COMPSCI 340
Operating Systems
Assignment 2 - User space file system
Worth 10%
Due date: 09:30pm 22nd Sept 2023
Total - 30 marks

Introduction

In week 6, we introduced the FUSE (file system in user space) library. This assignment requires you to first work with an existing user space file system, and then subsequently create your own.

Setup

1. Environment:

Do the assignment either on Ubuntu in the labs or on your own machine (using virtual machines, macOS, or Windows Subsystem for Linux version 2) or use FlexIT (flexit.auckland.ac.nz). Search for the "Ubuntu Linux 20.04" desktop icon after sign-in.
Note: The markers will grade using FlexIT Ubuntu image.

2. Files:

Download fuse.py, passthrough.py, memory.py, and a2fuse1.py from Canvas -> Assignment 2 page.
These files have originally come from -

- fuse.py It offers Python bindings for the FUSE filesystem, enabling developers to create custom filesystems in Python without altering kernel code.
- memory.py It is a simple in-memory filesystem that carries out various filesystem operations without persisting any data to the actual disk.
- passthrough.py A passthrough filesystem would allow you to mount a directory to another location in your file system and all operations you perform in the mounted location get transparently passed through to the original directory.

Part 1: Familiarization with Existing User Space File System

1. Create two directories: **source** and **mount**. Place files **one**, **two**, **three** and **four** (from Canvas) in the **source** directory.
2. You will need two terminal windows open. **Terminal One** (to run the user space file system) and **Terminal Two** (for file operations).
3. In Terminal One, run: `python3 a2fuse1.py source mount`
4. In Terminal Two, do:

```
ls -l source
ls -l mount
```

Question 1: [2 marks]
Explain the Terminal Two output. What did you see and why was it like that?

Question 2: [4 marks]
In Terminal Two (within the 'mount' directory - perform "`cd mount`" first), execute:

```
cat > compsci340.txt
my compsci340 2023 assignment
^D
```
(this is control-D)

COPY the output generated by the user space file system in Terminal One and **EXPLAIN** each method called. You can get some information from the Python documentation and using *man*.

Example:

```
DEBUG:fuse.log-mixin:-> getattr /compsci340.txt (4,)
DEBUG:fuse.log-mixin:<- getattr {'st_atime': 1692772400.0, 'st_ctime': 1692772400.0, 'st_gid': 62215, 'st_mode': 33188, 'st_mtime': 1692772400.0, 'st_nlink': 1, 'st_size': 0, 'st_uid': 2486084}
```

It retrieves the attributes (getattr) of the file compsci340.txt. You can explain what the attributes mean.

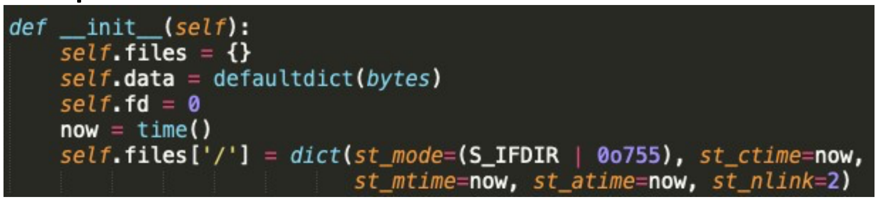
After tasks, shut down the user space file system using **fusermount -u mount** (Ubuntu) or **umount mount** (macOS) from the main directory (perform "`cd ..`" if still in mount directory). Verify the contents of both **source** and **mount** directories.

Part 2: Understanding Operations

The **Operations** class in **fuse.py** is the one which does the work we are interested in. Both **passthrough.py** and **memory.py** subclass it. While **passthrough.py** provides a mirrored directory (i.e., provides a copy of one directory mounted in a different location and passes all requests back to the original directory); **memory.py** offers a separate in-memory file system (i.e., when the file system is shut down those files are lost).

Question 3: [6 marks]
For each listed method in the Memory class (memory.py), provide a detailed (statement by statement) explanation of what exactly each method does:
`_init, getattr, readdir, open, create, unlink, write, read`

Example:



```
def __init__(self):
    self.files = {}
    self.data = defaultdict(bytes)
    self.fd = 0
    now = time()
    self.files['/'] = dict(st_mode=(S_IFDIR | 0o755), st_ctime=now,
                           st_mtime=now, st_atime=now, st_nlink=2)
```

Creates an empty dictionary self.files for the files. This will use the path names as the keys. Each value in the dictionary will be another dictionary.
self.data is a dictionary for the files' data. The path names are the keys. The values are the data of that file.
Sets the starting value for the file descriptors, these are going to be used as unique file identifiers. Grabs the current time and sets the file attributes for the root of this file system. It is a directory, with creation, modified and accessed times set to now. It has two links.

Part 3: Implementing the Hybrid File System

Now, create your own user space file system named **a2fuse2.py**. It works like a mix of both passthrough.py and memory.py. You can subclass Passthrough or Memory if you want. You will probably have to implement at least the same methods you described in Question 3 in this part.

1. **Execution:**
   - Command: **python a2fuse2.py source1 source2 mount**
   - Create two source directories, **source1** and **source2**. Put file one, two in source1 and put file three, four in source2 respectively. Make sure **mount** is initially empty.
   - The file system works very much like memory.py but it starts with some real files from the source1 and source2 directories. So, the file system has two classes of files which are in the mount directory. One consists of real files from the source1 and source2 directories and the other of files which only exist in memory.

2. **Requirements:**
   Specifically, create your FUSE to meet the following two requirements.
   - Enable the FUSE system to mount two source directories into a single mount point.
   - Distinguish files originating from **source1** and **source2** from newly created ones in **mount**. Newly created files exist only in memory. However, if a file came from the source directories, any modifications get passed back as with **passthrough.py**.

3. **Working** [12 marks]
   - The file system should work correctly with cat, ls, rm on a variety of files. Files in the source1 and source2 directories can be modified but any new files created only exist in the mount directory (in memory).

Example:

Start your FUSE in Terminal One: `python3 a2fuse2.py source1 source2 mount`
Start in the same directory in Terminal Two.

```
ssin820@D2P1RB1Ubu203:~/Desktop/A2_2023$ ls -l source1
total 2
-rw-r--r-- 1 ssin820 all 23 Aug 23 20:10 one
-rw-r--r-- 1 ssin820 all 24 Aug 23 20:10 two
ssin820@D2P1RB1Ubu203:~/Desktop/A2_2023$ ls -l source2
total 2
-rw-r--r-- 1 ssin820 all 24 Aug 23 20:09 four
-rw-r--r-- 1 ssin820 all 23 Aug 23 20:09 three
ssin820@D2P1RB1Ubu203:~/Desktop/A2_2023$ ls -l mount
total 0
-rw-r--r-- 1 ssin820 all 24 Aug 23 20:09 four
-rw-r--r-- 1 ssin820 all 23 Aug 23 20:10 one
-rw-r--r-- 1 ssin820 all 23 Aug 23 20:09 three
-rw-r--r-- 1 ssin820 all 24 Aug 23 20:10 two
ssin820@D2P1RB1Ubu203:~/Desktop/A2_2023$ cd mount
ssin820@D2P1RB1Ubu203:~/Desktop/A2_2023/mount$ cat > five
this is my fifth file.
ssin820@D2P1RB1Ubu203:~/Desktop/A2_2023/mount$ ls -l
total 0
```

```
-rw-r--r-- 1 ssin820 all 23 Aug 23 20:12 five
-rw-r--r-- 1 ssin820 all 24 Aug 23 20:09 four
-rw-r--r-- 1 ssin820 all 23 Aug 23 20:10 one
-rw-r--r-- 1 ssin820 all 23 Aug 23 20:09 three
-rw-r--r-- 1 ssin820 all 24 Aug 23 20:10 two
ssin820@D2P1RB1Ubu203:~/Desktop/A2_2023/mount$ cat one two > three
ssin820@D2P1RB1Ubu203:~/Desktop/A2_2023/mount$ cd ..
ssin820@D2P1RB1Ubu203:~/Desktop/A2_2023$ ls -l mount
total 0
-rw-r--r-- 1 ssin820 all 23 Aug 23 20:12 five
-rw-r--r-- 1 ssin820 all 24 Aug 23 20:09 four
-rw-r--r-- 1 ssin820 all 23 Aug 23 20:10 one
-rw-r--r-- 1 ssin820 all 47 Aug 23 20:13 three
-rw-r--r-- 1 ssin820 all 24 Aug 23 20:10 two
ssin820@D2P1RB1Ubu203:~/Desktop/A2_2023$ ls -l source1
total 2
-rw-r--r-- 1 ssin820 all 23 Aug 23 20:10 one
-rw-r--r-- 1 ssin820 all 24 Aug 23 20:10 two
ssin820@D2P1RB1Ubu203:~/Desktop/A2_2023$ ls -l source2
total 2
-rw-r--r-- 1 ssin820 all 24 Aug 23 20:09 four
-rw-r--r-- 1 ssin820 all 47 Aug 23 20:13 three
ssin820@D2P1RB1Ubu203:~/Desktop/A2_2023$ cat source2/three
this is my first file.
this is my second file.
ssin820@D2P1RB1Ubu203:~/Desktop/A2_2023$ fusermount -u mount
ssin820@D2P1RB1Ubu203:~/Desktop/A2_2023$ ls -l mount
total 0
ssin820@D2P1RB1Ubu203:~/Desktop/A2_2023$ ls -l source1
total 2
-rw-r--r-- 1 ssin820 all 23 Aug 23 20:10 one
-rw-r--r-- 1 ssin820 all 24 Aug 23 20:10 two
ssin820@D2P1RB1Ubu203:~/Desktop/A2_2023$ ls -l source2
total 2
-rw-r--r-- 1 ssin820 all 24 Aug 23 20:09 four
-rw-r--r-- 1 ssin820 all 47 Aug 23 20:13 three
ssin820@D2P1RB1Ubu203:~/Desktop/A2_2023$ cat source2/three
this is my first file.
this is my second file.
```

Question 4 [2 marks]
Try appending data to a file from source1 within the mount directory using the command `echo "some new data" >> filename`. Describe the output and which methods get called in terminal one.

Question 5 [2 marks]
Delete a file from source2 inside the mount directory. Is it also removed from the actual source2 directory on the disk?

Submission

Use the Canvas submission system to submit your assignment. Answer the given questions in a file named A2.txt, and zip together A2.txt and a2fuse2.py and submit.

Bonus Points [2 marks]

- 1 mark: Include your name and login in both files.
- 1 mark: Ensure files created by your file system have the correct user and group IDs.

Hints:

1. You can utilize logging for debugging: **logging.debug("your message")**, as long as you model your code on that in a2fuse1.py. This will then appear as output in Terminal One.
2. To make this assignment easier it only tests positively. i.e., any command executed by the markers will only be ones that should execute without causing an error. You do not need to worry about non-existent files, privileges, symbolic links, sparse files, or nested directories.

⚠ Sharing assignment solutions and source code does not help learning. Consequently, our academic integrity policy does not permit sharing of solutions or source code leading to solutions, nor does it allow sourcing solutions or source code from any third party. Violation of this will result in your assignment submission attracting no marks, and you may face further disciplinary action. Therefore, please do not share assignments, assignment solutions and/or source code leading to assignment solutions, or use material from others in your assignments. You must not publish assignments or solutions in any form online at any time. You will be liable if someone copies your solution. There are also copyright and IP issues. Please come talk to us if you have any doubt over what is legitimate and what is not.
You can refer to online tutorials, generative-AI tools (such as ChatGPT) and online resources. However, please learn from them and implement the solutions yourself based on what you learnt from these sources. You must not copy from these sources. Don't leave your computers, devices, and belongings unattended — you must secure these at all times to prevent anyone having access to your assignments or solutions. If others are found to have used your solution, you will also face disciplinary action.