Programming Project 1: UNIX Shell

Course: CPRE 308
Instructor: Mai Zheng
TAs: Om (Section 1), Gavin (Section 2), Duo (Section 4)

Overview

- Project Description:
  - Create your own version of basic UNIX shell
  - Perform similar functions (e.g.: run built-in commands)
- Submission details:
  - Document containing a cohesive summary of what you learned in this project **(15 pts)**
    - Up to 2 paragraphs (3 if you have implemented the extra credit)
  - A zip (or tar) file containing:
    - Makefile that compiles your program without errors **(10 pts)**
    - Source code is neatly formatted and contains comments describing parts of the program **(10 pts)**
    - Required functionality works **(65 pts)**
      - Refer to Section 4 for expected output

Tasks

- Task 1: Basic interface (10 pts)
  - Custom prompt (5pts)
    - Your shell should accept a “`-p <prompt>`” option on the command line
    - If prompt is not provided then the default prompt must be “`308sh> `”
  - Run in infinite loop until user requests to exit (5pts)
    - Suggestion: Use `fgets()` to read user input
- Task 2: Execute Built-in commands (5 pts)
  - Refer to Section 2.2 for the built-in commands to be supported by your shell
  - Refer to Section 3.2 to reuse some system/library calls to execute a few built-in commands

Tasks (contd.)

- Task 3: Execute Program commands (30 pts)
  - Spawn a child process to execute user input **(15 pts)**
    - Use `fork()` to create child process
    - Use `execvp()` system call to execute user input
      - Refer to man page of `execvp` for its usage
  - Shell should notify if the requested command is not found and cannot be run **(5 pts)**
  - Print process ID (PID) when child process is spawned **(5 pts)**
    - Print before executing program command
  - Your shell should wait until child exits
    - No prompt will should be available for additional user input
    - Your shell should wake when most recently executed child process completes
      - Check `waitpid`
  - When a child process finishes, prints its PID and exit status **(5 pts)**
    - Refer to Lab 2 (handling exist status)

Tasks (contd.)

- Task 3: Background processes (10 pts)
  - Support executing commands in the background when “&” is in the input command
    - Child process should run in the background
      - Your shell will ***not*** wait for child process to exit
    - You still need to print the creation and exit status of these background processes
  - HINT:
    - Get exit status of background child process by calling `waitpid` with PID= -1 and `WNOHANG` set
    - Check periodically by executing every time the user enters a command

Tasks (contd.)

- Task 4: Extra credit (+5 pts)
  - Add a built-in command “jobs” that outputs the names and PID of your child processes
    - Maintain a data structure (e.g. linked-list) that can store required information
  - Clearly describe your implementation in the summary