ECE 364 Lab 02 Handout

Bash File I/O and Arrays

August 28, 2017

Passing this lab will satisfy course objective CO1.

Instructions

- Work in your Lab02 directory.
- Copy all files from ~ee364/labfiles/Lab02 into your Lab02 directory. You may use the following command: cp -r ~ee364/labfiles/Lab02/* ./
- 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.
- To verify your SVN submission, you may run the following sequence of commands:

```
svn update
svn list
```

1 Arrays (30 pts)

```
SCRIPT NAME:           process_logs.bash
INPUT:                 From file
OUTPUT:                To STDOUT
NUMBER OF ARGUMENTS:   1
ARGUMENTS:             Input filename
RETURN CODE:           0 for success
                       1 for wrong number of arguments
                       2 for invalid input file
```

While working for a data hosting company, you begin to notice that some of the servers in the data center are overheating. In order to investigate the root cause of the problem, you set up a system to log temperatures throughout the day. This system generates files that contain a single line with header information followed by data. The sample file log1 has been provided to show you the format of this file. The header is always of the form:

```
time    name1    name2    name3    nameN
```

where the fields are separated by tabs and N is any integer greater than 1 (in other words, there is always at least one machine, but there may be many more).

The remaining lines are of the form:

```
time-value    temp1    temp2    temp3    tempN
```

where the fields are separated by tabs, the time-value is an integer greater than or equal to 0, and the temperature fields are integers greater than or equal to 0.

You decide to write a script to parse the large amounts of data coming from the logging system. This script has the following requirements:

- The script checks that only 1 argument has been provided. If this is not met, the script should print `Usage: process_logs.bash <input file>` and exit with a return code of 1.
- The script validates that the argument provided is a readable file. If it is not, the message `Error: <filename> is not a readable file.` is displayed and the script exits with return code 2.
- For each line of data in the file, the script should output a line in the form:

  ```
  Average temperature for time <time value> was <average> C.
  Median temperature for time <time value> was <median> C.
  ```

  where the "time value" is the integer from the file and the "average" is the arithmetic mean, 'median' is the median of all the temperature readings for the corresponding line rounded to two decimals places.
- At the end, script should print the average and median of all temperature corresponding to each machine (name1, name2,..)

  ```
  Average temperature for <machine> was <machine_average> C.
  Median temperature for <machine> was <machine_median> C.
  ```
- Save the output to a file called `<input file>.out`.
- The script should exit with return code 0 after saving the output to a file.

The following session gives sample output. **Only part of the output for the final command execution is given here (to save paper).** This is indicated by `<truncated>`. **Your script should give the full output.**

```
$ process_logs.bash
Usage: process_logs.bash <input file>

$ process_logs.bash not_readable_file
Error: not_readable_file is not a readable file.

$ process_logs.bash log1

$ cat log1.out
Average temperature for time 0 was 54.50 C.
Median temperature for time 0 was 53.50 C.

Average temperature for time 5 was 42.25 C.
Median temperature for time 5 was 41.00 C.

Average temperature for time 10 was 50.00 C.
Median temperature for time 10 was 50.00 C.

Average temperature for time 15 was 51.50 C.
Median temperature for time 15 was 50.00 C.

......... <truncated>..........

Average temperature for io was 46.14 C.
Median temperature for io was 44.00 C.

Average temperature for shay was 53.90 C.
Median temperature for shay was 55.00 C.

Average temperature for min was 53.23 C.
Median temperature for min was 52.00 C.

Average temperature for yara was 58.90 C.
Median temperature for yara was 62.00 C.
```

Note that the number of temperature readings will vary depending on the input file. You can see this by looking at the two given sample files, log1 and log2. In addition, the number of temperatures (lines) in a file will also vary depending on the file.

2 Sorting (30 pts)

```
SCRIPT NAME:           sort_logs.bash
INPUT:                 From file
OUTPUT:                To STDOUT
NUMBER OF ARGUMENTS:   1
ARGUMENTS:             Input filename
RETURN CODE:           0 for success
                       1 for wrong number of arguments
                       2 for invalid input file
                       3 for error when removing existing file
```

After completing the previous script, you realize that you would like to actually get a sorted list of all the highest temperature readings in the format:
`machine_name,time,temperature`
You will write a script that meets the following requirements to do this (Read all of these before beginning):

- There should be one argument passed to the script. If this is not the case, print the message `Usage: sort_logs.bash <input file>` and exit with error code 1.
- The script should check to make sure that the input file is readable. If it is not, print the message `Error: <filename> is not a readable file.` and exit with return code 2.
- The format of the input file is the same as in Part 1. To start, you have your script generate a new file of unsorted data when it is first run. The file name for this file should be `<input filename>.unsorted`. So if your input file is `log1`, the first output file should be named `log1.unsorted`. The data in this file is of form `machine_name,time,temperature`
- Once you have generated the unsorted output file, you should use the `sort` command to generate another output file that places the highest temperatures at the top of the file. Any ties should be broken alphabetically using the machine name. This output file will have the name `<input filename>.sorted`, similar to the unsorted filename in the above bullet point.
- If either output file already exists, you should attempt to remove the file. If you are unable to remove the file(s) using `rm -f`, exit with return code 3 after printing the message:
  `Error: Could not remove <filename>.`
  If the file is successfully removed, print the message:
  `Note: Removing existing file <filename>.`
- Your script should still output the average, median temperature for each line as in Part 1. It might be a good idea to simply copy over the script from Part 1 and use it as a starting point for this part.

Sample output is given here (Note that data is truncated as in the sample output for Part 1):

```
$ sort_logs.bash
Usage: sort_logs.bash <input file>

$ sort_logs.bash nofile
Error: nofile is not a readable file.

$ sort_logs.bash log1

$ cat log1.out
Average temperature for time 0 was 54.50 C.
Median temperature for time 0 was 53.50 C.
```

```
Average temperature for time 5 was 42.25 C.
Median temperature for time 5 was 41.00 C.

Average temperature for time 10 was 50.00 C.
Median temperature for time 10 was 50.00 C.

......... <truncated>..........

Average temperature for io was 46.14 C.
Median temperature for io was 44.00 C.

Average temperature for shay was 53.90 C.
Median temperature for shay was 55.00 C.

Average temperature for min was 53.23 C.
Median temperature for min was 52.00 C.

Average temperature for yara was 58.90 C.
Median temperature for yara was 62.00 C.

$ cat log1.unsorted
io,0,55
shay,0,52
min,0,78
yara,0,33
io,5,52
shay,5,47
min,5,35
<truncated>

$ cat log1.sorted
yara,100,79
yara,25,79
min,0,78
yara,75,78
min,95,77
io,70,76
shay,70,74
io,50,73
shay,50,72
yara,85,72
yara,95,72
yara,50,71
shay,25,69
yara,20,69
<truncated>

$ sort_logs.bash log1
Note: Removing existing file log1.unsorted.
Note: Removing existing file log1.sorted.

$ cat log1.out
Average temperature for time 0 was 54.50 C.
```

```
Median temperature for time 0 was 53.50 C.

Average temperature for time 5 was 42.25 C.
Median temperature for time 5 was 41.00 C.

Average temperature for time 10 was 50.00 C.
Median temperature for time 10 was 50.00 C.

......... <truncated>..........
```

3 I/O Redirection (20 pts)

```
SCRIPT NAME:           run.bash
INPUT:                 None
OUTPUT:                To STDOUT and to file
NUMBER OF ARGUMENTS:   0
ARGUMENTS:             None
RETURN CODE:           0 for success
```

3.1 Introduction

You have been provided with a directory called `c-files`, which, as the name states, contains several different .c files. You task is to compile and execute all of these files. You decide to write a short script called run.bash that performs this task.

3.2 Implementation Details

Your script must meet the following requirements:

- For each of the files in `c-files`, compile it using the command `gcc -Wall -Werror <filename>` and print the message "`Compiling file <filename>`".
- Direct STDERR to /dev/null. This means that if you get any errors while compiling, suppress them from the terminal and direct them to /dev/null.
- Check the return code of the gcc command to see if compilation succeeded. If it didn't, print the message "`Error: Compilation failed.`"
- However, if gcc was successful, print the message "`Compilation succeeded.`" and execute the compiled code in `a.out`. If execution produces any output, send it to a file called `<filename>.out`, where `<filename>` is the name of the `.c` file you compiled.

3.3 Sample Output

The following session gives an example of the output format that is expected.

```
$ ./run.bash
Compiling file err.c... Compilation succeeded.
Compiling file helloWorld.c... Compilation succeeded.
Compiling file sel_sort.c... Compilation succeeded.
Compiling file shift.c... Error: Compilation failed.

$ cat err.out
Running file err.c
$ cat helloWorld.out
Hello World
$ cat sel_sort.out
The array was:
        4       2       32      26      94
        53      3       10      24      17

The sorted array is:
        2       3       4       10      17
        24      26      32      53      94
```