# Writing a Simple Shell

2016-17, CSCI 3150 - Assignment 2

Release: 6 Oct 2016

**Early Bird Deadline: 13 Oct 2016 11:59AM**
**Deadline: 20 Oct 2016 11:59AM**

## Contents

# 1 Introduction

In this assignment, you are going to implement a mini-shell, namely, `asg2-shell`, using C/C++. If one invokes the program is `asg2-shell`, we shall see something like this:

```
root@4d167d2bbaa4:/opt/os# ./asg2-shell
[3150 Shell:/opt/os]=> ls
Makefile  asg2-shell  asg2-shell.c  grader.sh  testcases
[3150 Shell:/opt/os]=> bye
root@4d167d2bbaa4:/opt/os#
```

Figure 1: Your shell

The prompt has the format of `[3150 Shell:<current directory>]=>`. Your shell shall support:

- Execution of any Linux built-in command (e.g., `ls`) with basic error handling.
- Three shell-specific commands: `gofolder`, `log`, and `bye`.
- Basic signal handling: i.e., a user cannot exit the shell simply by typing `ctrl-c`.
- Basic command chaining: `&&` and ||.

## 1.1 Execution of any Linux built-in command

Your shell shall allow a user to execute any Linux built-in command, with basic error handling. Note that the user can input either an absolute path (e.g., `/bin/ls`) or just a filename (e.g., `ls`). If an absolute path is not given, your shell should search the program in the following sequence:

`/bin` → `/usr/bin` → `.` (current directory)

In case your shell cannot locate the program, your shell should report an error message "{`command name`}: `command not found`" (see Figure below).

```
[3150 Shell:/opt/os]=> ls
Makefile  asg2-shell  asg2-shell.c  demo-asg2  grader.sh  hello  testcases
[3150 Shell:/opt/os]=> /bin/ls
Makefile  asg2-shell  asg2-shell.c  demo-asg2  grader.sh  hello  testcases
[3150 Shell:/opt/os]=> hello
Hello World
[3150 Shell:/opt/os]=> ls -l
total 64
-rw-r--r-- 1 1000 staff    52 Oct  4 05:13 Makefile
-rwxr-xr-x 1 1000 staff 14008 Oct  4 06:10 asg2-shell
-rw-r--r-- 1 1000 staff  8795 Oct  4 07:03 asg2-shell.c
-rwxr-xr-x 1 1000 staff 14003 Oct  3 04:52 demo-asg2
-rwxr-xr-x 1 1000 staff   631 Oct  3 07:13 grader.sh
-rwxr-xr-x 1 1000 staff  8519 Oct  4 04:13 hello
drwxr-xr-x 1 1000 staff   204 Oct  4 04:13 testcases
[3150 Shell:/opt/os]=> hehe
{hehe}: command not found
[3150 Shell:/opt/os]=>
```

*

## 1.2 Three shell-specific commands

### 1.2.1 `gofolder`

This command is equivalent to `cd` command in Linux, for changing the working directory, like below:

```
[3150 Shell:/opt/os]=> ls
Makefile  asg2-shell  asg2-shell.c  demo-asg2  grader.sh  hello  testcases
[3150 Shell:/opt/os]=> gofolder testcases
[3150 Shell:/opt/os/testcases]=> gofolder ..
[3150 Shell:/opt/os]=> gofolder not-exist
{not-exist}: cannot change directory
[3150 Shell:/opt/os]=> gofolder
gofolder: wrong number of arguments
[3150 Shell:/opt/os]=> gofolder a b c
gofolder: wrong number of arguments
[3150 Shell:/opt/os]=>
```

After a successful operation, your prompt shall be updated with current directory name. Your shell shall also include basic error handling.

### 1.2.2 `log`

This command is equivalent to `history` command in Linux, which lists the log of the commands that a user has issued. The log shall list all commands (valid or invalid), like below:

```
[3150 Shell:/opt/os]=> ls
Makefile  asg2-shell  asg2-shell.c  demo-asg2  grader.sh  hello  testcases
[3150 Shell:/opt/os]=> hello
Hello World
[3150 Shell:/opt/os]=> hehe
{hehe}: command not found
[3150 Shell:/opt/os]=> log
[1]: ls
[2]: hello
[3]: hehe
[4]: log
[3150 Shell:/opt/os]=>
```

### 1.2.3 `bye`

This command is equivalent to `exit` command in Linux, which lets a user to terminate the shell (and back to the normal Linux `bash` shell), like below:

```
[3150 Shell:/opt/os]=> bye a
bye: wrong number of arguments
[3150 Shell:/opt/os]=> bye a b c
bye: wrong number of arguments
[3150 Shell:/opt/os]=> bye
root@ce237f19975d:/opt/os#
```

## 1.3 Basic signal handling

Your shell shall handle the following list of signals as follow:

| **Signal** | **Action** |
|---|---|
| `SIGINT (Ctrl + C)` | Ignore the signal. |
| `SIGTERM` (default signal of command "kill") | Ignore the signal. |
| `SIGQUIT (Ctrl + \)` | Ignore the signal. |
| `SIGTSTP (Ctrl + Z)` | Ignore the signal. |

## 1.4 Basic command chaining

Your shell should handle two logical operations: **AND**(&&) and **OR**(||), like below:

```
[3150 Shell:/opt/os]=> ls && hello
Makefile  asg2-shell  asg2-shell.c  demo-asg2  grader.sh  hello  testcases
Hello World
[3150 Shell:/opt/os]=> hello && ls -a
Hello World
.   .DS_Store  .gitignore  asg2-shell    demo-asg2  hello
..  .git       Makefile    asg2-shell.c  grader.sh  testcases
[3150 Shell:/opt/os]=> ls && hehe
Makefile  asg2-shell  asg2-shell.c  demo-asg2  grader.sh  hello  testcases
{hehe}: command not found
[3150 Shell:/opt/os]=> hehe && ls
{hehe}: command not found
[3150 Shell:/opt/os]=>
```

```
[3150 Shell:/opt/os]=> ls || hello
Makefile  asg2-shell  asg2-shell.c  demo-asg2  grader.sh  hello  testcases
[3150 Shell:/opt/os]=> ls -a || hello
.   .DS_Store  .gitignore  asg2-shell    demo-asg2  hello
..  .git       Makefile    asg2-shell.c  grader.sh  testcases
[3150 Shell:/opt/os]=> abc || ls
{abc}: command not found
Makefile  asg2-shell  asg2-shell.c  demo-asg2  grader.sh  hello  testcases
[3150 Shell:/opt/os]=> abc || ls -l
{abc}: command not found
total 64
-rw-r--r-- 1 1000 staff    52 Oct  4 05:13 Makefile
-rwxr-xr-x 1 1000 staff 14008 Oct  4 07:26 asg2-shell
-rw-r--r-- 1 1000 staff  8795 Oct  4 07:03 asg2-shell.c
-rwxr-xr-x 1 1000 staff 14003 Oct  3 04:52 demo-asg2
-rwxr-xr-x 1 1000 staff   631 Oct  3 07:13 grader.sh
-rwxr-xr-x 1 1000 staff  8519 Oct  4 04:13 hello
drwxr-xr-x 1 1000 staff   204 Oct  4 04:13 testcases
[3150 Shell:/opt/os]=>
```

# 2 Your assignment

You are given the following files:

| **Name** | Description |
|---|---|
| `/asg2-shell.c` | Source code of a runnable but non-functioning shell (**Work on it**). |
| `/asg2-shell` | Executable of a runnable but non-functioning shell. (Try to run it; Type `ctrl-d` to quit) |
| `/demo-asg2` | Executable, serve as the demo of what you shall implement. Our grading will also use this to define test cases. That is, the behavior of your shell shall **exactly** follow this demo. |
| `/hello` | Executable, a hello world program. |
| `/testcases/data` | Test data. |
| `/Makefile` | Makefile to compile `asg2-shell`. |
| `/grader.sh` | We will run this script to grade your assignment (**Don't touch**). |

## 2.1 Submission

Submit only 1 file, `asg2-shell.c`, to eLearning.

**Warning: DON'T change the file names or otherwise you get 0 marks**

## 2.2 To begin

Run `grader.sh`, you shall see something like this:

```
                                                          > [3150 Shell:/opt/os]
=> [3150 Shell:/opt/os]=>
Failed case        (left: your answer, right: correct answer)
[3150 Shell:PATH]=> ifconfig                              | [3150 Shell:/opt/os]
=> {ifconfig}: command not found
[3150 Shell:PATH]=>                                       | [3150 Shell:/opt/os]
=> [3150 Shell:/opt/os]=>
Failed case        (left: your answer, right: correct answer)
[3150 Shell:PATH]=> gofolder                              | [3150 Shell:/opt/os]
=> {gofolder}: command not found
testcases                                                 | [3150 Shell:/opt/os]
=> [3150 Shell:/opt/os]=>
[3150 Shell:PATH]=>                                       <
[Result] 0/25 test cases passed
```

This shell script feeds in some test cases to `asg2-shell` (which is not-quite functioning) and `demo-asg2` (which is functioning) matches their outputs, and reports the number of test cases passed. You can try supplying it with a test case number, e.g. `asg2-shell 2` to run test case 2. Initially, it shall report 0/25 test cases passed.

## 2.3 Your job

Your job is to start from the given `asg2-shell.c` and program it to make `grader.sh` reports:

```
[Result] 25/25 test cases passed
Congrats!
```

## 2.4 Assumptions

You can assume the following always holds in this assignment:

### Input

- An input command line has a maximum length of 255 characters, including the trailing newline character.
- An input command line ends with a new line character `\n`
- There would be no leading or trailing space characters in the input command line.

- A token is a series of characters without any space character. Each token is separated by **exactly one space character** only.

# 3 Grading

1. 25 test cases in total.
2. Passing one test case will get 4 marks. To encourage early submission, students who submit their **final** version before the early bird deadline, passing one test case will get 4.4 marks.
3. Your script shall output results to standard output stream (stdout). Otherwise, you will get 0 mark.
4. You are not allowed to invoke `system(3)` library call. Otherwise, you will score 0 mark.
5. You are not allowed to invoke any existing shell program, including but not limiting to: "`/bin/sh`", "`/bin/bash`", and etc. Otherwise, you will score 0 mark.
6. Note that your shell should not leave any zombies in the system when it is ready to accept a new user input. Otherwise, you will have **1 test case marks deducted.**
7. **Hardcoding won't work. We will use another similar set of test data and expected output when grading**.
8. The grading will be fully automated. The grading platform is our course's VM: Ubuntu 14.04 32-bit. Once we start the grading processing, we reserve the right to deduct marks from you for any request that requires TA's extra manual effort.
9. Our grader defines test cases based on the given `demo-asg2`. Therefore, the behavior of your shell shall **exactly** follow that demo. For example, if that demo shell outputs a message like:

```
command not␣␣found
```

with two spaces between `not` and `found`. Your shell shall also follow! The good news is that, **if this demo has not implemented something, your shell also does not need to do so**.

### 3.1 Late Submission Policy

We follow the late submission policy specified in the course outline.

## 4 Questions

If you have doubts about the assignment, you are encouraged to ask questions on our Facebook group.

## 5 Academic Honesty

We follow the University guide on academic honesty against any plagiarism.