University of Technology Sydney
Faculty of Engineering and Information Technology

Subject: 32547 UNIX Systems Programming, Autumn 2018

Assignment

Description

This assignment is an individual programming assignment using Perl. 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 in any way**.

Marks

The assignment corresponds to 30% of the total marks.

Submission

The completed assignment is due by **5:00pm of Friday 8 June 2018**.

PLEASE PAY CAREFUL ATTENTION TO THE FOLLOWING SUBMISSION INSTRUCTIONS:

1. The assignment has to be submitted electronically as a single ZIPPED file attached to an email to the Subject Co-ordinator, Massimo Piccardi (Massimo.Piccardi@uts.edu.au).
2. The email's subject must be: **32547 - assignment submission - <*yourStudentID*>**. Example: **32547 - assignment submission - 99887799**.

   Email without this subject or with a different subject may not be accepted as a valid submission. Submit the email **from your UTS email account**, otherwise the submission may get blocked or not be received.
3. The filename for the zipped archive attached to the email must be: **USP_<*yourSurname*>_<*yourStudentID*>.zip**.
   Example: **USP_Piccardi_99887799.zip**. Archives with a different filename may not be accepted as a valid submission.
4. Late assignments will be deducted one mark per day late, more than seven days late the assignment will receive zero. Special consideration, for late submission, must be arranged in advance with the Subject Co-ordinator.
5. If you would like a receipt, just please add a request for a Return Receipt to your email. No other types of receipts can be guaranteed.

If you do not follow the instructions above, your file may not be received or tracked correctly and thus may not be marked.

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. Examples: • the program does not work as expected with two or more options; • the program generates unhandled errors; • the program does not solve the assignment problem.

The assignments will be marked within two weeks from the submission deadline.
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.

Title: locale with Perl

In this assignment, you will write a Perl program inspired by Unix command `locale`. Your Perl program will parse a file containing information about local language environments and character maps and will generate output depending on the command line.

These are the specifications for your Perl program:

1. It must be named *locale.pl*

2. It must be invoked as:

   `locale.pl` *`option locale_info_file`*

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

3. File *locale_info_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 four fields separated by commas.

   c. The four fields are: *type, language*, *filename*, *size in bytes*.

   d. The *type* field can only have as value the 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.

   f. The *size in bytes* field is a positive integer number.

   The following is an example of file *locale_info_file*:

```
locale,English,en_AU,6251
locale,French,fr_BE,25477
charmap,English,EN,5423
locale,English,en_US,100000
charmap,Chinese,GBK,75443
```

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

4. Your program can be invoked with option: -a. In this case, it must print the following:

```
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 *locale_info_file* given above:

Command line:

```
locale.pl –a locale_info_file

Available locales:
en_AU
fr_BE
en_US
```

In the case in which file *locale_info_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 following:

```
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 *locale_info_file* given above:

Command line:

```
locale.pl –m locale_info_file

Available charmaps:
EN
GBK
```

In the case in which file *locale_info_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: -s. In this case, it must only print the following string:

```
Total size in bytes of all locales: <total size in bytes of all locales>
```

Example with the example *locale_info_file* given above:

Command line:

```
locale.pl –s locale_info_file
```

Output:

```
Total size in bytes of all locales: 131728
```

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

```
Total size in bytes of all locales: 0
```

7. Your program can be invoked with option: -l <language>. Argument <language> follows the same rules as the *language* field. In this case, it must print:

```
Language <name>:
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 *locale_info_file* given above:

Command line:

```
locale.pl -l English locale_info_file
```

Output:

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

Another example with the example *locale_info_file* given above:

Command line:

```
locale.pl -l Chinese locale_info_file
```

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 *locale_info_file*, your program must print:

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

8. Your program can be invoked with option: -v. In this case, it must only print your name, surname and student ID in a format of your choice.

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

10. If your program is invoked with a valid file argument, but any other syntax than what specified above, it must only print the following string to the standard output:

```
Invalid command syntax
```

Example:

Command line:

```
locale.pl -Z locale_info_file
```

Output:

```
Invalid command syntax
```

11. Zip your file *locale.pl* into a file named USP_<*yourSurname*>_<*yourStudentID*>.*zip* and submit it with the modalities specified above. Several free zip utilities are available on the WWW.

Please be reminded that:

- **this assignment must be your own work and you should not be helped by anyone to prepare it in any way; your assignment may be tested by anti-plagiarism software that detects superficial changes such as changing variable names, swapping lines of code and the like.**
- **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.**