Assignment 1

Assignment 1 description

INFO1112 - 2022 - Assignment 1

Due: Week 4, Thursday, 25th of August at 23:59 (Sydney local time)

This assignment is worth of your overall grade for the course.

Assessment

The assignment will be marked with an automatic testing system on Ed. A mark will be given based on a percentage of tests passed (5%) and a manual mark will be given for overall style, quality, readability, etc (3%). You are expected to write your own tests and submit them with your code, and a mark will be given based on coverage and manual inspection of your tests (2%).

There will be public test cases made available for you to test against, but there will also be extra non-public tests used for marking.Success with the public tests doesn't guarantee your program will pass the private tests.

Introduction

This assignment involves developing a program to check the access and permissions to files and change their permissions.

Your program will read a text file that specifies what file paths to check. Then check the access for each file path and test readability and executability of them. You will need to get the size, permissions, ownership, last modified, and last accessed date of all of the files. You will also need to change the permissions of the file path as instructed below.

Structure of the program

Your program will have a file called that is the main body of your work. This file needs to access the attributes of the file paths and also needs to flip the permission for for the file paths. Please refer to the "File Ownership and Permission" section in this document to learn more about file permissions and Owners.

and

The list of file paths that need to be processed for is in . This file contains a couple of lines. Each line shows a file path name to be processed.

This is the format of the file:

```
filepath1
filepath2
...
```

(You should change these file path names, based on what you add to the space.)

This means for : first, check its access and read its permissions and attributes, and write them in the output file.

Then change the permission of the file for accordingly:

- If the current permission is "read, write, -" or "read, write, not execute" permission, change it to "read, write, execute".
- If the current permission is "read, write, execute" permission, change it to "read, write, -" or "read, write, not execute".

Output File:

Your program will create an output file that saves all the information from processed file paths. The output file is named . It has several lines of text, each line shows the access and current permissions to each file path in the .

The format for the output file is:

filepath current-access file-properties

or

filepath **error**

- *filepath* is the file path that has been shown in

- *current-access* shows the accessibility of the file (before flipping the permission): existence, readability and executability. The format that you are writing this output is as follows:

  "*Group Readable: {True/False}, Group Executable: {True/False}*"

- *file-properties* shows the following attributes for each file: size, ownership, last modified, and last accessed date. The format that you are using to write these attributes in the output file is as follows:

  "*Size: {}, Owner: {}, Group: {}, last modified date: {}, last access date: {}*"

Error handling

Your program should do error checking. In particular,

- if cannot be found you should write to output file and terminate.
- if any of the file paths cannot be found you should write to output file and move to the next file in the list.

Your programs should never crash. If an error occurs a suitable message or status is produced. All other error messages than the above should be reported and sent to .

An experienced programmer checks for as many error conditions as they can. Thinking that "this can never happen therefore I won't check for it" is a bad idea.

An example of the output file

Here is an example of the output file when has only one line , and there is no error:

```
file2.ext Group Readable: True, Group Executable: True Size: 10, Owner: NazaninBorhan, Group: staff
```

Note that if the program is run again, then the "Group Executable" should be , since the program should flip that permission.

Here is an example of the output file , when has only one line

, and the file path can not be found:

```
filex.ext can not be found
```

Here is an example of the output file when the can not be found:

```
filelist.txt can not be found
```

File Ownership and Permission

You will need to understand file ownership and permissions for this assessment. In Unix operating systems every file and directory has the following three permissions:

- **Read:** This permission give you the authority to open and read a file. Read permission on a directory gives you the ability to list its content.
- **Write:** The write permission gives you the authority to modify the contents of a file. The write permission on a directory gives you the authority to add, remove and rename files stored in the directory. Consider a scenario where you have to write permission on a file but do not have to write permission on the directory where the file is stored. You will be able to modify the file contents. But you will not be able to rename, move or remove the file from the directory.
- **Execute:** In Windows, an executable program usually has an extension which you can easily run. In Unix/Linux, you cannot run a program unless the execute permission is set. If the execute permission is not set, you might still be able to see/modify the program code (provided read & write permissions are set), but not run it.

Every file and directory in Unix operating system has the following three types of users: , (if applicable, exclude), and (everyone else).

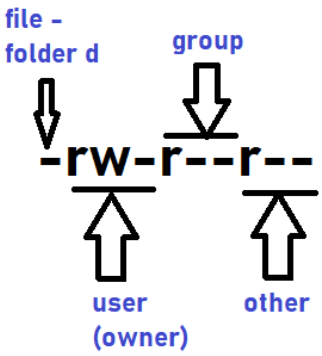
- A (owner) is the one who created the file, by default, who creates the file becomes the owner of the file. A can create, delete, or modify the file.
- A can contain multiple users, all the users belonging to a group have the same access permission for a file.
- Anyone who has access to the file other than and comes in the category of . has neither created the file nor is a member of .

All the three owners in the Unix system have three types of permissions characters assigned to them: : **Read**, : **Write,** and for **Execute.** ***Which makes the nine characters that represent the file permissions (- means no permission).***

Check this example in the bash terminal when running :

```
Projects — -zsh — 80×24
NazaninBorhan@AC02H10BZQ05P Projects % ls -l
total 8
-rw-r--r--  1 NazaninBorhan  staff   70 12 Jul 16:07 first.py
drwxr-xr-x  5 NazaninBorhan  staff  160 12 Jul 15:41 firsttest
```

The first line is a reference to the file called , that has this permission: .



That means has Read and Write permissions for user: and Read permission for group and Read permission of . The first only shows that is a file and not a folder, otherwise, it would be .

Changing Permissions With

In Unix, you can change the permissions with command accordingly to your need. A good reference to learn more about the command is the page of it.

Syntax:

▶ Expand

Here are some examples to change the permissions for different types of users.

For assigning permissions:

-

And for removing permissions:

-

Examples:

1) Assigning execute permission to for user , where refers to .

```
chmod u+x file2.ext
```

2) Removing execute permission from for , where refers to :

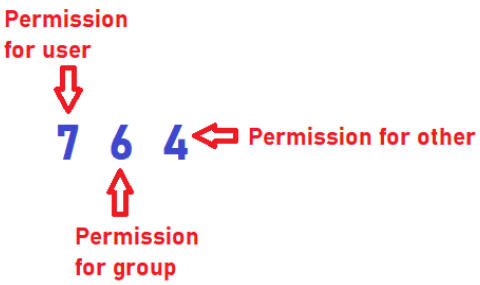
Numeric format to change permissions with

The numeric format to change the file permission with is very popular too. In this mode, file permissions are not represented as characters but as a three-digit octal number. The table below gives numbers for all permissions types:

Number	Permission Type	Symbol
0	No Permission	—
1	Execute	–x
2	Write	-w-
3	Execute + Write	-wx
4	Read	r–
5	Read + Execute	r-x
6	Read +Write	rw-
7	Read + Write +Execute	rwx

In the next example I am running this command:

Here is how you can analyze this number:



```
[NazaninBorhan@AC02H10BZQ05P Projects % chmod 764 first.py
[NazaninBorhan@AC02H10BZQ05P Projects % ls -l first.py
 -rwxrw-r--  1 NazaninBorhan  staff  70 12 Jul 16:07 first.py
```

This means now has Read and Write and Execute permissions for user , Read and Write permission for Group and Read permission only for .

Useful links

There is a lot more to learn here and a lot of resources are available on the internet about file ownerships and permissions in Unix file systems, but you can start with the following links:

https://en.wikipedia.org/wiki/File-system_permissions

https://www.guru99.com/file-permissions.html

https://en.wikipedia.org/wiki/Chmod

https://linux.die.net/man/1/chmod

https://man7.org/linux/man-pages/man1/chmod.1p.html

Please note, that further clarifications to this specification may be posted on Ed, and new editions of the specification will be published in Ed Lessons.

Implementation

The assignment is to be implemented in a Python program. A scaffold file is provided. You are expected to write legible code following the PEP-8 style guide.

What is described above in the "Changing Permissions With " section is a way of changing the permission of the file paths (You can run this in the terminal). However, you are using Python to change the permission of the files. That means you can not use the command **directly** to change the permissions of the file in your Python program. There are Pythonic ways to change permissions of the files, and you are expected to utilize them.

The following functions in Python are not allowed, as they can be used to call or other Unix tool in a y way:

- os.system()
- os.popen()

- os.fork() and os.exec*()

Testing

You are expected to write a number of test cases for your program. These should be simple input/output tests. You are expected to test every execution path of your code. Your test files should be located in a folder named . You may assume that the program will be presented in the folder when the test suit is examined.

Namely, the folder should have the following structure:

- The input of the program .
- The expected output file of the program and you should work this file out manually.
- Other regular files that are recorded by . Those files should have various permissions in order to test more execution paths.

You will also need to test and show that your program can handle errors for incorrect input.

You can write a simple testing script in Bash ( ) to run your input/output tests to simplify and automate the testing process, however, this file is optional and does not contribute to your marks.

Friendly note

Sometimes we find typos or other errors in specifications. Sometimes the specification could be clearer. Students and tutors often make great suggestions for improving the specification. Therefore, this assignment specification may be clarified up to **Week 3, Tuesday, August 16th**. No major changes will be made. Revised versions will be clearly marked and the most recent version will be in here.

Submitting your code

An Ed assessment workspace will be available for you to submit your code.

You should submit one file and a folder containing at least two files and . Optionally, is a shell script that helps run your tests. The folder should also contain files that are listed in .

Some test cases will be released, but there will be a set of unreleased test cases which will be run

against your code after the due date.

Any attempt to deceive the marking system will be subject to academic integrity proceedings.

Academic Declaration

By submitting this assignment, you declare the following:

I declare that I have read and understood the University of Sydney Student Plagiarism: Coursework Policy and Procedure, and except where specifically acknowledged, the work contained in this assignment/project is my own work and has not been copied from other sources or been previously submitted for award or assessment.

I understand that failure to comply with the Student Plagiarism: Coursework Policy and Procedure can lead to severe penalties as outlined under Chapter 8 of the University of Sydney By-Law 1999 (as amended). These penalties may be imposed in cases where any significant portion of my submitted work has been copied without proper acknowledgment from other sources, including published works, the Internet, existing programs, the work of other students, or work previously submitted for other awards or assessments.

I realize that I may be asked to identify those portions of the work contributed by me and required to demonstrate my knowledge of the relevant material by answering oral questions or by undertaking supplementary work, either written or in the laboratory, in order to arrive at the final assessment mark.

I acknowledge that the School of Computer Science, in assessing this assignment, may reproduce it entirely, may provide a copy to another member of faculty, and/or communicate a copy of this assignment to a plagiarism checking service or in-house computer program, and that a copy of the assignment may be maintained by the service or the School of Computer Science for the purpose of future plagiarism checking.

Changelog

All notable changes to this project will be documented in this file.

`[1.0] 2022-08-08 00:00 AEST`

- Release.

`[1.1] 2022-08-08 20:00 AEST`

- Update **Error handling**.

`[1.1.1] 2022-08-09 01:30 AEST`

- Update examples and test cases to align with the update above.
- Remarking last 10 submissions.

`[1.1.2] 2022-08-09 15:10 AEST`

- Refine update 1.1 wording.

`[1.1.3] 2022-08-11 10:33 AEST`

- Refine the **Structure of the program**

`[1.1.4] 2022-08-11 14:30 AEST`

- Update test script to detect non-empty and .
- Remarking last 10 submissions.

Assessment 1 - File access control

This is where you write your program. Please refer to "Assignment 1 description" to get the description of the assessment.