

Scripting Languages

Assignment 1.2: Portfolio 2

ASSIGNMENT BRIEF

Important points before you begin:

- Read this brief very carefully in full, and at the earliest opportunity
- If any of this brief's stipulations are unclear to you, it is **your** responsibility to seek timely clarification from the lecturer/unit coordinator well before the assessment's due date
- Be aware that if you misinterpret this assignment brief, either in part or in full, this misinterpretation will not be accepted as grounds for an appeal of the result allocated

Overview

In this assignment you will be required to write a script that demonstrates the extent to which you have understood the *shell script (bash)* commands, programmatic techniques and concepts addressed in **Modules 1 to 8 inclusive**. This assignment is worth **20% (20 marks)** of your unit grade and its completion will help you build the remaining skills required for the final assignment to follow.

Your Academic Integrity Obligations

Your tutor, lecturer, and unit coordinator all take Academic Integrity very seriously and it cannot be stressed strongly enough how important it is that you fully understand your academic integrity obligations as a student of the University. In regard to all of this unit's assessments, all suspected instances of academic misconduct will be reported for investigation, which may result in substantial academic penalties for those concerned. If you are unfamiliar with the University's Academic Integrity Policy as it applies to all assignments you submit for this unit, [you can familiarise yourself with it here](#). If you are unsure of anything, please contact the Unit Coordinator for clarification before submitting this assessment.

AI Tool Utilisation

Whereas AI tools can serve as a very useful tool in the software engineering workplace, they are **not** to be used to complete any of the assessments in this unit, either *in-part* or *in totality*. This is because the focus of this unit is the development of authentic knowledge of, and skill in applying, various scripting languages to achieve specific outcomes, and not the use of AI tools to act on a programmer's behalf. This acquisition of authentic knowledge extends beyond the outcomes of this unit and is important to a student's future ability to be successful in obtaining a Work Integrated Learning placement, and ultimately, a job in industry. Employers already use AI tools extensively and know what they can and cannot do – they see no value in hiring graduates whose demonstrable skills do not exceed that of existing AI tools. Additionally, **where it's suspected that a student is [representing AI output as their own work](#), they will be called upon to [demonstrate functional knowledge of this work](#), which, if not forthcoming, may in turn lead to [allegations of academic misconduct](#).**

Please read the checklist below and watch the [associated video](#) **before** submitting this assignment.

ACADEMIC INTEGRITY TICK-BEFORE-SUBMIT CHECKLIST

PLAGIARISM

- ✓ I have not copy and pasted from external sources without appropriate citation
- ✓ My in-text and end-text citations follow APA 7 guidelines
- ✓ I have not used my own or other student's previous assignment work



COLLUSION

- ✓ I have not worked with any other students on this assignment unless permitted
- ✓ My assignment is not based on or derived from the work of any other students
- ✓ I have not shown or provided other student(s) with my assignment at any point



CONTRACT CHEATING

- ✓ I have not asked or paid someone to do this assignment for me
- ✓ I have not used any content from a "study notes" or "tutoring" service / website
- ✓ I have not had a friend or family member assist me with this assignment



IF YOU ARE UNSURE ABOUT ANY OF THE ABOVE, DO NOT SUBMIT YOUR ASSIGNMENT BEFORE SPEAKING WITH YOUR UNIT COORDINATOR OR ECU LEARNING ADVISOR

General Assignment Requirements

- Your script will be run in the *Azure Linux VM* provided at the beginning of semester. It is expected that you will develop and test your script in this Azure environment to ensure full compatibility.
- Your script **must** be a *bash* script and nothing else, i.e. the shebang line used is `#!/bin/bash`. Scripts of any other type, e.g. `#!/bin/sh` will receive a grade of zero (0).
- Ensure the script you write is *fully self-contained* and is not configured to be dependent on external files, libraries or resources to run. Non-observance of this requirement may cause your script to run incorrectly or not at all.
- Carefully check your submission before uploading it to Canvas. **What you submit is what gets assessed!** If you make a submission error, e.g. submit a wrong file, an empty .zip archive etc, no further/subsequent submissions will be accepted, which may result in a substantial loss of marks, or even a zero (0) result in some cases.
- You must only submit a **single** shell script (.sh) file named *logparser.sh* contained within a .zip file with the stipulated name in any *individual upload action*. Do **not** upload multiple files/zip archives in the same upload action as all will be considered invalid and will not be assessed. Also note that only the most recent individual submission made (as determined by the timestamp Canvas allocates) will be assessed.
- *If your script does not run for **any** reason, e.g. hard-coding of files/directories/paths, use of a development environment not compatible with the Azure Linux VM provided at the beginning of semester, only a partial mark may be awarded on a code-readthrough basis at the assessor's discretion. Your assessor will **not** fix non-functional, dysfunctional or incompatible scripts.*

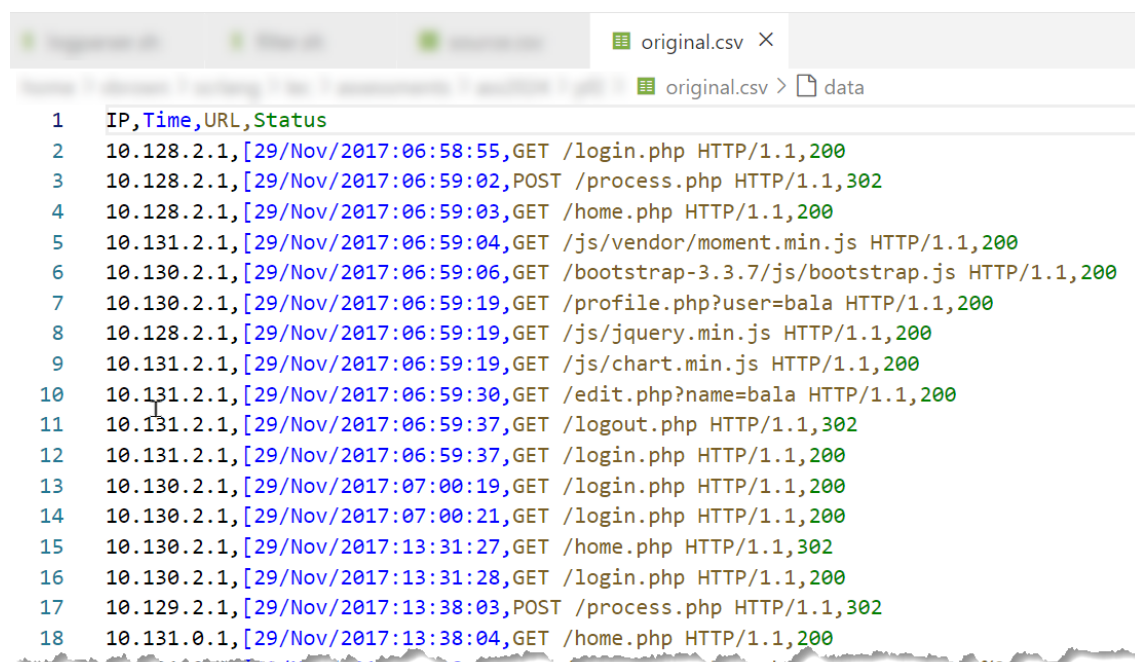
Task – Log Parser (20 marks)

Problem Description

Now that you have become somewhat famous in your organisation for writing the *filter.sh* script for the sales staff, a.k.a. Portfolio 1, something they are all now raving about, the people over in the ITSEC department have now asked you to work your shell scripting magic for them as well. Their problem is that they receive a raw, unformatted web log report each day that they then need to feed into a security analysis tool they use to spot potentially malicious attacks. However, this tool expects the .csv file uploaded into it to be in a specific six (6) column format, and with each column having a specific heading, otherwise it will not be accepted. Presently, ITSEC staff are spending a great deal of time manually converting the four (4) column raw web log file they receive each day into the formatted six (6) column equivalent their analysis tool requires. Your task therefore is to develop a bash/shell script tool that automates this process.

Raw Log File

The below screenshot shows a sample of the raw, unformatted web log report (in .csv format) that the ITSEC department receive each day.

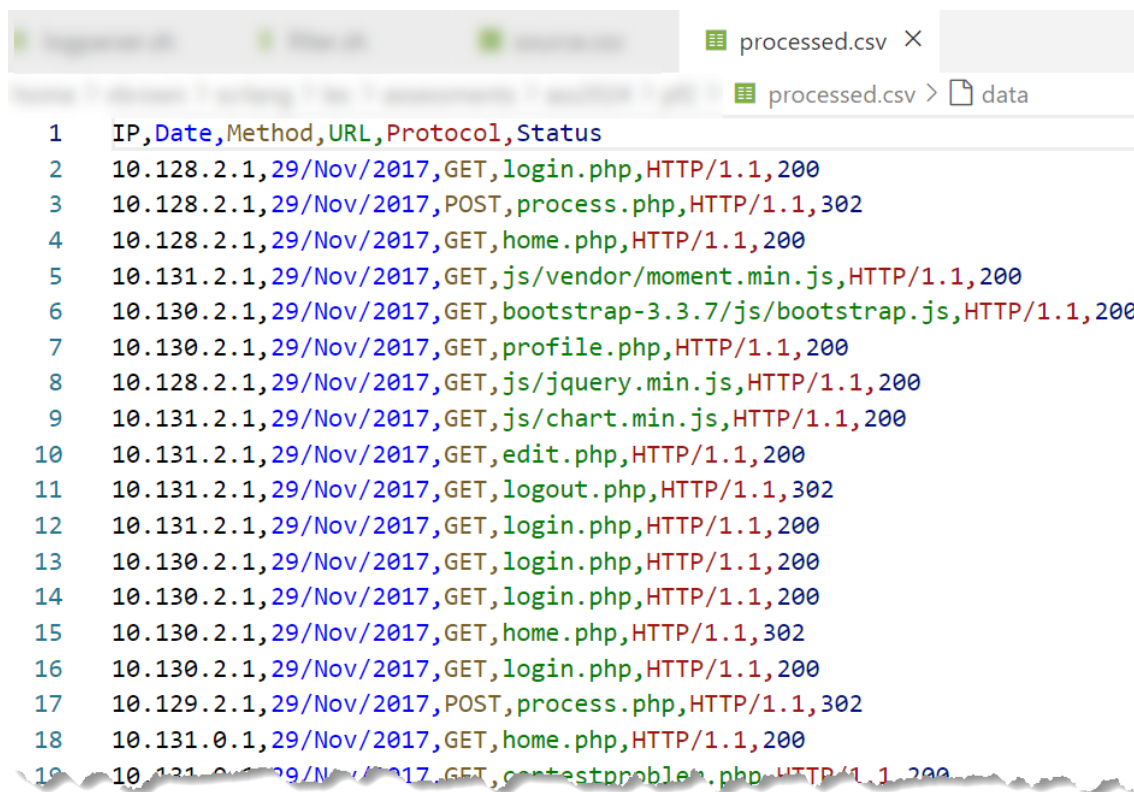


```
original.csv X
original.csv > data

1 IP,Time,URL,Status
2 10.128.2.1,[29/Nov/2017:06:58:55,GET /login.php HTTP/1.1,200
3 10.128.2.1,[29/Nov/2017:06:59:02,POST /process.php HTTP/1.1,302
4 10.128.2.1,[29/Nov/2017:06:59:03,GET /home.php HTTP/1.1,200
5 10.131.2.1,[29/Nov/2017:06:59:04,GET /js/vendor/moment.min.js HTTP/1.1,200
6 10.130.2.1,[29/Nov/2017:06:59:06,GET /bootstrap-3.3.7/js/bootstrap.js HTTP/1.1,200
7 10.130.2.1,[29/Nov/2017:06:59:19,GET /profile.php?user=bala HTTP/1.1,200
8 10.128.2.1,[29/Nov/2017:06:59:19,GET /js/jquery.min.js HTTP/1.1,200
9 10.131.2.1,[29/Nov/2017:06:59:19,GET /js/chart.min.js HTTP/1.1,200
10 10.131.2.1,[29/Nov/2017:06:59:30,GET /edit.php?name=bala HTTP/1.1,200
11 10.131.2.1,[29/Nov/2017:06:59:37,GET /logout.php HTTP/1.1,302
12 10.131.2.1,[29/Nov/2017:06:59:37,GET /login.php HTTP/1.1,200
13 10.130.2.1,[29/Nov/2017:07:00:19,GET /login.php HTTP/1.1,200
14 10.130.2.1,[29/Nov/2017:07:00:21,GET /login.php HTTP/1.1,200
15 10.130.2.1,[29/Nov/2017:13:31:27,GET /home.php HTTP/1.1,302
16 10.130.2.1,[29/Nov/2017:13:31:28,GET /login.php HTTP/1.1,200
17 10.129.2.1,[29/Nov/2017:13:38:03,POST /process.php HTTP/1.1,302
18 10.131.0.1,[29/Nov/2017:13:38:04,GET /home.php HTTP/1.1,200
```

Processed Log File

The below screenshot shows the result once the raw, unformatted four-column web log report has been converted into the formatted six-column format (also in the form of a .csv file) that the analysis tool requires.



```
processed.csv X
processed.csv > data
1 IP,Date,Method,URL,Protocol,Status
2 10.128.2.1,29/Nov/2017,GET,login.php,HTTP/1.1,200
3 10.128.2.1,29/Nov/2017,POST,process.php,HTTP/1.1,302
4 10.128.2.1,29/Nov/2017,GET,home.php,HTTP/1.1,200
5 10.131.2.1,29/Nov/2017,GET,js/vendor/moment.min.js,HTTP/1.1,200
6 10.130.2.1,29/Nov/2017,GET,bootstrap-3.3.7/js/bootstrap.js,HTTP/1.1,200
7 10.130.2.1,29/Nov/2017,GET,profile.php,HTTP/1.1,200
8 10.128.2.1,29/Nov/2017,GET,js/jquery.min.js,HTTP/1.1,200
9 10.131.2.1,29/Nov/2017,GET,js/chart.min.js,HTTP/1.1,200
10 10.131.2.1,29/Nov/2017,GET,edit.php,HTTP/1.1,200
11 10.131.2.1,29/Nov/2017,GET,logout.php,HTTP/1.1,302
12 10.131.2.1,29/Nov/2017,GET,login.php,HTTP/1.1,200
13 10.130.2.1,29/Nov/2017,GET,login.php,HTTP/1.1,200
14 10.130.2.1,29/Nov/2017,GET,login.php,HTTP/1.1,200
15 10.130.2.1,29/Nov/2017,GET,home.php,HTTP/1.1,302
16 10.130.2.1,29/Nov/2017,GET,login.php,HTTP/1.1,200
17 10.129.2.1,29/Nov/2017,POST,process.php,HTTP/1.1,302
18 10.131.0.1,29/Nov/2017,GET,home.php,HTTP/1.1,200
19 10.131.0.1,29/Nov/2017,GET,contestproblem.php,HTTP/1.1,200
```

Your task therefore is to develop a shell script tool that automatically converts an existing raw, unformatted four-column web log file (.csv) to the equivalent formatted six column .csv file shown above.

Important Points:

- The tool (script) you develop will be used by ITSEC staff on the command line of their Linux workstations as this is the environment they spend most of their time in.
- Your script will be used by ITSEC staff of varying skill and experience, so robust user input validation and informative user messaging are essential.
- Your script may well become someone else's responsibility to manage and extend upon in the future, therefore it must be properly and accurately documented to allow for this without any input from you as the original developer.
- The script must adhere to the organisation's coding guidelines (as addressed in the relevant lectures).
- Assume that the script will always be run in the same directory as the unformatted four-column web log .csv file that needs converting.

Required Script Functionality

- The user will provide the name of the log file to be parsed (converted) and the name they want for the parsed file (when created by your script) at the command line, i.e.
`./logparser.sh original.csv processed.csv`
- The user is **not** to be *prompted* for input at any point in the script's execution.

- The arguments provided by the user on the command line are to be fully validated **before** the script's main logic is allowed to execute. If any of the input arguments are invalid, the user is to be accurately *informed* of what the error is and how to correct it when they next attempt to run the script. The script must always terminate with an appropriate error code when an error is encountered.
- Your script will also need to report the total number of records that have been processed to the terminal when finished

```

$ ./logparser.sh original.csv processed.csv
Processing...
500 records processed...
$ ls -l
-rw-r--r-- 1 vbrown vbrown 37697 Mar  4 13:42 original.csv
-rw-r--r-- 1 vbrown vbrown 28614 Aug  3 09:10 processed.csv

```

- Perhaps needless to say, the results in the output file **must** be fully *correct* and *complete* in the context of the command line arguments provided.

Other Compulsory Requirements

- Call your script *logparser.sh*
- Your full name and student number **must** be placed at the top of your script (as comments) immediately after the *shebang* line.

```

$ logparser.sh
1  #!/bin/bash
2
3  # Student Name: Your Full Name
4  # Student ID: Your Student Number
5

```

- To construct your script, use a carefully considered combination of commands, utilities and programmatic techniques sourced from *Lectures 1-8* inclusive. This now **includes** *awk*, *sed* and functions. **Do not** use *bash/shell* script commands, utilities and programmatic techniques not addressed in *Lectures 1-8* inclusive as this will result in a substantially lower mark being awarded (see *rubric* for details).
- Your script **must** be properly documented, accurately explaining **all** of the code elements it contains in your own words. *Note: Documentation (comments) that is not relevant and accurate in regard to the code they describe, or a complete lack of documentation, will not only attract a lower mark, but may be viewed as suggestive of possible academic misconduct.*
- **IMPORTANT NOTE: AI-generated comments will receive a mark of zero (0) in this section of the rubric**
- Any *temporary* files and/or folders created by your script in the course of its execution **must** be programmatically removed from the assessor's system when the script terminates.

- The efficiency and correctness with which the commands, utilities and programmatic techniques within your script have been utilised will also form part of your mark, so please pay close attention to this aspect of your code as well. For example, your *logparser.sh* script is expected to make use of **appropriately selected** and **correctly applied** commands/options from *Lectures 1-8* inclusive. The total lines-of-code your script employs will also be considered, with deductions applicable should this significantly exceed the total deemed necessary to achieve the stipulated outcomes.

Test Files

Two (2) files have been provided to you, these being:

1. *original.csv*
2. *processed.csv*

The *original.csv* file contains 500 unparsed records. The *processed.csv* file contains the same 500 records, however, these have been parsed into the six (6) columns stipulated (by your lecturer's solution). You'll be able to use the *processed.csv* file provided as a reference against which to compare the results your own *logparser.sh* solution is producing throughout the development process to determine if your code is working correctly and generating accurate outputs.

Important Note:

Your assessor will **not** use either the *original.csv* or *processed.csv* test files for the marking process so do not hard code either of them into your script. The input .csv file your assessor uses for assessment will be precisely the same as *original.csv* in every respect **except** it will:

1. Have a *different* name (but will still have the .csv file extension)
2. Contain a *different* set of values (different set of records, but with the same column structure)
3. It will contain a *different* number of rows (it won't be 500 rows as is the case for *original.csv*)

Further, your assessor will not use the name *processed.csv* for the output file name, but rather, something unique (but will still have the .csv file extension).

It's **very** important that you understand these statements about the test files provided, and seek timely clarification if you do not!

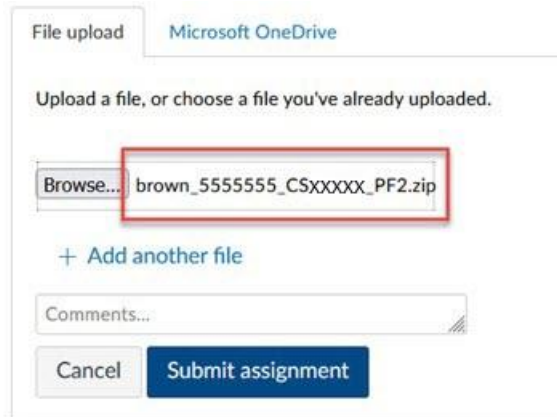
Assessment Rubric

On Canvas.

How to submit your portfolio to Canvas

Submit a **single** shell script (.sh) file named *logparser.sh* contained within a **.zip** file to Canvas with the following naming format (use *your* surname/student number):

[surname]_[student-ID]_CSP2101_PF2.zip



File upload [Microsoft OneDrive](#)

Upload a file, or choose a file you've already uploaded.

Browse... brown_555555_CSXXXX_PF2.zip

[+ Add another file](#)

Comments...

[Cancel](#) [Submit assignment](#)

Do **not** submit any files other than that stipulated above. Further, even though there is no restriction on how many times you make individual submissions (each of which gets its own unique timestamp in Canvas), do **not** upload multiple files/zip archives in the *same upload action* as all will be considered invalid and not be marked.

END OF ASSIGNMENT BRIEF