

Scripting Languages

Assignment 2.1: 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, students suspected of representing AI output as their own work may 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 **strongly** recommended that you develop and/or test your script in this environment. If you have chosen not to use this environment, it is **your** responsibility to ensure your script functions as expected within this environment before submission. The submission of incompatible scripts that do not operate as a student expects when graded is not uncommon, and this scenario often results in a substantial loss of marks, or in some cases, a zero (0) result.
- 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 in some cases.
- You must only submit a **single** shell script (.sh) file named *getimgs.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.

Task – Get Images (20 marks)

Using **only** the commands, utilities and programmatic techniques addressed in lecture slides 1-8 inclusive, write a script that downloads and reports upon all of a *user-specified image type*, e.g. png, jpg, jpeg, gif, at a *user-provided URL*.

Required Script Functionality

- The user is to be *prompted* for the desired **URL** and **image type**
- Your assessor will *always* enter a valid URL, so you don't need to validate for this
- Your script is to scan for and download images of the following four (4) types **only**:
 1. *.jpg*
 2. *.jpeg*
 3. *.gif*
 4. *.png*
- Keep in mind that URLs often contain multiple copies of the same image file, hyperlinked in different locations on the page users interact with, therefore, your code will need to account for this so that such duplicates, where they exist, are **neither** downloaded **nor** displayed in the final summary or any of its statistics
- URLs that contain **no** image files of the allowed types will also be used during the assessment process, so your script will need to allow for this scenario as well, informing the user of this with an appropriate terminal message when it occurs and exiting the script exited gracefully
- In the event that allowable image file(s) of the type the user stipulates *are* available at the URL provided, all of them must be downloaded from the URL and stored in a directory with a *unique* name that will not conflict with any other directories that currently exist or which may be created in the future
- It is the responsibility of the script to handle directory naming and creation, **not** the users'.
- Once all allowed image files have been downloaded from the URL provided (where they exist), the following information is to be displayed to the terminal in a neat and easy-to-read format:
 - A count of how many were downloaded (*excluding* duplicates, where they exist)
 - The directory name into which they were downloaded
 - A tabulated summary, with an appropriate header, of all of the image files downloaded (*excluding* duplicates, where they exist) and the size of each on disk in bytes, kilobytes or megabytes as applicable
- Your script is also to accommodate an *optional -z* option (to be used at the command line) that, when used, will facilitate the adding of all the downloaded image files to a zip archive that is to reside in the same directory as the downloaded image files and given the same name as this directory
- When the *-z* option is used, the user is to be informed that the requested zip archive has been created successfully and where it is located as part of the final summary output
- If the user provides an invalid flag, **or** any other invalid input at all, at the command line, they are to be informed of this with an appropriate error message and the script terminated with an appropriate exit code

Other Compulsory Requirements

- Call your script *getimgs.sh*
- Your full name and student number **must** be placed at the top of your script (as comments) immediately after the *shebang* line.
- Immediately after your name and student number, provide a statement of no more than 220 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 (from Modules 1-8 inclusive), the role each serves in the context of the script's functionality, and how all have been integrated to achieve the stipulated outcomes. This statement is of course to take the form of comments. Please split the statement over multiple lines for easy readability.
- To construct your script, use any combination of commands, utilities and programmatic techniques addressed in the unit's lecture slides (*Modules 1-8 inclusive*). *Do **not** use commands, utilities or programmatic techniques not addressed in the unit's lecture slides (Modules 1-8 inclusive).*
- Any temporary files your script creates in the course of its execution **must** be removed upon the script's termination. This, of course, does **not** include the downloaded image files, the directory in to which they were downloaded, or the zip archive created if the -z option is used.
- Your script **must** contain accurate and helpful 'in-situ' comments that explain **all** of the code elements it contains. *Be aware – comments that are not accurate and helpful in regard 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.*
- 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. For example, your *getimgs.sh* script is expected to make use of *at least one (1)* genuinely useful function, the getopts command (correctly applied), piping, command substitution, array(s) etc.
- Marks will also be deducted if the number of lines-of-code in your script *significantly* exceeds that your assessor deems were necessary to achieve the stipulated outcomes. The most common reasons for excessive lines-of code include, but are not limited to, poor command selection, non-use of functions for repetitive tasks, and limited use of piping and command substitution.

Test URLs

To develop and test your script, collect a set of URLs that each contain numerous images, preferably of varying types. The more URLs you test your script with, the more confident you can be that it is functioning as required. For example, web pages that display sets of product images are ideal for this purpose. You can easily examine what kinds of images files any URL contains by using the *View Source* option in your browser.

Important Note: *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 will be awarded on a code-readthrough basis (at the assessor's discretion). Your assessor will **not** fix non-functional, dysfunctional or incompatible scripts.*

Sample Output Screenshots

If your script has been properly coded and is functioning as expected, its outputs will very closely resemble the following screenshot examples:

Please note:

1. Most, but not all, of the script's required functionality is demonstrated in the screenshot examples below
2. Specific URLs and image file names have been blurred out to prevent students hard coding them into their own scripts

```
⊗ dromag@PTOP-40396376: ~/src/lang/lec/ass/ass2023/pt $ ./getimgs.sh -p
Invalid option error. Only -z for zip images is valid - exiting..
⊗ dromag@PTOP-40396376: ~/src/lang/lec/ass/ass2023/pt $ ./getimgs.sh hello
Invalid input error. Only -z for zip images is valid - exiting..
```

```
⊗ dromag@PTOP-40396376: ~/src/lang/lec/ass/ass2023/pt $ ./getimgs.sh
Enter a URL and image file type, e.g. http://somedomain.com jpg: https://www.unusualpetvets.com.au/vets-list/reptile/ wav
Unsupported image file type entered. Exiting...
⊗ dromag@PTOP-40396376: ~/src/lang/lec/ass/ass2023/pt $ ./getimgs.sh
Enter a URL and image file type, e.g. http://somedomain.com jpg: https://www.unusualpetvets.com.au/vets-list/reptile/
One or more arguments have not been provided. Exiting... missing argument - no file type
```

```
⊗ dromag@PTOP-40396376: ~/src/lang/lec/ass/ass2023/pt $ ./getimgs.sh
Enter a URL and image file type, e.g. http://somedomain.com jpg: https://www.unusualpetvets.com.au/vets-list/reptile/ jpg
Scanning for jpg files at https://www.unusualpetvets.com.au/vets-list/reptile/
60 jpg files detected at URL, which includes 1 duplicate(s)
Downloading unique jpg files
59 jpg files have been downloaded to the directory jpg_2023_08_10_09_07_33
Image Name | Size (Bytes)
blue-tongue-lizard-1.jpg | 57.90Kb
carpet-python-1.jpg | 110.20Kb
olive-python-care.jpg | 56.06Kb
reptiles-1-2023-08-10.jpg | 11.22Kb
reptiles-1-2023-08-10.jpg | 12.18Kb
reptiles-1-2023-08-10.jpg | 18.51Kb
reptiles-1-2023-08-10.jpg | 26.89Kb
reptiles-1-2023-08-10.jpg | 29.79Kb
reptiles-1-2023-08-10.jpg | 2.09Kb
reptiles-1-2023-08-10.jpg | 84.90Kb
reptiles-1-2023-08-10.jpg | 10.06Kb
```

```
⊗ dromag@PTOP-40396376: ~/src/lang/lec/ass/ass2023/pt $ ./getimgs.sh -z
Enter a URL and image file type, e.g. http://somedomain.com jpg: https://www.unusualpetvets.com.au/vets-list/reptile/ jpg
Scanning for jpg files at https://www.unusualpetvets.com.au/vets-list/reptile/
60 jpg files detected at URL, which includes 1 duplicate(s)
Downloading unique jpg files
59 jpg files have been downloaded to the directory jpg_2023_08_10_09_18_25
Image Name | Size (Bytes)
blue-tongue-lizard-1.jpg | 57.90Kb
carpet-python-1.jpg | 110.20Kb
olive-python-care.jpg | 56.06Kb
reptiles-1-2023-08-10.jpg | 11.22Kb
reptiles-1-2023-08-10.jpg | 12.18Kb
reptiles-1-2023-08-10.jpg | 18.51Kb
reptiles-1-2023-08-10.jpg | 26.89Kb
reptiles-1-2023-08-10.jpg | 29.79Kb
reptiles-1-2023-08-10.jpg | 2.09Kb
reptiles-1-2023-08-10.jpg | 84.90Kb
reptiles-1-2023-08-10.jpg | 10.06Kb
59 jpg files archived to jpg_2023_08_10_09_18_25.zip in the jpg_2023_08_10_09_18_25 directory.
```

```
• browser@PF201-43:29437% ~/src/lang/lec/ass/ass2023/pf2 $ ls
archive  getimgs.sh  jpg_2023_08_10_09_18_25
• browser@PF201-43:29437% ~/src/lang/lec/ass/ass2023/pf2 $ ls -lh jpg_2023_08_10_09_18_25/
total 1.3M
-rw-r--r-- 1 brown brown 58K Jun 20 12:34 blue-tongue-eating-king.jpg
-rw-r--r-- 1 brown brown 111K Feb 7 2023 target-python-king.jpg
-rw-r--r-- 1 brown brown 1.6M Aug 10 09:18 jpg_2023_08_10_09_18_25.zip
-rw-r--r-- 1 brown brown 57K Nov 22 2021 alive-python-care-sheet.jpg
-rw-r--r-- 1 brown brown 12K Nov 4 2019 reptiles-1-216x284.jpg
-rw-r--r-- 1 brown brown 13K Nov 4 2019 reptiles-1-216x284.png
```

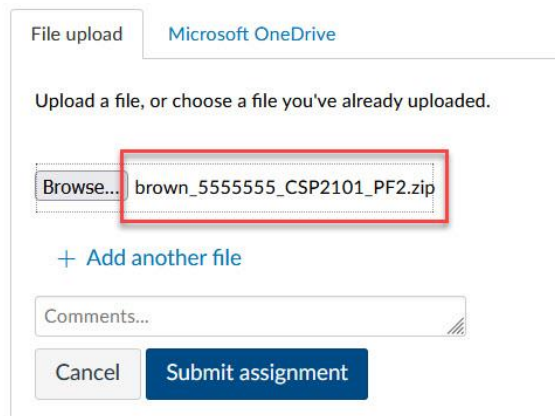
Assessment Rubric

On Canvas.

How to submit your portfolio to Canvas

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

[surname]_[student-ID]_CSxxxxx_PF2.zip



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