**University of Technology Sydney**
**Faculty of Engineering and Information Technology**

**Subject 32547 UNIX Systems Programming, Autumn 2024**

# Assignment

***Description***

This assignment is an individual programming assignment using Python. It addresses objectives 2 and 3 as listed in the Subject Outline document.

No limits apply to the number of lines of code of the program.

Assignments are **to be completed individually** (this might be checked with the use of anti-plagiarism tools such as Turnitin). **You should not receive help in the preparation of this assignment, nor ask anyone else to prepare it on your behalf, nor utilise generative AI tools such as GitHub Copilot or similar in any way**.

***Marks***

The assignment corresponds to 30% of the total marks.

***Submission***

The completed assignment is due by **5:00pm of Friday 17 May 2024**.

**PLEASE PAY ATTENTION TO THE FOLLOWING SUBMISSION INSTRUCTIONS:**

1. Your Python program has to be submitted in Canvas, following the "Assignments" menu and then the "Assignment" link by the due date. You can prepare your Python program anywhere you like, but probably the easiest option is to develop it in Ed STEM. You can submit your program multiple times if you like, and we will mark the last submission. Note that Canvas adds a small suffix to the filename ("-1", "-2" etc) for any submission after the first one, but it's perfectly fine, we'll remove the suffix ourselves before marking.
2. Please submit only your Python program, and nothing else (no data files).
3. Late submissions will be deducted one mark per day late; more than seven days late, the assignment will receive zero. Special considerations for a late submission must be arranged in advance with the Subject Coordinator.

***Academic Standards***

The assignments in this Subject should be your own original work. Any code taken from a textbook, journal, the Internet or any other source should be acknowledged. For referencing, please use the standard referencing conventions (http://www.lib.uts.edu.au/help/referencing).

***Marking Scheme***

| Mark Range | |
|---|---|
| 30 | All requirements of the assignment are met. The submitted program can be executed by the lecturer "as is" and produces the requested output. |
| 24-29 | The program works correctly with any valid file and for all options, but its execution experiences some minor problems. |
| 18-23 | The program does not work as expected with one of the options. |
| 0-17 | This range goes from no submission at all (0 marks) to a submission which does not meet major requirements of the assignment.<br>Examples:<br>• the program does not work as expected with two or more options;<br>• the program generates unhandled errors;<br>• the program does not solve the assignment problem. |

The assignments will be marked within two weeks from the submission deadline or as soon as possible.

**Important notes:**

- **Submission of this assignment is not compulsory to pass the subject**; do not feel that you have to submit it "at all costs" and perhaps be tempted to seek unauthorised assistance.
- There are no minimum requirements on the marks on this assignment to pass the subject.
- This assignment **must be your own work** and **you should not be helped to prepare it in any way**; your assignment may be tested with **anti-plagiarism software** that detects superficial changes such as changing variable names, swapping lines of code and the like.
- The subject coordinator **may** ask you to describe your assignment **in a "viva voce" in Zoom** to finalise your mark.
- Understanding the assignment specifications is part of the assignment itself and no further instructions will be provided; on the other hand, whatever is not constrained you can implement it according to your own best judgment.

***Title: Processing a "locale" file with Python***

In this assignment, you will write a Python program vaguely inspired by Unix command `locale`. Your Python program will parse a file containing information about language "locales" and "charmaps" and will generate output depending on the command line.

These are the specifications for your Python program:

1. It must be named *locale.py*

2. It should be invoked as:

   `python locale.py` *`option argument_file`*

   In the command line above, *`option`* means one of the options described below, and *`argument_file`* means the **chosen argument file**, which can have **any arbitrary name**.

   The program must check that argument *argument_file* exists, is a file and is readable. If not, it must print an error message to **the standard output** and exit. The specifications for the *argument_file* and *option* arguments are given below.

3. File *argument_file* can have any arbitrary name. It must be a file of text with the following format:

   a. The file consists of an arbitrary number of lines (including, possibly, zero lines).

   b. Each line must contain three fields separated by commas.

   c. The three fields are: *type*, *language*, *filename*.

   d. The *type* field can only have as value the literal strings `locale` or `charmap`.

   e. The *language* and *filename* fields are each a string of characters of arbitrary (yet reasonably limited) length. Acceptable characters include: lower and upper case letters, digits, underscore, dot.

**Fundamental note**: your program is not expected to verify that file *argument_file* complies with the above specifications. It will only be tested with compliant files.

The following example should be regarded as the reference specification for the format of file *argument_file*:

```
locale,English,en_AU
locale,French,fr_BE
charmap,English,EN
locale,English,en_US
charmap,Chinese,GBK
```

4. Your program can be invoked with option: -a. In this case, it must print the filenames of all the available locales, in this format:

```
Available locales:
filename of first locale in appearance order
filename of second locale in appearance order
…
filename of last locale in appearance order
```

Example with the example *argument_file* given above:

Command line:

```
python locale.py -a argument_file
```

Expected output:

```
Available locales:
en_AU
fr_BE
en_US
```

In the case in which file *argument_file* is empty or no available locales exist, your program must instead only print:

```
No locales available
```

5. Your program can be invoked with option: -m. In this case, it must print the filenames of all the available charmaps, in this format:

```
Available charmaps:
filename of first charmap in appearance order
filename of second charmap in appearance order
…
filename of last charmap in appearance order
```

Example with the example *argument_file* given above:

Command line:

```
python locale.py -m argument_file
```

Expected output:

```
Available charmaps:
EN
GBK
```

In the case in which file *argument_file* is empty or no available charmaps exist, your program must instead only print:

```
No charmaps available
```

6. Your program can be invoked with option: -l *language*. The *language* argument has the same format as the *language* field in the argument file. In this case, your program must search for the language in the argument file, and if it finds it, print the following information:

```
Language language:
Total number of locales: total number of locales in that language
(possibly 0)
Total number of charmaps: total number of charmaps in that language
(possibly 0)
```

Example with the example *argument_file* given above:

Command line:

```
python locale.py –l English argument_file
```

Expected output:

```
Language English:
Total number of locales: 2
Total number of charmaps: 1
```

Another example with the example *argument_file* given above:

Command line:

```
python locale.py –l Chinese argument_file
```

Expected output:

```
Language Chinese:
Total number of locales: 0
Total number of charmaps: 1
```

In the case in which language *language* is not present at all in *argument_file*, your program must print:

```
No locales or charmaps in this language
```

Example with file *argument_file* given above:

Command line:

```
python locale.py -l German argument_file
```

Expected output:

```
No locales or charmaps in this language
```

7. Your program can be invoked with option: -v. In this case, it must only print your name, surname, student ID and date of completion of your assignment, in a format of your choice. Please note that argument *argument_file* is still required.

8. No options can be used simultaneously. This means that your program can only be invoked with one of the options at a time.

9. If your program is invoked with any other syntax than those specified above, it must print a message of your choice **to the standard output** and exit.

   Examples of incorrect syntax:

   `python locale.py -Z argument_file` *(this option doesn't exist)*

   `python locale.py –a` *(it misses the argument file)*

   `python locale.py -l argument_file` *(it misses the language argument)*