Shell: Process control in Foreground and Background

[Description] [Do Nots] [Hints] [Grading Guideline]

Description

This assignment helps you learn about processes and basic process management in a shell. You are asked to write a simple shell program called `sh550`.

Here is a sample code to get you started. It reads a line of input, breaks it up into individual tokens, and then waits for next line of input. Typing "exit" terminates the shell.

Your shell must work as follows. You start the shell by running your `sh550` program. This will print a prompt of your shell as follows:

```
sh550>
```

From here onwards, you should be able to execute and control **any program/command** just as you would in a normal shell. For instance

```
sh550> ls
[ Output of ls shown here. Your shell waits for ls to finish. ]
sh550>
```

Additionally, your shell should be able to do the following:

1. Execute commands with multiple arguments. For example:

```
sh550> Command arg1 arg2 arg3
[ Output of Command shown here. Your shell waits for Command to finish. ]
sh550>
```

2. Execute commands in either foreground or background mode. In foreground mode, the shell just waits for the command to complete before displaying the shell prompt again (as in the above example). In background mode, a command is executed with an ampersand `&` suffix. The shell prompt appears immediately after typing a command name (say Command1) and shell becomes ready to accept and execute the next command (say Command2), even as Command1 continues executing in the *background*. For example:

```
sh550> Command1 &
sh550> Command2
[Output of Command1 and Command2 may interleave here in arbitrary order. Your shell waits for Command 2 to finish.]
sh550>
```

3. Maintain multiple processes running in background mode simultaneously. For example:

```
sh550> Command1 &
sh550> Command2 &
sh550> Command3 &
sh550>
[Output of Command1, Command2, and Command3 may interleave here in arbitrary order. Shell does not wait for any of the commands to fi
```

4. List all currently running background jobs using "listjobs" command.

```
sh550> Command1 &
sh550> Command2 &
sh550> Command3 &
sh550> listjobs
List of backgrounded processes:
Command 1 with PID 1000 Status:RUNNING
Command 2 with PID 1005 Status:RUNNING
Command 3 with PID 1007 Status:FINISHED
sh550>
```

5. Bring a background process to foreground using the fg command with process ID as argument. For instance, continuing from the previous example:

```
sh550> fg 1005
[ Your shell waits for Command2 to finish. ]
sh550>
```

6. The `exit` command should terminate your shell. Take care to avoid orphan processes.

Do Nots:

- DO NOT use any special wrapper libraries or classes to borrow the basic functionality required in this assignment. If in doubt, ask the instructor first BEFORE doing so.
- DO NOT use the **system(...)** syscall to execute the programs in your shell directly.
- DO NOT write five or six different programs, one for each feature. Write **one single program** that includes all the above features.

Hints:

- Here is a sample code to get you started. It reads a line of input, breaks it up into individual tokens, and then waits for next line of input. Typing "exit" terminates the shell.
- Build and test one functionality at a time.
- Make backup copies of partially working versions of your code. This way, if you irreparably mess up your code, then you can at least roll back to your last backup copy.

- First design your data structures and code-structure before you begin coding each feature. Anticipate specific problems you will face.
- Check out man page for the following:
  - fork()
  - execv(), execl(), execlp(), execvp() (which one should you use?)
  - waitpid()
  - kill()

Grading Guidelines

This is how we will grade your assignment during the demo. So please prioritize your work accordingly.

- 10 - README, Makefile, Compilation without errors or warnings
- 10 - Executing a command with no arguments in foreground
- 10 - Executing a command with multiple arguments in foreground
- 10 - Executing a single command in background
- 10 - Executing multiple commands in background simultaneously
- 10 - Printing an accurate list of background processes with `listjobs` command.
- 10 - Bringing a background process to foreground using the fg command.
- 10 - Error Handling: Most important part here is to make sure that you check and handle the errors returned by ALL systems calls used in your program. Also check for other common error conditions in your program. But don't go overboard with error checking. We will not try to intentionally break your code with bad input that's irrelevant to the assignment's goals.
- Total score = 80
- During the demo, the TA will ask questions about your code. To get full credit, you need to be able to fully understand and explain your code, besides showing that the code works.