School of Computer Science, McGill University

COMP-206 Introduction to Software Systems, Winter 2021

Mini Assignment 3: Advanced Unix Utils

Due Date March 10th, 18:00 EST

This is an individual assignment. You need to solve these questions on your own. If you have questions, post them on Piazza, but do not post major parts of the assignment code. Though small parts of code are acceptable, we do not want you sharing your solutions (or large parts of them) on Piazza. If your question cannot be answered without sharing significant amounts of code, please make a private question on Piazza or utilize TA/Instructors office hours. Late penalty is -15% per day. Even if you are late only by a few minutes it will be rounded up to a day. Maximum of 2 late days are allowed.

You MUST use `mimi.cs.mcgill.ca` to create the solution to this assignment. You must not use your Mac command-line, Windows command-line, nor a Linux distro installed locally on your laptop. You can access `mimi.cs.mcgill.ca` from your personal computer using **ssh** or **putty** and also transfer files to your computer using **filezilla** or **scp** as seen in class and in Lab A and mini assignment 1.

All of your solutions must be composed of commands from the list provided at the end of this assignment description and your scripts must be executable as is in mimi.

For this assignment, you will have to turn in one shell script. Instructors/TAs upon their discretion may ask you to demonstrate/explain your solution. No points are awarded for commands that do not execute at all. (Commands that execute, but provide incorrect behavior/output will be given partial marks.) All questions are graded proportionally. This means that if 40% of the question is correct, you will receive 40% of the grade.

Please read through the entire assignment before you start working on it. You can loose several points for not following the instructions. There are also some helpful hints given at the end of this document that can be useful to you.

Lab E provides some background help for this mini assignment.

Total Points: 20

Ex. 1 — Parsing sensor logs for analysis (20 Points)

The output produced by specialized software applications often contain a mixture of diagnostic details as well as useful data. By processing their log files, we can glean for various useful information. In this assignment, we will use the advanced Unix utilities that we covered in class to analyze the output log files from a temperature sensors monitoring program.
The log files that we will be using for this assignment is available under the directory hierarchy of `/home/2013/jdsilv2/206/m3/sensorlogs`. Please note that this directory may not be accessible through FileZilla, etc. It is primarily meant to be accessed from the Unix command line in mimi. These will also be the files that TAs will be using to test your scripts.
The log files are generated by a program that reads five different temperature sensors, once every hour (24 readings in a given day) and records these readings. If it was unable to read a particular sensor, it will indicate the corresponding sensor's reading as `ERROR`. Along with this, the program also logs various other information (such as rebooting sensors, etc.) which we are not concerned with.
A sample output of one of these log files is given below. (truncated for brevity). You can deduce the message formats from the log files given to you as part of this assignment. Please note that positive temperature readings do not have an explicit + sign associated with them. For simplicity, you can assume that the temperature values are limited in the range of `100.00` to `-100.00`, inclusive.

```
2021-02-01 00:02:07 sensor readouts  -12.35 -11.90 -11.97 -11.05 -11.65
```

```
2021-02-01 01:03:01 rebooting sensor 4
2021-02-01 01:03:02 sensor readouts  -13.85 -11.90 -12.97 ERROR -11.65
2021-02-01 02:04:00 rebooting sensor 5
...
2021-02-01 18:03:57 rebooting sensor 2
2021-02-01 18:08:47 rebooting sensor 3
2021-02-01 18:12:00 sensor readouts  -10.35 ERROR ERROR -7.05 -3.65
2021-02-01 19:04:22 sensor readouts  -11.85 -7.40 -6.97 -7.55 -4.15
...
2021-02-01 23:00:39 sensor readouts  -12.85 -10.90 -7.47 -10.55 -8.65
```

You will be writing a shell script `dataformatter.sh` that would process these log files.

1. **(1 Point)** The shell script is expected to be given the name of a directory, under which it will start the search for log files whose names are of the form `sensordata-*.log` as its argument. (Do not hard code the directory name in your script).
   If the script is not invoked with the correct number of arguments, it should throw an usage message and terminate with a code of 1.

```
$ ./dataformatter.sh
Usage ./dataformatter.sh sensorlogdir
```

2. **(1 Point)**
   If the passed argument is not a valid directory name, it should throw an error message and terminate with code 1. For this particular situation (and only here), the error message must be send to the standard error and not the standard output.

```
$ ./dataformatter.sh /nosuchdir
Error! /nosuchdir is not a valid directory name
```

   You do not have to explicitly check if you have the permissions to access the directory or the log files.

3. **(2 Points)** Within the shell script, use an appropriate Unix command to look for files starting under the given directory hierarchy that matches the specific file name pattern mentioned above (keep in mind that the recursive check log files might be under some subdirectories, etc.). Each log file only contains the information for that specific day. And each day has its own log file and never spread across multiple log files. No points are awarded for this question even if you miss one valid log file or include files which does not follow the pattern given to you.

4. **(8 Points)** For each log file found, you should produce an output of the following format that contains only the temperature information from the sensors along with a header. (Truncated for brevity). As you can see, a more structured format like this could be easily used by applications that visualize data, etc.

```
Processing sensor data set for <full path to the filename here>
Year,Month,Hour,Sensor1,Sensor2,Sensor3,Sensor4,Sensor5
2021,02,01,00,-12.35,-11.90,-11.97,-11.05,-11.65
2021,02,01,01,-13.85,-11.90,-12.97,-11.05,-11.65
2021,02,01,02,-14.35,-11.90,-13.47,-12.55,-11.65
...
2021,02,01,22,-12.85,-9.40,-7.47,-10.05,-7.15
2021,02,01,23,-12.85,-10.90,-7.47,-10.55,-8.65
====================================
```

   The script is basically only including the year, month, day and hour information, followed by the temperature reported by each sensor at that time. If a sensor's reading is `ERROR` in the original log file for that particular time, the script must instead output the previous readout for that sensor. Such "data cleaning" steps are necessary to use data sets with many analytical applications that cannot work with missing data.
   For simplicity, you can assume that the first readout for all the sensors in a given day does not error out. The output should follow the same order of time as in the original log file.
   When you are processing multiple logfile, you can process them in any order.

5. **(4 Points)** Immediately following the previous output produced from a log file, the script should produce the statistics as to what was the maximum temperature reported for a given hour and which sensor reported it,

as well as the minimum temperature and the sensor responsible for that. The format is given below. For this report, **it is important to ignore the sensors that has reported error for that hour** and consider only the sensors that were functioning and produced a valid reading in that hour.

```
Readout statistics
Year,Month,Hour,MaxTemp,MaxSensor,MinTemp,MinSensor
2021,01,30,00,-8.28,Sensor5,-10.22,Sensor4
2021,01,30,01,-8.28,Sensor5,-11.22,Sensor4
2021,01,30,02,-8.30,Sensor2,-11.78,Sensor3
...
2021,01,30,22,-1.78,Sensor5,-8.78,Sensor3
2021,01,30,23,-1.80,Sensor2,-6.22,Sensor4
====================================
```

As in the previous case, the output should follow the same order of time as in the original log file.

6. **(4 Points)** Once the script is done producing the above two statistics for each log file, we want the script to report on the health of the sensors across all those days (log files). For this purpose, we will have to count the number of times that each sensor reported an error for each day. If a sensor did not report an error, indicate with the value 0. The last field in each line is the total number of sensor errors on that day (sum of the individual sensor errors).

```
Sensor error statistics
Year,Month,Day,Sensor1,Sensor2,Sensor3,Sensor4,Sensor5,Total
2021,01,30,2,2,6,2,3,15
2021,01,25,3,6,3,1,0,13
2021,01,31,0,3,1,1,6,11
2021,02,01,2,3,1,2,2,10
====================================
```

The output should be sorted such that the dates with the larger number of (total) errors on the top (descending order). If two (or more) dates have the same number of errors, then order their lines in the output in the chronological order of dates. (I.e. Jan 31 is before Feb 1, etc. if they both have same number of errors.)

- You must write a reasonable amount of comments (to understand the logic) in your script. You can lose up to **-2 points** for not writing comments.
- Follow the sample output format that is given to you for the valid invocation. It does not take much effort to implement them. Not following it can result in a deduction of **-2 points** or more.
- The script **MUST NOT** create any temporary/intermediate files to do its work. Use the techniques already covered from previous assignments and labs to pass output of one command/utility to another. Violations would result in a deduction of **-3 points**.
- Any error messages from your program should be as a result of an explicit `echo` command in your script. Any error messages from commands/utilities used by your script should be handled by the script itself and not reported to the user. Violating this would result in **-2 points** deduction.
- Your script should run correctly irrespective of any valid date/time in the log file and should not depend on the values being only for specific year, month, etc. (**-3 points** deduction).
- Your submission should be a single script (file), specifically, do not put `awk` commands, etc., in a separate file. (**-2 points deduction**).
- For the log files in the test directory given to you for testing, your script should run under 5 seconds (clock time). Scripts that take longer than this may not get graded or maybe graded only for the outputs produced in that time. To give some perspective, a simple, unoptimized implementation of this solution runs well under 1 second.

WHAT TO HAND IN

Upload your script, `dataformatter.sh` to MyCourses. If MyCourses is not allowing files with `.sh` extension, you may use the `tar` command as discussed in class to make a `tar` file or a `tar.gz` file that contains your script and submit them. Please download your submitted tar files to double check if they are good. Erroneous invocation of the tar command can sometimes result in a bad file. Such submissions cannot be unfortunately graded.

ASSUMPTIONS

- You may assume that any files and directories that your script needs to access will have the necessary permissions for it to execute the tasks outlined in the assignment.
- The entries in the log files follow the order of time.
- Although the exact minute/second in which the information is recorded by the program is not very accurate, you can rely that each hour will have only one instance of it reading the sensors and that there are no missing entries.
- You can assume that the first instance of sensor readout in a file does not have any sensor errors.
- You can assume that all five sensors will not error out at the same time.
- You can assume that the valid values of temperatures are in the range `100.00` to `100.00` inclusive.

HINTS

This is a high-level outline to get you started with the first part of the output format in case you feel stuck. You are not obliged to follow it.

- Use the `grep` command to extract only the lines of interest from the log file.
- Can you find a way to use `sed` to change the date and time fields in this output so that it in the next step,
- You can use `awk` to extract only the fields of interest.
- In class we saw how to declare `awk` variables and use them. Figure out a way to apply that approach in keeping track of the previous value of each sensor so that you can use it if the current value is `ERROR`.

For the last part, (again, not obliged, there could be other ways to implement this).

- Use `awk` to perform most of your data extracting / formatting tasks for individual log files.
- Can you pipe the output of a `for` loop to another command?
- How to use `sort` command to work with a non-space delimiter?
- Figure out how `sort` command can be used to impose reverse sorting on one field (keys) followed by regular order on some other fields.

COMMANDS ALLOWED

You may use any option provided by these commands, even ones that have not been discussed in class. Not all of them may be required for you to build your solution.

`[[ ]]`	`!`	`basename`	`dirname`	`break`
`cd`	`continue`	`date`	`diff`	`echo`
`exit`	`export`	`for`	`grep`	`if`
`ls`	`printf`	`pwd`	`$( )`	`cp`
`shift`	`while`	`mv`	`find`	`sed`
`awk`	`sort`			

You may also use commands discussed in class but not listed here. You must not use a general programming language like Python, Java, C or other scripting languages like Perl, etc.

TESTING

There is no tester associated with this assignment. An file is provided that contains all the expected output for the valid invocation in `dataformatter.out.txt`. Please refer to this if you have questions on the format and also to compare your output with the solution.

Once you have done your basic verification, you can test your script in the following manner against the test directory hierarchy that contains the actual log files.

```
$ ./dataformatter.sh /home/2013/jdsilv2/206/m3/sensorlogs
```

This is how TAs will be testing your submission and against the same directory.

QUESTIONS?

Please use piazza. However, before posting your question, use the search functionality to check it has been already discussed. You should also look under "Mini 3 general clarifications" pinned post to check if a popular question has been already included there. They will not get individual responses again.