CMSC 257 Assignment 5: Sample Unix Shell

Project due date: 11:59 pm EST, 04/26/17

What is a shell?

- Command line interpreter
 - You type "ls /etc"
 - The shell invokes the first parameter as a command, with the remainder as the parameters
 - eg: exec(ls,"/etc")
- Built-in commands
 - Most commands are separate executable programs
 - ls, rm, mv, make, gcc
 - Some commands are interpreted by the shell
 - cd, exit, pid, ppid.

Interactive vs Batch

- Interactive
 - User types commands in, hits return to invoke them
- Batch
 - shell reads from an input file
- What is the difference?
 - where the commands come from
- You need to implement the Interactive shell model.

Input/Output

- C has 3 standard files prepared for you
 - stdin = input
 - stdout = output
 - stderr = error output
- printf("foo") == fprintf(stdout,"foo")
- scanf("%s",str) == fscanf(stdin,"%s", str)
- fprintf(stderr,"Panic!") prints an error message separately

Process Control

- Your shell should execute the next command line **after** the previous one terminates
 - you must wait for any programs that you launch to finish
- You don't have to provide the functionality of launching multiple simultaneous commands with ";" separating them

Hints

- A shell is a loop
 - read input
 - execute program
 - wait program
 - repeat

- Useful routines
 - fgets() for string input
 - strtok() for parsing
 - exit() for exiting the shell
 - getpid() for finding the current process ID
 - getppid() for finding the parent process ID
 - getcwd() for getting the current working directory
 - getenv()/setenv()
 - chdir() for changing directories
- Executing commands
 - fork() creates a new process
 - execvp() runs a new program and does path processing
 - wait(), waitpid() waits for a child process to terminate

Requirements:

- <executable> -p -p should allow the user to select an user-defined prompt.
 Otherwise, the default should be "257sh>".
 - o Shell functions to be implemented separately: exit, pid, ppid, cd.
 - o For implementing "exit" from the shell, use the raise() signal handler.
 - o "cd" prints the current working directory; whereas "cd <path>" will change the current working directory.
 - All other shell commands will need a child process using fork() and then calling execvp().
- No input will be greater than 50 characters.
- Only the interactive system needs to be implemented (batch system is not needed)
- Background process execution (using &) is NOT required.
- Each time a child process is created, evaluate its exit status and print it out.
- ^C should not take us out of the shell; use a signal handler. Hint: you can use the same signal handler code from the slides.

Note: Late assignments will lose 5 points per day upto a maximum of 3 days. Code must be submitted in the prescribed format.

For questions on grading, contact: Joseph Nalluri <nalluriji@mymail.vcu.edu>