Homework 3: Write your own shell

The programming part

For this assignment you are to write a simple program that will act as a shell. The program shall:

- Display a command prompt and read in a command line from the user(the prompt must be `CS361 >` , otherwise it cannot be detected by the test script)
- Your shell must support basic I/O redirection line the unix shell.
    1. `$ command` Run command, with stdin and stdout connected to their usual files. When `command` finishes, the parent should output `pid:%d status:%d\n` (with the proper relevant values inserted into the format string), and then print the prompt again and wait for more input.
    2. `$ command > filename` Run as in #1, but redirects the output of the command to `filename` . The existing contents of filename are overwritten.
    3. `$ command < filename` Run as in #1, but redirect the input of the command from `filename` .
    4. `$ command1 ; command2` Run command1 as in #1, wait for it to finish, and then run command 2 as in #1. You should only print the pid and status once all child processes have exited.
- Parse the command line into arguments, creating an array of character pointers, where array[0] points to the actual command and rest of the array elements point to the arguments to the command (Similar to main()'s argv[])
- Fork off a child and have the child load the requested program by passing the argument vector created in step 2 to exec() family of system calls. The parent should report the PID of the child before proceeding to the next step. (report means you are required to return strings contain "pid" and "status", e.g. `pid:11111 status:0`)
- Wait for the child to complete executing and report why it ended (exited or uncaught signal) and its exit value if available.
- Repeat for first step forever till user enters the command `exit`

  Command reads its input from filename instead of from stdin.

- Your shell should handle the following signals:
- SIGINT - Generated by Ctrl-C. This signal allows a user to terminate a running program. Your shell should not exit when user presses Ctrl-C or the process receives SIGINT but simply report that the SIGINT signal has been received by the shell. If the process receives SIGINT, you must print the string "caught sigint" on a line by itself, and then show the prompt again.
- SIGSTP - Generated by Ctrl-Z. This signal allows a user to terminate a running program. Your shell should not exit when user presses Ctrl-Z or the process receives SIGSTP but simply report that the SIGSTP signal has been received by the shell. If your shell receives SIGSTP, you must print the string "caught sigstp" on a line by itself, and then show the prompt again.
- The shell need not support background processes or running more than one child at a time.

your personal repository

Note that there is *no* skeleton code for the homework. Going to lab sections is highly advisable, as the TA has been and will be explaining basics of getting started with a shell. Both the lab section code, as well as code found in the book or book slides, are "fair game" from which to begin your coding. Previous solutions, or other students' work, are off limits and any cheating will be prosecuted to the fullest extent of university rules. *There are solutions on the Internet.* If we find you using them, we will give you an F in the class. Don't look at them, don't use them. It is not worth it.

Template

WHen you turn in the assignment, the TA should be able to compile your program by running `gcc -o hw3 hw3.c` in the `hw3` directory(Case sensitive). You are free to include any files e.g. `hw3.h` , it just has to compile using the above command.

Grading

Grading will happen using an autograder on gradescope. However, **the autograder will not be available until 1 week before the deadline**. This will **not** be enough time to finish the assignment if you wait to start until then.

Due Date

This assignment is due Monday, October 15, at 11:59 PM. See the syllabus for the late turnin policy. This assignment is worth just as much as every other homework, and they will only get harder from here, so getting as much credit on it as possible is important (don't turn it in late!).

Chris Kanich (https://www.cs.uic.edu/~ckanich/) · UIC Computer Science (https://www.cs.uic.edu/)