sort_data` for reading
 - Read each line of the file and write it into (the parent's) OUT pipe to the child process that has exec'd `sort`
 - Read the file back from the child running `sort` from the (parent's) IN pipe, and write the required results to stdout (or a file if you choose)
 - `sort` must sort the data sent by the parent **first by phone number area code** and then **by last name within each area code**
 - Your required output will be a summary of the sorted results returned by the child running `sort` that consists of a **listing of each unique area code and the number of people found in the data sent to sort for that code**
 - For the `grep` command: from your parent process, open the file `~bill/cs308/cs308a2_grep_data_1` for reading
 - Read each line of the file and write it into (the parent's) OUT pipe to a child process that exec'd `grep`
 - The child process running `grep` must filter its incoming lines, looking for any line that has the string `"123"` in it
 - The child process running `grep` must write back each matching line to the parent process by writing it into the (parent's) IN pipe
 - You (the parent) must read back `grep's` output from your IN pipe and print out a **single number** indicating how many lines had the matching string as your report. You **must count the lines**

returned by grep in your parent process (**do not have grep count the lines**)

- You must now repeat this **grep** exercise using the file **~bill/cs308/cs308a2_grep_data_2**
 - Report the behavior of your program using **~bill/cs308/cs308a2_grep_data_2**, and **how it differed** from your run with **~bill/cs308/cs308a2_grep_data_1**. **Explain what you observed.**