University of Technology Sydney
Faculty of Engineering and Information Technology

Subject 32547 UNIX Systems Programming, Spring 2022

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 in any way**.

Marks

The assignment corresponds to **30% of the total marks**.

Submission

The completed assignment is due by **5:00pm of Friday 28 October 2022**.

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.
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. 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 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 by anyone 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 log file with Python

In this assignment, you will write a Python program that reads from a log file containing information about Unix processes (NB: a log file is just a text file storing the output of chosen commands or scripts). Your Python program will read the log file and produce the required output depending on its command-line option and argument. These are all the specifications for your Python program:

1. It must be named process_info.py

2. It must be called with the following command line:

   `python process_info.py` *`option log_file`*

   In the command line above, *`option`* means one of the options described below, and *`log_file`* means the name of the chosen log file.

   Your program must check that the *log_file* argument is an existing and readable file. If not, it must print a message of your choice **to the standard output** and exit. The format for file *log_file* and the values for the *option* argument are described hereafter.

3. File *log_file* can have any arbitrary name (including the extension). It must be a file of text with the following format:

   a. The file consists of an arbitrary, yet reasonable, number of lines (including, possibly, zero lines). Each line corresponds to a process.

   b. Each line must contain four fields separated by spaces.

   c. The four fields are: *process ID*, *memory size*, *CPU time* and *program name*.

   d. The *process ID* field is a string representing an integer between 0 and 32768.

   e. The *memory size* and *CPU time* fields are both strings representing integers between 0 and 99999999.

   f. The *program name* field is a string between 1 and 20 characters in length containing no spaces.

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

```
1 4730 1031782 init
4 0 6 events
2190 450 0 top
21413 5928 1 sshd
22355 1970 2009 find
```

Fundamental note: your program does not need to verify that file *log_file* complies with the specifications. You can simply assume that the file meets all the specifications given above.

4. Your program can be invoked with option: -a. In this case, it must print all the lines of file *log_file* sorted alphabetically by field *program name* (if two or more lines have the same value of field *program name*, they can be printed in any arbitrary order; in all cases, we will not test this case).

   Example with the example *log_file* given above:

   Command line:

   ```
   python process_info.py -a log_file
   ```

   Output:

   ```
   4 0 6 events
   22355 1970 2009 find
   1 4730 1031782 init
   21413 5928 1 sshd
   2190 450 0 top
   ```

   In the case in which file *log_file* is empty, your program will instead print:

   ```
   No processes found
   ```

5. Your program can be invoked with option: -m. In this case, it must print:

   ```
   Total memory size: <total memory size for all processes in log_file> KB
   ```

   In the above, *<total memory size for all processes in log_file>* must be replaced by the total memory size computed by adding up the values of field *memory size* for all lines.

   Example with the example *log_file* given above:

   Command line:

   ```
   python process_info.py -m log_file
   ```

   Output:

   ```
   Total memory size: 13078 KB
   ```

   In the case in which file *log_file* is empty, your program will instead print:

   ```
   No processes found
   ```

6. Your program can be invoked with option: -t. In this case, it must print:

   ```
   Total CPU time: <total CPU time for all processes in log_file> seconds
   ```

   In the above, *<total CPU time for all processes in log_file>* must be replaced by the total CPU time computed by adding up the values of field *CPU time* for all lines.

Example with the example *log_file* given above:

Command line:

```
python process_info.py -t log_file
```

Output:

```
Total CPU time: 1031798 seconds
```

In the case in which file *log_file* is empty, your program will instead print:

```
No processes found
```

7. Your program can be invoked with option: -s *memory threshold*. In this case, it must print all the lines of file *log_file* where the value of field *memory size* is greater than or equal to the value of *memory threshold*, in appearance order. The format of argument *memory threshold* is the same as that of field *memory size*.

   Example with the example *log_file* given above:

   Command line:

   ```
   python process_info.py -s 2000 log_file
   ```

   Output:

   ```
   1 4730 1031782 init
   21413 5928 1 sshd
   ```

   In the case in which no values of field *memory size* meet the requirement (including the case of an empty file), your program must instead print:

   ```
   No processes found with the specified memory size
   ```

8. 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 *log_file* is still required.

9. Please note that the options cannot be used simultaneously. This means that your program must only be invoked with one of the options at a time.

10. 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 process_info.py -z log_file` (i.e., wrong option)

    `python process_info.py -v` (i.e., missing the filename argument)

    `python process_info.py -s log_file` (i.e., missing the *memory threshold* argument)