Breakout / Lab 13

Simple Foreground Jobs and Builtins

CSCI 1730 – Fall 2015

Problem / Exercise

Extend/adapt the REPL code you wrote for Lab 12 into a simple shell that supports simple foreground jobs and a subset of the builtins required for your `1730sh` project.

- A *simple foreground job* here means a foreground pipeline job with no quoted arguments nor IO redirections. When a foreground job runs, the shell waits until the job encounters certain status changes before re-prompting the user. The types of status changes that should bring the prompt back are job terminations (either normal or abnormal) and when a job is stopped. When a status change occurs, a message indicating the change should print out before the next prompt just as described in the project description. Remember, you can wait for a status to change to occur using `waitpid(2)`.
- The following interactive and job-control signals should be ignored by your shell's process: `SIGINT`, `SIGQUIT`, `SIGTSTP`, `SIGTTIN`, `SIGTTOU`, and `SIGCHLD`. The signals should **NOT** be ignored in any child processes created for job processes (instead they should assume their default disposition).
- For this lab, your prompt should look the same way it is described in the `1730sh` project description.
- The following builtins are required to be implemented for this lab (as documented in the `1730sh` project description): `cd`, `exit`, and `help`.

1 Group Brainstorm

Breakup into groups based on your seating and brainstorm about how to solve the problem or exercise. Make sure everyone understands the problem, and sketch out potential ways to move towards a solution. Perhaps something that was discussed during lecture might be useful?

2 Submit Individual Brainstorm

Login to eLC and submit a version of your group's brainstorm, written in your own words. You may add additional information if you want. You need to write enough in order to convince the grader that you understand the problem or exercise and that you have a plan for moving forward towards a solution. Please include the last names of the other people in your group in your submission. The brainstorm submission should be available on eLC in your assignment dropbox. We prefer that you submit your individual brainstorms before the end of your breakout period, however, you generally have until 3PM on the day of your breakout (as indicated on eLC) to submit them.

NOTE: Submissions that do not include an individual brainstorm will not be graded. Also, once you have completed your individual brainstorm, please use the remainder of the lab period to actually work on the lab. Failure to do this may result in the taking of attendance.

3 C++ Code

Make sure that all of your files are in a directory called `LastName-FirstName-lab13`, where `LastName` and `FirstName` are replaced with your actual last and first names, respectively. The main code for this lab should be written in a file called `lab13.cpp`.

3.1 `Makefile` File

You need to include a `Makefile`. Your `Makefile` needs to compile and link separately. Make sure that your `.cpp` files compile to individual `.o` files. The resulting executable should be called `lab13`.

3.2 `README` File

Make sure to include a `README` file that includes the following information presented in a reasonably formatted way:

- Your Name and 810/811#
- Instructions on how to compile and run your program.
- A Reflection Section. In a paragraph or two, compare and contrast what you actually did to complete the problem or exercise versus what you wrote in your initial brainstorm. How will this experience impact future planning/brainstorms?

NOTE: Try to make sure that each line in your `README` file does not exceed 80 characters. Do not assume line-wrapping. Please manually insert a line break if a line exceeds 80 characters.

3.3 Compiler Warnings

Since you should be compiling with both the `-Wall` and `pedantic-error` options, your code is expected to compile without `g++` issuing any warnings. For this lab, compiling without warnings will be one or more of the test cases.

3.4 Memory Leaks

You are expected to ensure that your implementation does not result in any memory leaks. We will test for memory leaks using the `valgrind` utility. For this lab, having no memory leaks will be one or more of the test cases.

4 Submission

Before the day of the next breakout session, you need to submit your code. You will still be submitting your project via nike. Make sure your work is on `nike.cs.uga.edu` in a directory called `LastName-FirstName-lab13`. From within the parent directory, execute the following command:

```
$ submit LastName-FirstName-lab13 cs1730a
```

It is also a good idea to email a copy to yourself. To do this, simply execute the following command, replacing the email address with your email address:

```
$ tar zcvf LastName-FirstName-lab13.tar.gz LastName-FirstName-lab13
$ mutt -s "lab13" -a LastName-FirstName-lab13.tar.gz -- your@email.com < /dev/null
```