School of Computer Science, McGill University

COMP-206 Introduction to Software Systems, Fall 2022

Mini Assignment 3: Advanced Unix Utils

Due Date October 19, 18:00 EST

This is an individual assignment. You need to solve these questions on your own.

You MUST use `mimi.cs.mcgill.ca` to create the solution to this assignment. An important objective of the course is to make students practice working completely on a remote system. Therefore, 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. **If we find evidence that you have been instead using your laptop, etc., to do some parts of your assignment work, you might loose ALL of the assignment points.** This restriction applies only to the editing (coding) and testing of your programs. You are free to (and encouraged to) backup your source code regularly to your personal computer as a precaution against any misadventures. Delays incurred because you accidentally deleted your programs and you had not taken backups will not get any considerations.

All of your solutions should be composed of commands that are executable in `mimi.cs.mcgill.ca` and must be contained within the scripts that you submit. I.e., anybody else (such as your TA) should be able to just run these scripts in mimi on their own without any modifications and it should still work the same.

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.**TAs WILL NOT modify your scripts in anyways to make it work.**

Please read through the entire assignment before you start working on it. **You can loose up to 3 points for not following the instructions** in addition to the points lost per questions.

Lab C, D and E provides some background help for this mini assignment.

Total Points: 12

Ex. 1 — Parsing program logs for analysis (12 Points)

It is often necessary to parse the output produced by various specialized software systems to extract/generate data that is of specific interest to us. In this assignment, we will use the advanced Unix utilities that we covered in class to analyze the log files of a distributed software system.

The log files that we will be using for this assignment is available under the directory hierarchy of `/home/2013/jdsilv2/206/mini3/logset1`. 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.

The description of the software system is provided towards the end of the assignment under the section **logs of the distributed system**, to help you understand how the log files are generated, the meaning of various descriptions, etc. It is important to read it thoroughly to understand some of the techniques you will have to use to analyze the log files to generate the required output before continuing with the rest of the questions.

You will be writing a shell script `logparser.bash` that would process these log files.

1. **(0.5 Points)** The shell script is expected to be given the name of a directory as its argument, under which we will look for log files whose names are of the form `host.port.log`. Where `host` is made up of alpha-numeric

characters and - (like our SOCS servers). `port` is basically an integer. **Do not hard code the directory name in your script**. Each of these are log files of processes with identifiers `host:port`.
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.

```
$ ./logparser.bash
Usage ./logparser.bash <logdir>
```

2. **(0.5 Points)**
If the passed argument is not a valid directory name, it should throw an error message and terminate with code 2. For this particular situation (and only here), the error message must be send to the standard error and not the standard output.

```
 ./logparser.bash /tas/r
Error: /tas/r 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. **(6 Points)**
The script should parse all the log files and produce a comma separated output file (CSV) called `logdata.csv` of the following format (truncated for brevity), in the current directory (i.e., where you are currently in the shell when the script was invoked).

```
........
teach-node-05:40190,13,teach-node-05:40190,10:56:52.297790000,10:56:52.362437000,10:56:52.492420000
teach-node-05:40190,13,teach-node-05:40290,10:56:52.297790000,10:56:52.361714000,
teach-node-05:40190,13,teach-node-05:40390,10:56:52.297790000,10:56:52.362785000,10:56:52.363999000
teach-node-05:40190,13,teach-node-05:40490,10:56:52.297790000,10:56:52.363259000,10:56:52.364581000
teach-node-05:40190,13,teach-node-05:40590,10:56:52.297790000,,
teach-node-05:40190,13,teach-node-05:40690,10:56:52.297790000,10:56:52.366296000,10:56:52.366887000
........
```

In the above example, the first field refers to the process that initiated the broadcast, `teach-node-05:40190`, the second field is the message id (`13`), unique only to the broadcaster process, the third field is a process that recevied that broadcast message (so basically if you have 6 processes in the group, you will have a maximum of 6 entries in the CSV file per broadcast message - one per receiver). The fourth, fifth and sixth fields are broadcast time (available only in the sender's log), receive time and delivery time (the last two being available in the receveing process's log), respectively. So as you can see, the broadcast time is the same for all the 6 entries but the other fields could vary.
Please note that since the group communication (GC) messaging system is not very reliable, a broadcast message may not get to the other process (receive), or in some cases may not get actually delivered. This means that the entries for receive and/or delivery might be missing from the log files. In which case you leave those entries empty. You do not have to worry about duplicate delivery of the same message from a broadcaster. The output log CSV must be ordered based on the broadcast process identifier, message identifier, and receiver process identifier.

4. **(3 Points)** Using `logdata.csv`, the script must generate another CSV file (in the same directory), `stats.csv` which would contain a summary of the efficiency of the group communication system. (Truncated for brevity). Below example is only a sample format, the data may not be accurate.

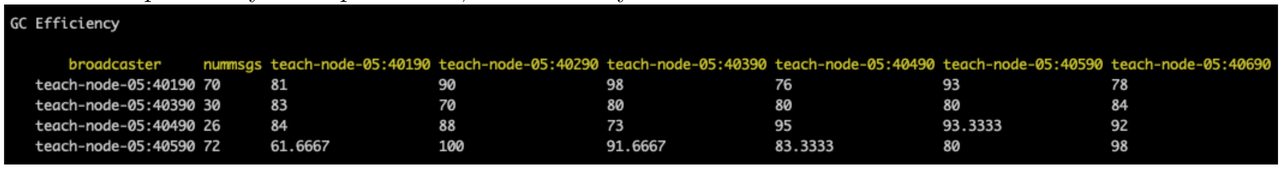
```
broadcaster,nummsgs,teach-node-05:40190,teach-node-05:40290,teach-node-05:40390,teach-node-05:40490,...
teach-node-05:40190,70,81,90,98,76,93,78
teach-node-05:40390,30,83,70,80,80,80,84
teach-node-05:40490,26,84,88,73,95,93.3333,92
teach-node-05:40590,72,61.6667,100,91.6667,83.3333,80,98
```

This is basically one row per processes' that do broadcasts, a corresponding column that shows how many messages were broadcasted by each of the process and a column per each receiver, indicating the percentage of messages that were delivered by the broadcaster at that receiver (as in there is an entry for delivery time in the receiver's log file for the said messages). Do not include processes that do not perform any broadcasts in the rows. The file should also have a header to name the columns, as given in the above example. Rows should follow the order of broadcast process identifiers and columns with in a row (left to right) should follow the order of the process identifers.

5. **(2 Points)** Next we will use the `stats.csv` file to generate an HTML file by the name `stats.html` (also in the same directory). This can be easily accomplished by replacing comma (`,`) with appropriate HTML tags. (Vi the sample file given to you to understand the format). Below example is only a sample format, the data may not be accurate.

```
<HTML>
<BODY>
<H2>GC Efficiency</H2>
<TABLE>
<TR><TH>broadcaster</TH><TH>nummsgs</TH><TH>teach-node-05:40190</TH>...</TR>
<TR><TD>teach-node-05:40190</TD><TD>70</TD><TD>81</TD>...</TR>
<TR><TD>teach-node-05:40390</TD><TD>30</TD><TD>83</TD>...</TR>
<TR><TD>teach-node-05:40490</TD><TD>26</TD><TD>84</TD>...</TR>
<TR><TD>teach-node-05:40590</TD><TD>72</TD><TD>61.6667</TD>...</TR>
</TABLE>
</BODY>
</HTML>
```

   You can also use the `lynx` command line browser available in mimi to look at the html file, which will be displayed like below (colors may be different/absent depending on your terminal software and is not relevant). Below example is only a sample format, the data may not be accurate.



```
GC Efficiency

      broadcaster     nummsgs teach-node-05:40190 teach-node-05:40290 teach-node-05:40390 teach-node-05:40490 teach-node-05:40590 teach-node-05:40690
   teach-node-05:40190 70      81                  90                  98                  76                  93                  78
   teach-node-05:40390 30      83                  70                  80                  80                  80                  84
   teach-node-05:40490 26      84                  88                  73                  95                  93.3333             92
   teach-node-05:40590 72      61.6667             100                 91.6667             83.3333             80                  98
```

ADDITIONAL RESTRICTIONS

- 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 (in the files) 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 should only create the files asked for and **MUST NOT** create any other 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, etc. 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 per occurrence.
- DO NOT assume that the process identifers will have only so many characters or so many integers.
- Your submission should be a single script (file), specifically, do not put `awk` commands, etc., in a separate file. (**-3 points deduction**). It might even be a 0 for the assignment, if it does not run as the TA expects because of this.
- For the log files in the test directory given to you for testing, your script should run under 1 minute (clock time), not "hang", etc.. 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 11 seconds. This could result in 0 or very low points depending on how far your script progressed.
- DO NOT Edit/Save files in your local laptop, not even editing comments in the scripts. This can interfere with the fileformat and it might not run on `mimi` when TAs try to execute them. This will result in a 0. No Exemptions !!. TAs will not modify your script to make it work.

WHAT TO HAND IN

Upload your script `logparser.bash` to MyCourses under the **mini 3** folder. Do not zip the file. Re-submissions are allowed.

You are responsible to ensure that you have uploaded the correct file to the correct assignment/course. There are no exemptions. **If you think it is not worth spending 5 minutes of your time to ensure that your submission that is worth 10% of your grade is correct, we won't either. NO Emailing of submissions**. If it is not in MyCourses in the correct submission folder, it does not get graded. **Because you do not know how to resubmit an assignment or run out of time to figure it out is not an excuse for emailing the assignment**. Imagine what will happen if all of you emailed your assignment instead of submitting it in MyCourses!

LATE SUBMISSIONS

Late penalty is -20% 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. Do not email me asking for extensions because you have other midterms, assignments, etc. You had the assignment out for a long time and the schedules were given to you in the beginning of the semester!

Neverthless, any requests made less than 24 hours from the deadline is automatically denied. It would have been too late to start your work anyways. Extensions are given only for extenuating circumstances.

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.
- Directory will have only valid log files in the correct format (name and contents). It will not have anything else. No empty directories either.
- The entries in the log files follow the order of time (as seen by that process). i.e., they are not deliberately scrambled to randomize time.
- You need not worry about the number of decimal digits for your stats computations, as long as the fractional part is in the same range. I.e., it is ok for `83.3333` to be `83.33` or `83.3333333`, etc. (minor approximations), but it should not be `84` or `83.345`, etc., which generally means you did the math incorrect some place.

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.

- Remember that each log file represents the events that happened in that process.
- Look for the broadcast messages to see which processes were broadcasting and what were their message ids.
- Remember that message ids are unique only with respect to a broadcaster process. I.e. a process p1 and another process p2 can both broadcast the same id, say i and both the messages will be send to all the process (and possibly received and delivered).
- Use the combination of process and message id to figure out which message you are tracking across the various log files.
- For each broadcast message, check in all the log files to see when it was received / delivered at each process.

A pseudo code for the main part is as given below (you are not obliged to follow it, if you do not understand it, build one based your logic). This is definitely not efficient, but will get you through the task.

```
for each process, get a list of message identifier's that they broadcasted
  find the broadcast timestamp of that message.
  go over each process log and extract the receive time and delivery time of that message in that process.
    produce an output that contains the broadcaster, message id, receiver, and the timestamps.
```

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

- This link provides a simple introduction to the concept of HTML tables.
- You will find it easier if you look at this problem as how to replace comma (`,`) with appropriate HTML tags as the separator between the fields. Verify the correctness of the data in the `stats.csv` file, etc., AND THEN think about the logic to transform that into an HTML table format.

COMMANDS ALLOWED

You may use any option provided by the following commands, even the ones that have not been discussed in class. Depending on your solution approach, you may not have to use all of these commands.

`$( )`	`$(( ))`	`$[ ]`	`[[ ]]`	`awk`
`basename`	`bc`	`break`	`case`	`cd`
`continue`	`date`	`diff`	`dirname`	`echo`
`exit`	`export`	`expr`	`find`	`for`
`function`	`grep`	`if`	`ls`	`pwd`
`read`	`sed`	`set`	`shift`	`sort`
`tar`	`while`	`cat`		

You can also use redirection, logical and/or/negation/comparison/math operators and any check operators (such as checking if something is a file) as required with the `[[ ]]` operator. You can also use shell variables and use/manipulate them as needed.

You are also welcome to use `awk` concepts not covered in class if you want to make life difficult, although they are not required.
You may not use any commands that are not listed here or explicitly mentioned in the assignment description. **Using commands not allowed will result in 3 points deduction per such command.**

LOGS OF DISTRIBUTED SOFTWARE SYSTEM

Read this inorder to understand the content and nature of the log files that you are analyzing.

The distributed software system that we are analyzing for this assignment consists of a set of processes that interact with each other by broadcasting messages to the entire group (to which all the processes are members). While all processes are "listeners" to get messages, only a subset of processes send out messages. By the property of a distributed system, every sender must ALSO send itself the message. Thus in an ideal situation, every process would get every message that was broadcast.

In order to accomplish the complex communication between multiple processes, the system leverages a group communication (GC) library.

Every processes logs the various events associated with into into their corresponding log files. The name of the log file follows a specific format, that helps us identify to which process it belongs to. For example, `teach-node-05.40490.log` belongs to the process `teach-node-05:40490` (i.e., remove the `.log` and replace `.` with `:` and you have the process identifier.

A sample log file (belonging to process `teach-node-05:40490`) is shown below, truncated (and artificially wrapped) at parts for brevity.

```
.........
Sep 19, 2022 10:56:52.030612000 AM DISLXXX.gcl.GCL broadcastMsg FINE: Broadcast message request received.
Translating to point-to-point messages : 15
.........
Sep 19, 2022 10:56:52.034887000 AM DISLXXX.gcl.GCL$GCLInbox run FINE: teach-node-05:40390:
Received a message from. message: [senderProcess:teach-node-05:40390:val:9]
.........
Sep 19, 2022 10:56:52.039017000 AM DISLXXX.dem206.GCLDemoCounter deliver INFO:
Received :4 from : teach-node-05:40390
.........
```

The above sample shows the three types of messages that are of interest to us. In the above description each message is wrapped into two lines due to lack of width in the PDF, but in the actual log files, they will be one continuous line (see the example log files given to you).

- The first message is what is generated by the GC library when it receives a message from its process to be broadcasted. The value at the very end is the message id (unique within a broadcast source process, but this means another process can also use the same id). So in the above example, since the log file belongs to the process `teach-node-05:40490`, this indicates that the process `teach-node-05:40490` requested the broadcast of message id `15` at time `10:56:52.030612000` (broadcast time). This is defintely an over simplification as we are ignoring the various other components of time. They can definitely be included with some extra engineering, but is not very interesting for our assignment.
- The second message is generated by the GC library when it receives a broadcasted message (including itself). In the above example, the GC library of process `teach-node-05:40490` is receiving a message with message id `9` from the process `teach-node-05:40390` at time `10:56:52.034887000` (receive time).
- The third message is generated by the receiving process when it actually recieves the message from it's GC library. In the above example, process `teach-node-05:40490` received the message id `4` that was broadcasted from the process `teach-node-05:40390` at `10:56:52.039017000` (delivery time).

A few extra things, some of it may be important to help understand how to analyze the logs, other might be totally irrelevant. So think about what is useful and ignore any other information.

- Within a process, the log messages are generated in the order of time. This means that the receive of a message will happen before the delivery of that message. Further, if the broadcasting process is also the same process, then the broadcast will happen before receiving (remember, even a sender needs to send itself the message).
- Since the group communication (GC) messaging system is not very reliable, a broadcast message may not get to the other process' GC library (receive), or in some cases may not get actually delivered. This means that the entries for receive and/or delivery might be missing from the log files. Of course you will not have a case where a message is delivered but not received or broadcasted, etc. No such absurdities.
- Also keep in mind that the notion of "current time" between different process may be different. This is because they are often running in different computers whose clocks maybe out of sync. What this means is that if you compare two timestamps `t` and `t'` from the log files of two different processes, `t` > `t'` does not mean the event associated with `t'` happened before `t`.
- Within a process, the messages are broadcasted in the sequential order of their message ids (with respect to that process). Message ids are also positive numbers.
- The messages are also received and delivered in the same sequential order with respect to a given sender process. I.e., there is no guarantee in ordering between messages comming from different senders, only within each sender.

TESTING

There is no tester associated with this assignment. Example files are provided for the expected outcome for the first log set directory. 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.

```
$ ./logparser.bash /home/2013/jdsilv2/206/mini3/logset1
```

This is how TAs will be testing your submission and against a similar directory / log files setup. The example output provided with the assignment corresponds to the above directory.

If you need an additional test case, you can try your script against another directory given below. But you will have to figure out for yourselves whether your outcome is correct or not.

```
$ ./logparser.bash /home/2013/jdsilv2/206/mini3/logset2
```

The log set directories are meant to be accessed from mimi. If you try to use FileZilla, etc., to browse it, you might encounter permission issues.

Remember !, the number of processes in each log set can be potentially different. Make sure that your script can figure this out by itself. The number of log files basically represent the total number of processes as each process has its own single log file.

QUESTIONS?

Remember to do step-by-step logic development. Do not write a very complex set of commands and pipeline them in the beginning. Take a single log file and see if your grep, sed, etc., is doing the transformation that you had in mind. Do not start off by writing a very complicated logic that looks like grep .... | sed .... | grep .. | awk ... | sed ... ! You will not know where your logic has a problem. Always ensure the first set of commands is doing what you want them to do, before you add the logic to the next set of commands! This is exactly what happens if you try to copy paste random commands from the Internet without knowing what they are doing. TAs will not help you if they sense this.

Remember, the files given to you represent the actual format, etc. So you should be able to look into these files to find some of the information you are looking for. The advanced commands you learned in class are also meant to be used for such verifications.

If you have other questions, post them on the Ed discussion board and tag it under mini 3, 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 the discussion board. If your question cannot be answered without sharing significant amounts of code, please make a private question on the discussion board or utilize TA/Instructors office hours.

Please remember that TA support is limited to giving any necessary clarification about the nature of the question or providing general advice on how to go about identifying the problem in your code. You are expected to know how to develop a high level logic, look up some syntax/options and most importantly, debug your own code. Lab D covers a lot of useful debugging techniques. We are not testing your TA's programming skills, but yours. Do not go to office hours to get your assignment "done" by the TAs.

Emailing TAs and Instructors for assignment clarifications, etc., is not allowed. TAs and instructors may convert private posts to public if they are not personal in nature and the broader student community can benefit from the information (to avoid repeat questions). Also check the pinned post "Mini 3 General Clarifications" before you post a new question. If it is already discussed there, it will not get a response. You can email your TA only if you need clarification on the assignment grade feedback that you received.

HALL OF FAME!

Need a bit more challenge? See if you can make your script fast and compete against your classmates. Make a copy of your completed assignment bash file as logparser_opt.bash. DO NOT modify your working assignment script !!

Now see if you can use techniques beyond what we discussed in class (and any commands you want to use) to make it faster. Test it against a larger data set.

```
$ time ./logparser_opt.bash /home/2013/jdsilv2/206/mini3/logsetM
```

A very naive solution takes 4 minutes to run on this data set.

Can you compete with your classmates and build the fastest solution (use the time command to measure the execution time). Post your execution time in Ed, under the pinned thread "Mini 3 Hall of Fame".

Remember **DO NOT** turn in the logparser_opt.bash for your assignment submission. Only your original script logparser.bash. Be careful not to modify your original assignment work and only play around with a copy.