CS3423 Pgm#8 Process Control (40 pts)

This assignment involves creating a parallel shell tool called PELL. It provides users of pell with the ability to execute commands, execute multistep processes connected by pipes, and execute concurrent independent processes.

PELL supports the following major commands:

**conc** *cmd1 args1 , cmd2 args2 , …*

**conc** causes PELL to execute the commands concurrently (i.e., in parallel). There is no communication between the commands, these simply happen in parallel. fork each of the children. The maximum number of commands is 5. If any of the commands redirect input or output, you must do the **redirection after forking,** but **before execing**. The getCommands function has been provided to simplify the process of getting the command arguments and redirection values.

Example: conc ls –l /bin > lsOne.txt , ls –l /usr/bin > lsTwo.txt , ls –l /etc > lsThree.txt

* Each of the ls commands are executing in parallel.
* The conc command prints each of the parallel commands showing the parent's process Id, child's process ID, the command, and command arguments. Your actual PID values will be different.

33009 33011: ls –l /bin

33009 33012: ls –l /usr/bin

33009 33013: ls –l /etc   
That output is written to stderr to not interfere with stdout.

* Since there are three commands, PELL has to create three children, (in this example) redirect stdout for each child, and **execvp** to the particular command for each child.

**pipe** *cmd1 args1 , cmd2 args2*

**pipe** causes PELL to create a pipe and fork a child for each *cmdi*. There are only two commands (however, for exta credit the maximum commands is 5). *cmd1* can have stdin redirected from a file. *cmd2* can have stdout redirected to a file.

You will have to use dup2 to redirect the pipes.

Example: pipe ls –l Data , sort –k5 -n > sort.out

* The pipe command prints each step showing a sequence, parent's process ID the child's process ID, and its command

1 33043 33045: ls –l Data

2 33043 33046: sort –k5 –n  
That output is written to stderr to not interfere with stdout.

* Since there are two commands, PELL has to create one pipe, and two children. The pipe is the output for step 1 and the input for step 2.

Notes:

1. To redirect a file to stdin:
   * open the filename as O\_RDONLY
   * dup2 its fd to STDIN\_FILENO
2. To redirect a file to stdout:
   * open the filename as O\_WRONLY|O\_CREAT|O\_EXCL
   * dup2 its fd to STDOUT\_FILENO
3. Copy the files for this assignment to your execution directory:

cp –r /usr/local/courses/clark/cs3423/2017Fa/Proj8/\* .

1. Create a directory named **Dout** below your execution directory. This will be useful in keeping the output in a separate directory which you can manually clean up easily (rm Dout/\*).
2. Larry provided the following files:

**cs3423p8Driver.c** driver program which calls your functions. It also provides a simple **split** function and **getCmdList** which gets the command list from the input. It invokes your functions. Do not code your code into this file.

**p8Input.txt** sample command text file to exercise your code

**p8Extra.txt** sample command text file for the extra credit.

**cs3423p8.h** include file to be used by your code and the driver.

**makefile** used by the make utility to compile your code and create the executable, **pell**. To compile and create pell:

$ make pell

For more information about the make utility, please see

http://www.cs.utsa.edu/~clark/setup/UnixMakeUtility.pdf

1. Place your code in **cs3423p8.c**. It should contain at least the following functions:

**concCmd** int concCmd(Cmd cmdM[], int iCmdCnt, Token tokenM[], int iTokenCnt)

* + - This is passed a pointer to a CmdList structure which has already been populated by the **getCmdList** function.
    - concCmd returns 0 if all children were launched successfully.
    - See the description of the **conc** command for more information.

**pipeCmd** int pipeCmd(Cmd cmdM[], int iCmdCnt, Token tokenM[], int iTokenCnt)

* + - This is passed a pointer to a CmdList structure which has already been populated by the **getCmdList** function.
    - pipeCmd returns 0 if all children were launched successfully.
    - See the description of the **pipe** command for more information.

1. Extra Credit (10pts + 100 / N)

* Change **pipeCmd** to support up to five processes; therefore, up to four pipes. Managing the open/close and use of dup2 will be the challenge.
* Your code must meet all requirements.
* It should not create any zombies or orphans.
* Late work will not receive extra credit.
* Your code must meet my programming standards to be eligible for extra credit.

1. Turn in a zip file (named LastnameFirstname.zip using your name). It should contain

* cs3423p8.c - your source code
* cs3423p8.h - (if you created one)
* makefile - your makefile to make the **pell** executable.
* In the notes in BlackBoard, specify if extra credit was completed.