CSCD 340
Lab 6

There are multiple pieces to this lab. You will submit your code and a PDF that illustrates you completed each part very systematically before you moved on to the next part. Your initial stubbed out code will not contain anything extra. Follow these directions very closely.

You are provided cscd340Lab6a.c – this contains the pipe code we wrote in class. You need to finish the parent portion of the code. Compile and execute it and show that it works. After you compile the code but before you run the file issue the ls –l | wc –w command on the command line and capture the output, then run your Lab 6a code and verify the results are the same. Save the command and the results in your PDF. NOTE: I didn't provide a makefile for this command I presume you can type gcc yourself and get it to compile.

You are provided cscd340Lab6.c and some .h files and a Makefile. None of these files are changeable

- You will need your makeArgs.c
- You will need your process.c

You must write

- containsPipe – In this function print out the count of the number of pipes in the string
    - Save this output in the PDF under the heading containsPipe

- parsePrePipe – This function will parse the string up to the pipe

  for ls -l | wc -w --> ls -l is what is parsed and stored in prePipe – hint makeargs

  print out the prePipe array in the function before you return it – save the output in the PDF under the heading prePipe

- parsePostPipe – This function will parse the string after the pipe

  for ls -l | wc -w --> wc -w is what is parsed and stored in postPipe – hint makeargs

  print out the postPipe array in the function before you return it – save the output in the PDF under the heading postPipe

- pipeIt – This function executes the pipe code similar to lab 6a; however, you have the prePipe in a 2D array and the postPipe in a 2D array so you will use execvp instead of execlp.

  NOTE
  - Your pipeIt command will only run one time and you will blow away the process image in the parent. This is the expected behavior right now

  Save your output in the PDF under the heading basicPipeIt

- Fix your pipeIt code so it does not blow away the process image. You will accomplish this by forking

  The parent process of that fork command is lab 6 – you will need a waitpid here waiting on the child to finish

The child of the fork command needs to become the new parent so

- In the child pipe
- fork again

The parent process from this fork is still the child of lab 6 but it is the parent process for the execution of the pipe -> in our example this is the wc -w

The child process from this fork is the grand child of lab 6 or the child of the pipe process -> in our example the ls -l

Once everything is complete for a valid pipe you should return to the prompt and be able to enter a new command.

If either side of the pipe can't be executed you must deal with this.

Run a valid command such as ls -l | wc -w and run 2 commands that are not valid such as ls -l | pws or pws | wc -w

Include the output runs in the pdf under the heading fixedPipeIt

Also include in the pdf the answer to the following questions

- Since exec changes the process image does your program leak memory?
- What happens when an invalid command is passed to the pipe either as prePipe or postPipe?
- If prePipe or postPipe are invalid is memory leaked?

TO TURN IN

- A zip
- All your code
- A PDF named cscd340Lab6systematic.pdf – containing your systematic creation of your code and your answers to the questions.
- A PDF named cscd340Lab6val.pdf – containing a complete valgrind run of your code showing it is absolutely 100% leak and ERROR SUMMARY free.

  HEAP SUMMARY:
  ==26593== in use at exit: 0 bytes in 0 blocks
  ==26593== total heap usage: 80 allocs, 80 frees, 790 bytes allocated
  ==26593==
  ==26593== All heap blocks were freed -- no leaks are possible
  ==26593==
  ==26593== For counts of detected and suppressed errors, rerun with: -v
  ==26593== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

Your zip file is named your last name first letter of your first name lab6.zip (Example steinerslab6.zip)