

1 Objectives

To practice `fork()` and `exec` by implementing a very simple shell.

2 Overview

The first thing you need to do is to copy the directory `shelljr` we have left in the `grace` cluster under the `exercises` directory. Remember that you need that folder as it contains the `.submit` file that allows you to submit.

3 Specifications

For this exercise you will implement a very simplified shell. A shell is a C program that executes commands by forking itself and using `exec`. We have been using `tcsh`, but there are other shells like `ksh`, `sh`, etc. The name of our shell will be `shell_jr`.

4 Shell Jr Functionality

Your shell will have a loop that reads command lines and process them. The prompt for your shell will be `"shell_jr: "`. The commands your shell are required to handle are:

1. `exit` - When the user types `exit` the shell will stop executing by calling `exit()`. Before executing `exit`, the shell will print the message `"See you"`.
2. `hastalavista` - Has the same functionality as `exit` above.
3. `cd` - It will allow you to change to a directory. You can assume the user will always provide a directory.
4. A command with a maximum of one argument (e.g., `wc location.txt`). That means your shell should be able to handle commands like `pwd`, `date`, or `wc location.txt`.

5 Requirements

1. You must NOT use an `exec*` function to implement the functionality associated with the commands `exit`, `hastalavista`, and `cd`. For other commands you must create a child (via `fork()`) and use `execvp` to execute the command.
2. If the user provides an invalid command, the message `"Failed to execute "` followed by the command name will be printed. In this case the child will exit returning the error code `EX_OSERR`. Use `printf` to display the message and flush the buffer (`fflush(stdout)`). Notice the shell is not terminated by executing an invalid command.
3. You don't need to handle the case where the user just types `enter` (you can assume we will always provide a command).
4. Make sure you use `printf` to print the shell prompt and that you flush the buffer.
5. It is your responsibility to verify that your program generates the expected results in the submit server.
6. You must use `execvp` (and no other `exec*` system call).
7. Your code must be written in the file `shell_jr.c`.

8. You may not use `dup2`, `read`, `write` or `pipes`.
9. You may not use `system()` in order to execute commands.
10. You can assume a line of input will have a maximum of 1024 characters.
11. Provide a makefile that builds an executable called `shell_jr`. Name the target that builds the executable `shell_jr`. Feel free to add any other targets you need.
12. All your C programs in this course should be written using the compiler `gcc`, with the options defined in the `gcc_aliases_info.txt` file. This file can be found in the `info` folder of the public `grace` account.
13. Your program should be written using good programming style as defined at <http://www.cs.umd.edu/class/spring2015/cmsc216/content/resources/coding-style.html>

6 How to Start

You should start by creating a loop that reads lines and displays them. Then you should begin to process each line (starting with the `exit` and `cd` commands). You are free to process each line any way you want, however, reading a whole line using `fgets` and then processing the line using `sscanf` could make things simple. Keep in mind that if `sscanf` cannot read a value into a string variable, it will not change the variable. This could help you identify when a command has an argument or not.

7 Submitting your assignment

1. In the assignment directory execute the command **submit**.
2. Your assignment must be electronically submitted by the date and time above to avoid losing credit. See the course syllabus for details.

8 Grading Criteria

Your assignment grade will be determined with the following weights:

| | |
|--------------------------|-----|
| Results of public tests | 28% |
| Results of release tests | 72% |

9 Academic integrity statement

Please **carefully read** the academic honesty section of the course syllabus. **Any evidence** of impermissible cooperation on assignments, use of disallowed materials or resources, or unauthorized use of computer accounts, **will be submitted** to the Student Honor Council, which could result in an XF for the course, or suspension or expulsion from the University. Be sure you understand what you are and what you are not permitted to do in regards to academic integrity when it comes to assignments. These policies apply to all students, and the Student Honor Council does not consider lack of knowledge of the policies to be a defense for violating them. Full information is found in the course syllabus– please review it at this time.