University of Technology Sydney
Faculty of Engineering and Information Technology

Subject 32547 UNIX Systems Programming, Spring 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 ChatGPT, 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 1 November 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. 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 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 an interview 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: pkginfo with Python

In this assignment, you will write a Python program loosely inspired by Unix command `pkginfo` (a command available in a few versions of Unix). Your Python program will read a file containing information about the installed software packages and will generate output depending on the command line.

These are the specifications for your Python program:

1. It must be named *pkginfo.py*

2. It must be invoked as:

   `python pkginfo.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 four fields **separated by commas**.

   c. The four fields are: *category*, *name*, *description*, *size in kilobytes*.

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

   e. The *description* field is a string of characters of arbitrary (yet reasonably limited) length. Acceptable characters include: lower and upper case letters, digits, underscore, dot, '+', '/', '-', space.

   f. The *size in kilobytes* field is an integer limited between 1 and 10000000.

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*:

```
system,SUNWdoc,Documentation Tools,1251
application,SPROcpl,C++ Compiler,25477
system,BRCMbnxe,Broadcom NIC Driver,5423
newcat,madeup,a made up line,100000
application,ecj,Eclipse JDT,75443
```

4. Your program can be invoked with option: -a. In this case, it must print the names of the installed packages in the order in which they appear in the argument file, in this format:

```
Installed packages:
first name in appearance order
second name in appearance order
…
last name in appearance order
```

Example with the example *argument_file* given above:

Command line:

```
python pkginfo.py -a argument_file
```

Expected output:

```
Installed packages:
SUNWdoc
SPROcpl
BRCMbnxe
madeup
ecj
```

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

```
No packages installed
```

5. Your program can be invoked with option: -s. In this case, it must print the total size in kilobytes of all the installed packages, in this format:

```
Total size in kilobytes: total size in kilobytes of all the installed
packages
```

Example with the example *argument_file* given above:

Command line:

```
python pkginfo.py –s argument_file
```

Expected output:

```
Total size in kilobytes: 207594
```

In the case in which file *argument_file* is empty, your program must print:

```
Total size in kilobytes: 0
```

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

```
Package: name
Category: category
Description: description
Size in kilobytes: size in kilobytes
```

Example with the example *argument_file* given above:

Command line:

```
python pkginfo.py –l ecj argument_file
```

Expected output:

```
Package: ecj
Category: application
Description: Eclipse JDT
Size in kilobytes: 75443
```

In the case in which a package with name *name* is not present in *argument_file*, your program must print:

```
No installed package with this name
```

Example with the example *argument_file* given above:

Command line:

```
python pkginfo.py –l not_there argument_file
```

Expected output:

```
No installed package with this name
```

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

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

   `python pkginfo.py -l argument_file` *(it misses an argument)*