University of Technology Sydney
Faculty of Engineering and Information Technology

Subject 32547 UNIX Systems Programming, Autumn 2021

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 Monday 24 May 2021**.

PLEASE PAY ATTENTION TO THE FOLLOWING SUBMISSION INSTRUCTIONS:

1. Your Python program has to be submitted on 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.

Title: Processing a faillog file with Python

In this assignment, you will write a Python program which is inspired by Linux command `faillog`. Your program will parse a text file containing usernames and dates/times which are supposed to represent failed login events, and will generate output according to the specifications below.

These are the specifications for your Python program:

1. It must be named *faillog.py*

2. It should be invoked as:

   `python faillog.py` *`option faillog_file`*

   The program must check that argument *faillog_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 *faillog_file* and *option* arguments are given below.

3. File *faillog_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 one space character.

   c. The three fields are: *username*, *date*, *time*.

   d. The *username* field is a string of variable length of maximum 32 characters. The characters allowed include lowercase letters, uppercase letters, digits, the underscore (_) and the hyphen (-).

   e. The *date* field is a string of characters representing a date in dd/mm/yyyy format (that is, day, month, year; examples: 01/02/2012; 31/12/2019; 17/09/1991).

   f. The *time* field is a string of characters representing a time in hh:mm:ss format (that is, hours, minutes, seconds; examples: 00:00:00 <- the earliest time in a day; 05:00:00 <- 5AM; 23:00:00 <- 11PM; 09:23:59; 12:05:08; 23:59:59 <- the latest time in a day).

   Fundamental note: your program is not expected to verify that file *faillog_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 *faillog_file*:

```
vanessa 31/12/2019 23:59:59
User_994 01/01/2020 00:00:00
massimo 21/04/2021 10:59:30
torvalds 17/09/1991 00:00:01
massimo 21/04/2021 10:59:33
User_17673 21/04/2021 11:13:25
torvalds 05/10/1991 00:00:02
```

4. Your program can be invoked with option: -a. In this case, it must print all the usernames, **only once**, **in the order in which they first appear in file *faillog_file***, in this format:

```
Usernames:
first username
second username
…
last username
```

Example with file *faillog_file* given above:

Command line:

```
python faillog.py -a faillog_file
```

Output:

```
Usernames:
vanessa
User_994
massimo
torvalds
User_17673
```

In the case in which file *faillog_file* is empty, your program must instead only print:

```
No usernames found
```

5. Your program can be invoked with option: -u *username*. The *username* argument must be entered in the same format as the corresponding field.

   In this case, your program must print all the lines of file *faillog_file* where the *username* field is **the same** as the *username* argument, in the order in which they appear in the file, preceded by this line:

   ```
   Events for user: username
   ```

   This is an example with file *faillog_file* given above:

Command line:

```
python faillog.py -u torvalds faillog_file
```

Output:

```
Events for user: torvalds
torvalds 17/09/1991 00:00:01
torvalds 05/10/1991 00:00:02
```

In the case in which no line of file *faillog_file* matches the request (including the case where the file is empty), your program must print:

```
No events found for the given username
```

6. Your program can be invoked with option: -t *date time*. The *date* and *time* arguments must be entered in the same formats as the corresponding fields.

   In this case, your program must print all the lines of file *faillog_file* where the *date* and *time* fields are **later or equal** to the *date* and *time* arguments, in the order in which they appear in the file. This is an example with file *faillog_file* given above:

   Command line:

   ```
   python faillog.py -t 01/01/2020 00:00:00 faillog_file
   ```

   Output:

   ```
   User_994 01/01/2020 00:00:00
   massimo 21/04/2021 10:59:30
   massimo 21/04/2021 10:59:33
   User_17673 21/04/2021 11:13:25
   ```

   In the case in which no line of file *faillog_file* matches the request (including the case where the file is empty), your program must print:

   ```
   No events found for the given date/time
   ```

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 *faillog_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 faillog.py -Z faillog_file` (i.e., wrong option)

`python faillog.py -a` (i.e., missing argument file)

`python faillog.py -t 01/01/2020` (i.e., missing *time* argument)

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.**