Scripting Languages
Assignment 3: Software-based Solution (Script and Video)

ASSIGNMENT BRIEF

Overview

In this assignment you will be required to write a more challenging script that demonstrates the extent to which you can apply the *shell script* concepts and practices covered in the unit to create a useful Linux (bash) utility that fulfills a specific brief. The assignment is comprised of two (2) components, these being:

1. A shell script (30 marks)
2. A video (10 marks)

For a *total* of **40** marks.

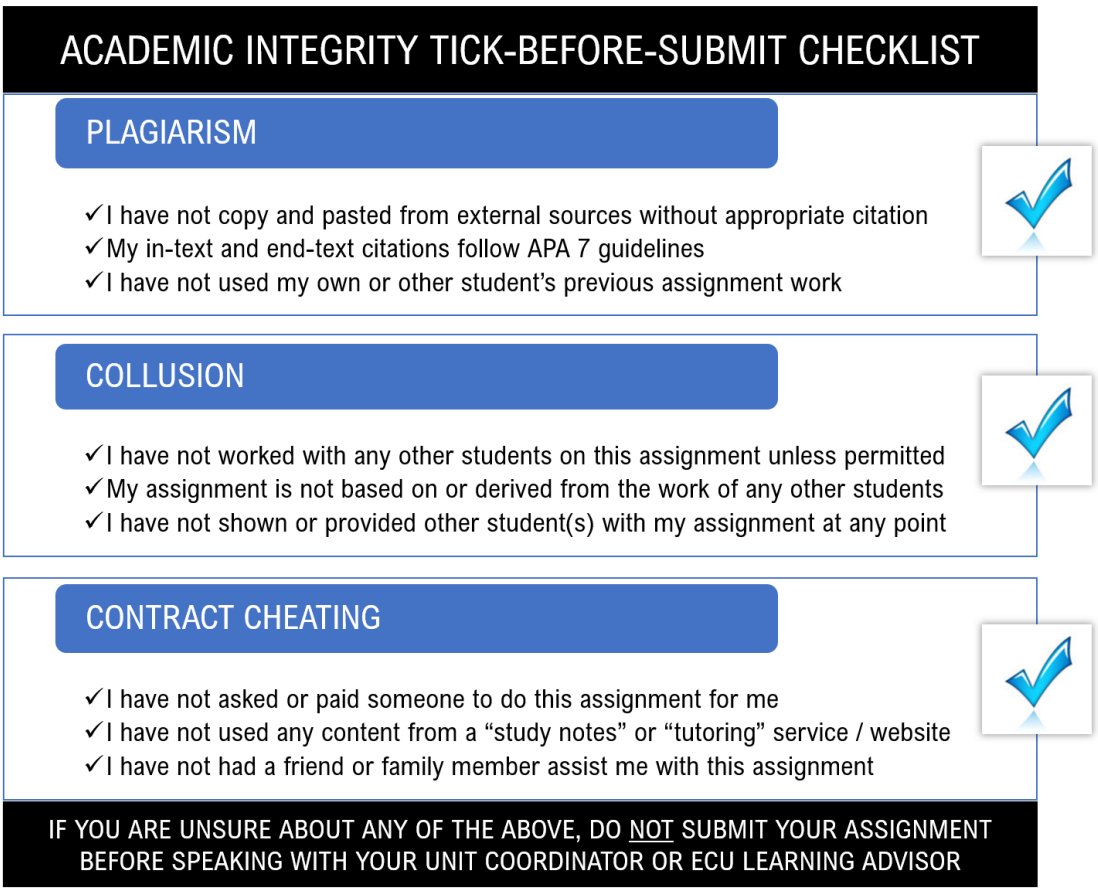
Your Academic Integrity Obligations

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, students representing AI output as their own work may be called upon to demonstrate functional knowledge of the 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.



General Assignment Requirements

- Your script will be run in *VS Code for Linux* on a *Ubuntu Desktop 18.0.4 LTS/GNU bash v5.0.17* instance or higher. Make sure you develop your script in an environment that meets this specification. Incompatible scripts often result in substantial loss of marks, or even no marks if they do not run at all.
- 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 in the assessor's environment, with substantial loss of marks, or even no marks, often being the outcome.
- 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. Also note that only the most recent submission made before the due date/time will be assessed.

Search Binaries – Script (30 marks)

Using **only** the commands, utilities and programmatic techniques addressed in the Unit's content materials (Modules 1-8 inclusive), write a script that allows the user to search the commands and utilities located in the Linux **bin** (*/bin*) directory based on specific options and arguments provided.

Required Script Functionality

1. The script is to support and implement **all** of the options (flags) and argument(s) in the table below when provided at the command line at run time.

OPTION	ARGUMENT
-s *[string]* *EXAMPLE:* `./searchbin.sh -s vmware`	Return binaries that **contain** the *specified string* *Note:* *Option is to be accepted as lowercase only* *Argument input is to be case-insensitive*
-b *[operator],[bytes]* *EXAMPLE:* `./searchbin.sh -b GT,500000` ***Note:*** *The comma is compulsory; no space is permitted between the operator and the number of bytes stipulated* ***Both** an operator and a bytes value must be provided*	Return binaries that **match** *both* the *operator* and *bytes value* provided **GT** – greater than,[value in bytes] **LT** – less than,[value in bytes] **LE** – less than or equal to,[value in bytes] **GE** – greater than or equal to,[value in bytes] **EQ** – equal to,[value in bytes] **NE** – not equal to,[value in bytes] ***Note:*** *Option is to be accepted as lowercase only* *Operator must be case-insensitive* *Argument must be a integer value only*
-l *(as in **l** for look)* *EXAMPLE:* `./searchbin.sh -l`	Return binary names that are **symbolic links** to another binary, and the alternate binary each points to ***Note:*** *Option is to be accepted as lowercase only*

Table 1: Script functionality requirements

2. If no flag/argument is provided when *searchbin.sh* is run, the script is to terminate with an appropriate error message, e.g. "*No options/args passed. Exiting...*"
3. The script is to allow for only **one (1)** option/argument(s) to be run at a time, so if multiple options/arguments are passed by the user, the script is to terminate with message to user to try again with just a single option/argument(s)
4. All options and arguments passed at the command line are to be **fully** validated, and in the event an invalid option and/or argument(s) are passed, this must be detected, and the script terminated with an appropriate error message, e.g. "*Invalid [specifics] passed. Exiting...*" and exit code
5. All output of results to terminal **must** be in a fully-aligned columnar format with binary sizes formatted precisely as shown in the screenshots below
6. In the event that *no matches* are found for any particular search (options -s, -b or -l), then the user is to be informed of this with an appropriate message, e.g. "*No matches found*", and the script terminated
7. Your script will be run in a working directory of the assessor's choosing, which will **not** be the */bin* directory. Do not, therefore, hard-code your script to run in/from any particular directory.
8. Apart from getting a listing (ls) of the /bin directory's contents for the purposes of the script's functions, your script is **not** to do anything else in or to the /bin directory, e.g. do **not** copy the */bin* directory to another directory! If anything like this is noticed during the tutor's initial code readthrough, your script will **not** be run, and only partial marks will be assigned.

9. Any files and/or folders created by your script in the course of its execution are to be removed upon the script's termination. *Please note: There will be a one (1) mark deduction from your achieved score if any files and/or folders created by your script are left behind on the assessor's machine when it terminates*.

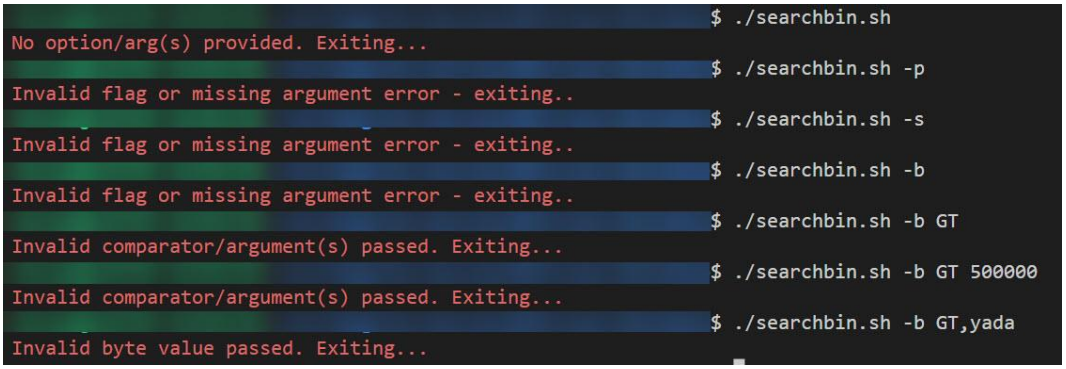
Other Compulsory Requirements

1. Call the script *searchbin.sh*. *Note: There will be a one (1) mark deduction from your achieved score if you script is submitted with a different name*.
2. Your full name and student number **must** be placed at the top of your script (as comments) immediately after the *shebang* line. *Please note: There will be a one (1) mark deduction from your achieved score if either or both of these items are absent*.
3. Immediately after your name and student number, provide a statement of no more than 150 words that outlines **how** you developed your script to solve for the stipulated task. Be sure to address the main commands, utilities and programmatic techniques you employed (making reference to the lecture you drew each from [1-8 inclusive]) and the role each serves in the context of the script's functionality. This statement is of course to take the form of comments. Please split the statement over multiple lines for easy readability.
4. To construct your script, use any combination of commands, utilities and programmatic techniques addressed in *Modules 1-8* (inclusive). *Be aware – marks will be lost if this stipulation is ignored. See rubric for details*.
5. Your script **must** contain concise and germane 'in-situ' comments that explain **all** of the code elements it contains. *Be aware – comments that are not germane to the code they describe, or a complete lack of comments, will not only cost marks, but may also be interpreted as suggestive of possible academic misconduct*.
6. The efficiency and correctness with which the commands, utilities and programmatic techniques within your script have been utilised will form also form part of your mark, so please pay close attention to this aspect of your code as well.

Important Note: *If your script does not run for **any** reason, e.g. broken/absent shebang line, hard-coding of files/directories/paths, use of a development environment not compatible with Ubuntu Desktop 18.0.4 LTS/GNU bash v5.0.17/ VS Code for Linux instance or higher, only a partial mark will be awarded on a code-readthrough basis. Non-functional scripts will **not** be fixed and you will **not** be permitted to resubmit. This is **not** negotiable.*

Sample Output Screenshots

Please note: Not all of the required functionality is demonstrated in the sample output screenshots below.



```
$ ./searchbin.sh -s helloworld
No matches found
$ ./searchbin.sh -s vmware
9 matches found...
NAME                    SIZE
vmware-checkvm          14.38kb
vmware-hgfsclient       14.38kb
vmware-namespace-cmd    22.30kb
vmware-rpctool          30.30kb
vmware-toolbox-cmd      46.59kb
vmware-vgauth-cmd       18.30kb
vmware-vgauth-smoketest 22.32kb
vmware-vmblock-fuse     22.72kb
vmware-xferlogs         14.38kb
```

```
$ ./searchbin.sh -b GT,24000000
No matches found
$ ./searchbin.sh -b GT,2500000
6 matches found...
NAME                    SIZE
git                     2.95Mb
perl                    3.32Mb
perl5.30.0              3.32Mb
python3.8               5.23Mb
snap                    22.52Mb
vim.basic               2.77Mb
$ ./searchbin.sh -b le,2
5 matches found...
NAME                    SIZE
atq                     2.00b
atrm                    2.00b
unxz                    2.00b
X11                     1.00b
xzcat                   2.00b
$ ./searchbin.sh -b EQ,7585
1 matches found...
NAME                    SIZE
zgrep                   7.41kb
```

```
$ ./searchbin.sh -l
151 matches found...
SYMBOLIC                POINTS TO
apport-collect          apport-bug
apropos                 whatis
apt-add-repository      add-apt-repository
atq                     at
atrm                    at
awk                     awk
btrfsck                 btrfs
byobu-screen            byobu
byobu-tmux              byobu
bzcmp                   bzdiff
bzegrep                 bzgrep
bzfgrep                 bzgrep
bzless                  bzmore
cal                     ncal
captoinfo               tic
chardet3                chardetect3
col2                    col1
col3                    col1
col4                    col1
col5                    col1
col6                    col1
```

Marking Key (Rubric)

Your assignment will be marked using an online rubric, the assessment criteria and weightings of which are outlined below:

ITEM#	ITEM	VALUE
1	Statement that outlines how script was developed to solve for stipulated task is logical, sensible, accurate and is fully reflective of the code provided	2
2	Script correctly implements **all** specifications listed in *Table 1: Script functionality requirements*, in a entirely error-free manner	4
3	When no flag/argument provided at run time, the script is to terminate with an appropriate error message	2
4	Script allows only one (1) option/argument(s) combination to be run at a time	2
5	All argument inputs are handled in a case-insensitive manner	2
6	All options and arguments passed at the command line are fully validated and error handled as stipulated	4
7	Display of results output to terminal formatted precisely as stipulated in all cases	4
8	When no matches returned for a search, user informed with appropriate message	2
9	Script code is properly structured at all stages	1
10	Script code is accurately and informatively commented at all stages	2
11	Script utilises its resident commands, utilities and programmatic techniques in a correct, efficient and completely error-free manner	2
12	Script **only** employs commands, utilities and programmatic techniques addressed in the unit's lecture slides (Modules 1-8 inclusive) *	3
	TOTAL	**/30**

* *Partial marks are not available for this criterion*

Search Binaries – Video (10 marks)

Record a ten (10) minute video providing a full explanation and demonstration of your Search Binaries script, both in-code and in-action.

- ☐ Begin with yourself appearing on-screen displaying your Student ID card and verbally stating your full name and student number. *If you do not have a Student ID card, a driver's license or valid passport details page is also be acceptable*. *Note: there will be a two (2) point deduction from your achieved score if this requirement is not met*.
- ☐ Demonstrate, i.e. run through **all** of your script's required functionality as listed in *Table 1: Script functionality requirements* **and** *Required Script Functionality* items 2-9 inclusive, using both valid and invalid inputs where applicable (to demonstrate validation and error handling). In the event you have not implemented one or more of the required functionalities, please state "*I have not implemented the [specific] functionality"* in each applicable case. You should allocate about 5-6 minutes for this demonstration.
- ☐ Referring to specific and relevant blocks of code in your script, explain how you implemented the required functionality using the bash/shell script commands, utilities and programmatic techniques covered throughout the unit (Modules 1-8 inclusive). You should allocate about 4-5 minutes for this explanation.
- ☐ Both the video and audio elements of your presentation must be of good quality. *Note: there will be a one (1) point deduction from your achieved score if there are quality issues with your video*.
- ☐ Your video must **not** be more than ten (10) minutes long. Note: Your lecturer will stop viewing your presentation at the ten (10) minute mark and anything you present thereafter will **not** factor into your grade. This is **not** negotiable.
- ☐ You **must** use the University's Panopto tool to record your Search Binaries demonstration/explanation video. Videos recorded/delivered in any other manner will **not** be viewed and will receive a score of zero.
- ☐ You **must** also make your lecturer an authorised viewer of your video (using their email address) in its settings in Panopto. Failure to do this may prevent your lecturer from viewing your video, and may result in a score of zero being allocated.

Marking Key (Rubric)

Your assignment will be marked using an online rubric, the assessment criteria and weightings of which are outlined below:

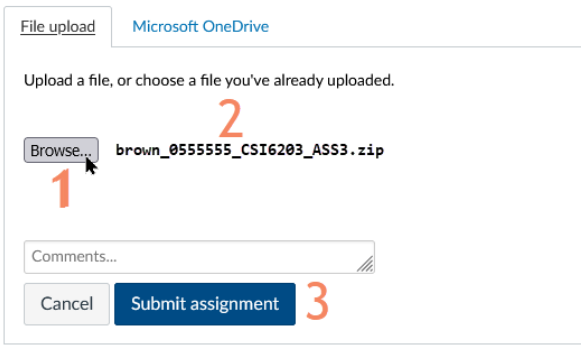
ITEM#	ITEM	VALUE
1	Demonstrated all of script's required functionality as listed in *Table 1: Script functionality requirements* and *Required Script Functionality items 2-9* inclusive, using **both** valid and invalid inputs where applicable	5
2	Referred to specific and relevant blocks of code in script, explaining **how** required functionality was implemented using the *bash/shell script commands*, *utilities* and *programmatic techniques* covered throughout the unit (Modules 1-8 inclusive)	5
	TOTAL	**/10**

How to submit your Assignment to Canvas

Script Component

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

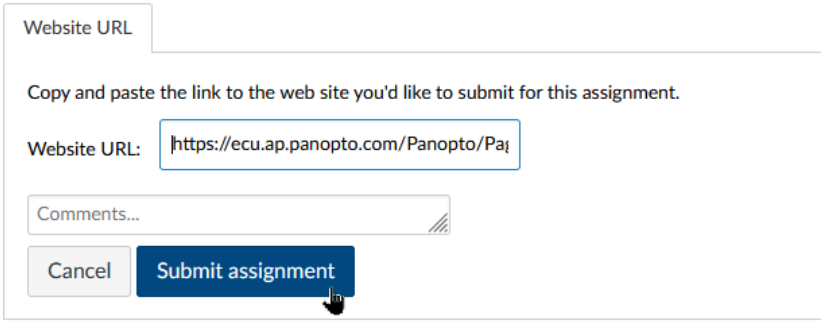
`[surname]_[student-ID]_CSI6203_ASS3.zip`



Do **not** submit any other files other than that stipulated above.

Video Component

From within your ECU student Panopto account, copy the link to your *Search Binaries* video, and then in Canvas, paste the link into the *Website URL* box that appears in the assignment's submission screen.



END OF ASSIGNMENT BRIEF