School of Computer Science, McGill University

COMP-206 Introduction to Software Systems, Fall 2020

Assignment 3: Advanced Unix Scripting

Due Date Oct 27rd, 23:55

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 -5% 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.

This is not a mini assignment. It is one out of the two regular assignments. It is worth 8% of your course grade.

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** as seen in class and in Lab A.

For this assignment, you will have to turn in two shell scripts. 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 lose up to 3 points for not following the instructions.

Lab D provides some background help for this assignment.
Total Points: 40

Ex. 1 — A script to parse webserver logs and produce metrics (30 points)

In the first part of the assignment, we will be to create a script, `webmetrics.sh`, which will parse webserver logs and produce three metrics:

1. Number of requests coming from the browsers Safari, Firefox, and Chrome.
2. Number of distinct users (distinct IP addresses) per day.
3. Top 20 popular requests by product ID.

You are given three sample log files (extract them for your use, do not write your scripts to run directly on tar files); **it is on these files that both you and the graders will be running the script.** Take a peek at the contents of the log files as you read the assignment description.
Hints are provided for the three main parts of Ex1. You do not have to solve the problems in the way described in the hint. You are free to come up with your own solutions as long as the result is reasonably similar.

1. Using `vim`, create a script on Mimi called `webmetrics.sh`.
2. **(8 points)** Print "`Number of requests per web browser`" to standard output, then count and display the number of requests coming from the browsers Safari, Firefox, and Chrome. For example,

```
Number of requests per web browser
Safari,171882
Firefox,48558
Chrome,148691
```

Hint: we recommend using `grep` with a simple pattern check to identify the appropriate requests. ***The numbers do not have to be exactly the same, but should have similar magnitudes and relative proportions between the numbers reported for the different browsers.***

3. **(10 points)** Print "`Number of distinct users per day`" to standard output, then count and display the number of distinct users per day where we assume that each IP address represents a unique user. For example,

```
Number of distinct users per day
22/Jan/2019,3987
26/Jan/2019,5567
```

Hint: we recommend isolating the date portion of each request using `awk`, creating a list of distinct dates, and then using `grep` to create a list of requests for each date. You can then isolate the IP address portion of each request, and count the number of distinct users.

4. **(12 points)** Print "`Top 20 popular product requests`" to standard output, count the number of requests per product ID, and print the top 20 popular product IDs with the number of requests for each. In the log files, you'll see many kinds of requests. Here, we'll be interested in those with a pattern of *<stuff>* `GET /product/XXXXX/`*<possibleotherstuff>* where `XXXXX` is the product ID (e.g., 31983). We will **not** consider any other patterns. An example output would be:

```
Top 20 popular product requests
34286,25
33954,18
33952,17
33956,16
33953,16
33670,15
11911,15
11947,12
33487,11
8658,11
34290,10
31828,10
8369,10
5681,10
33968,9
33712,9
33626,9
13027,9
11926,9
33722,8
```

Hint: we recommend that you start by isolating the lines with the requests described above, then extract the product ID from each line using `sed` or `awk`, sort the product IDs, count the number of occurrences for each product ID using `awk`. The `BEGIN` and `END` statements of `awk` along with some basic `if-else` constructs could help you perform this count. You can then sort this output according to the number of occurrences (requests) first and product ID second, and finally use an appropriate Unix command to get the top 20.

5. **(1 point)** If there are no errors, your script should exit with **exit code 0**.
6. **(1 point)** If multiple product IDs have the same number of requests, the highest product ID should come first, the second highest product ID should come second, etc. (see previous example). In any case only output a maximum of 20 product ids in the output.

7. **(1 point)** If no file is passed to the script, an error message should be printed and the script should exit with **exit code 1**.

```
$ ./webmetrics.sh
Error: No log file given.
Usage: ./webmetrics.sh <logfile>
```

8. **(1 point)** If the parameter does not represent a file, an error message should be printed and the script should exit with **exit code 2**.

```
$ ./webmetrics.sh this_file_does_not_exist
Error: File 'this_file_does_not_exist' does not exist.
Usage: ./webmetrics.sh <logfile>
```

9. **(1 point)** Your script should work with both relative and absolute paths.

Ex. 2 — Writing a script to run `webmetrics.sh` on some log files (6 Points)

For this exercise, you will now create a script, `runlogs.sh`, to run `webmetrics.sh` that you created in Ex1. We provided you with three log files `weblog1.txt`, `weblog2.txt`, and `weblog3.txt`. Your script must print "`Web metrics for log file <filename>`", a new line, and "`====================`" to standard output, and then run `webmetrics.sh` on each of them (see the included sample output `runlogs.txt` as to how everything should look in the end.). You `runlogs.sh` script should assume that the files will be present in the same directory from where it is being run.

Your counts do not need to be exactly identical, since the exact counts depend on the specific patterns you'll choose. The important thing will be that they are of the same magnitude, and similar relative proportions.

WHAT TO HAND IN

Upload both of your scripts, `webmetrics.sh` and `runlogs.sh`, to MyCourses.

COMMANDS ALLOWED

No restrictions. But you are not allowed to use other scripting languages such as Python, Perl, Java, etc.

TESTER

There is no tester script for this assignment. You should compare the output of your second script, `runlogs.sh`, with the sample output provided.

FOOD FOR THOUGHT!

The following discussion is meant to encourage you to search independently for creative and optimal ways to perform rudimentary tasks with less effort and does not impact the points that you can achieve in the above questions.

Try to optimize your webmetrics.sh script. Capture the time it takes to run for a very large log file and post it in piazza under the "A3 bragging rights" pinned post and see if you can compete and win against the execution times from your classmates.

You can find a huge file under /home/2013/jdsilv2/data/access.log

Do not copy it to your home directory as it is more than 3GB. You can run it in the following way.

```
$ time ./webmetrics.sh /home/2013/jdsilv2/data/access.log
```

Copy paste the real/user/sys time produced by the time command after executing your script. Also indicate on which host your ran the script. Does your script gets faster if you run it a second time immediately? Can you figure out what "optimizations" help in reducing the execution time of your script?