Programming Assignment 1

CS450 Spring, 2020

1. This assignment is an individual effort. It is due on 02/17/2020

2. **Requirements:**
   This programming assignment is inspired by one given at MIT (https://pdos.csail.mit.edu/6.828/2012/homework/xv6-shell.html). The MIT assignment asks the students to implement the pipe "|" and IO indirection ">" operators and somewhat more in a very simple shell. We ask you to implement ~~the sequence command execution operator ";" and~~ a program execution command called "`nonohup`" and the parentheses operators and more. You should do this assignment by modifying the skeleton code attached. Invoke your shell from the shell that comes with xv6 or from a robust UNIX.
   1) After your shell has started, it will give a prompt to the user. You should use "$S20" for the prompt.
   2) If the user types `cmd1;cmd2` after the prompt, `cmd1` will get executed, followed by `cmd2`. After `cmd2` is executed, your shell will give a new prompt in a new line, ready to execute the next command line. Already implemented.
   3) The commands `cmd1` etc. are real commands supported by xv6 and they take arguments. An example is the command `echo "A"`.
   4) If the user types `nonohup prog args`, when the user hits return, the prompt appears in a new line and the program `prog` and its arguments `args` will execute in the background. We call it `nonohup` because it is different from the POSIX command `nohup`. How is it the same and different?
   5) Your shell shall support a command line with a string of 3 or more commands connected by the ";" operator and `nonohup`. The commands and `nonohup` will be executed from left to right. The string "`cmd1;nonohup prog;cmd2`" will see `cmd1` gets executed first. After it finishes, `prog` and `cmd2` execute in parallel. After `cmd2` finishes, the prompt appears in a new line even though `prog` is still being executed.
   6) A command line, including `nonohup` if it is inside the command line, terminates when the user hit "`ctl+d`". The program that is handled by `nonohup` will terminate after it finishes.
   7) We ask you to write a program called `whatIf`. If invoked without an argument, it will take input from what you type into the keyboard until you hit a "`ctl+d`." If you give it a filename, it takes its input from that file. As soon as `whatIf` is invoked, it forks a child process. Then both the parent and the child will read from the same input, one line at a time. The parent will output what it gets into a file called `parent.txt` one line at a time and the child's output goes to a file

called `child.txt`. After they are finished with the input (from the keyboard or from the test file), they will exit individually. Run `whatIf` several times from your shell directly and from `nonhup`, and describe what you observe, and postulate what may have happened. You can do research to help your explanation.

8) Hint: `nonohup` can be developed in two ways: As a built in shell command; or as a computer program. The second approach will result in a simpler shell but may require you to use system calls like `signal()` that are not covered in your text book nor by me.

3. **Deliverables:**
    1) Source and executable objects with a README on how to build and execute them.
    2) A copy of the modified skeleton code, showing the modifications that you make and with comments that explains how your code works.
    3) A description and discussion on the behavior of the `whatIf` program. No more than one page. Font size must be greater than 8.5.
    4) The test data that you use and explanation on why the test data is of good quality. If you use the equivalence partitioning method to generate your test data, describe your equivalence partitions.
    5) A manual page for `nonohup` including how it is different from `nohup`.