ECE 364 Prelab 01 Handout

Conditionals, Loops, and Basic Commands in Bash

August 20, 2018

Name: ______________________________ ID: ee364_____

Passing this lab will satisfy course objective CO1.

Instructions

- Work in your Prelab01 directory
- Copy all files from ∼ee364/labfiles/Prelab01 into your Prelab01 directory. You may use the following command: cp ∼ee364/labfiles/Prelab01/* ./
- Remember to add and commit all files to SVN. **We will grade the version of the file that is in SVN!**
- Name and spell your scripts exactly as instructed. When you are required to generate output, make sure it matches the specifications exactly. Your scripts may be graded by a computer.

School Olympics (40 pts)

Introduction

A local high school recently held its own version of the Olympic Games for all students. There were a total of six different sports to compete in: Archery, Badminton, Diving, Fencing, Gymnastics, Swimming.

In your `Prelab01` directory, you are provided with a file called `olympics.txt` that contains a list of the names of all participants, along with how many medals each won in a given sport. The format of each line of the file is as follows:

```
<participant's name>,<sport>,<number of medals won>
```

Implementation Details

The judges of the Olympic Games have assigned you the task of collecting results from `olympics.txt` and printing out statistical data for each sport, outlined in the requirements below. You decide to use your newfound knowledge of Bash to write a script called `collect_stats.bash` that meets the given requirements.

1. The script should accept two arguments:
   (a) Name of the data file
   (b) Name of the olympic sport
2. If the correct number of arguments are not provided, print an appropriate message and exit with a return code of 1.
3. If the first argument is a non-existent file, print an error message and exit with a return code of 2.
4. Print the total number of contestants who participated in the given sport.
5. Print the total number of medals won by all contestants in a given sport.
6. Print the name of the contestant who won the most medals in a given sport, as well as the number of medals won.

Sample Output

Note: Your output must match the sample output exactly. Your script may be tested with a different data file.

```
$ ./collect_stats.bash
Usage: ./collect_stats.bash <file> <sport>

$ ./collect_stats.bash someFile Fencing
Error: someFile does not exist

$ ./collect_stats.bash olympics.txt Gymnastics
Total contestants: 59
Total medals won: 123
John B. Shadegg won the most medals: 8

$ ./collect_stats.bash olympics.txt Swimming
Total contestants: 50
Total medals won: 105
Brian Holtz won the most medals: 10
```

Mini Shell (40 pts)

Introduction

Your task is to write a mini version of a shell that can execute a small subset of commands.

Implementation Details

Your script, `mini_shell.bash`, should repeatedly ask the user what command they would like to execute, with the prompt "`Enter a command: `". The list of commands available to you, and the expected responses are given below:

- `hello`
  Prints "`Hello <username>`" to the terminal.
- `quit`
  Prints "`Goodbye`" and exits.
- `compile`
  Attempts to compile all files ending in ".c" in the current directory using the command
  `gcc -Wall -Werror -std=c99 <somefile.c> -o <somefile.o>`
  such that ".o" binaries with the same basename are created. For each compilation, you must check the return code of the gcc command and print a message indicating if the compilation succeeded or failed for that file.
- `run`
  Prompts the user to enter the name of an executable file, and a list of arguments to run the file with. If the given file is non-existent or not executable, print an appropriate error message. Otherwise, execute the file using the arguments provided.
- If the given command is none of the above, print an appropriate error message.

Sample Output

```
$ ./mini_shell.bash
Enter a command: hello
Hello ee364ta

Enter a command: compile
Compilation succeeded for: a.c
cc1: warnings being treated as errors
bad.c: In function main:
bad.c:2: error: implicit declaration of function echo
Compilation failed for: bad.c
Compilation succeeded for: b.c
Compilation succeeded for: works.c

Enter a command: run
Enter filename: asdknkjsd
Enter arguments: foo
Invalid filename

Enter a command: run
Enter filename: works.o
Enter arguments:
```

```
Hello World

Enter a command: run
Enter filename: b.o
Enter arguments: 125
You entered: 125

Enter a command: someCommand
Error: unrecognized input

Enter a command: quit
Goodbye
```