COMPSCI 340 / SOFTENG 370
Operating Systems

Assignment 2 - User space file system

Worth 7%

final date 11:59pm 2nd of October, 2017

Introduction

These are the specifications for the assignment. This is slightly different from the slides in lecture 15. Also note the final date. There are lots of assignments due around that time. Please start working on this now so that you have plenty of time to ask questions.

In lecture 15 the FUSE (file system in user space) library was introduced and explained. In this assignment you have to first of all work with an existing user space file system. Then you need to write your own.

Part 1

Do the assignment either on Ubuntu in the labs or on your own machine (virtual machines work too, but not the Windows subsystem for Linux). The markers will use the lab image.

Download the files `fuse.py`, `passthrough.py`, and `a2fuse1.py` from the A2 files section of Canvas into a directory.

`fuse.py` originally came from https://github.com/terencehonles/fusepy

`passthrough.py` originally came from https://github.com/skorokithakis/python-fuse-sample

Make two directories, one called `source` and one called `mount` in the directory. Put some files in the `source` directory (download the files `oneten`, `onethousand`, `twothousand`, and `hundredthousand` from Canvas).

You will need two terminal windows open: one to run the user space file system and display the work it is doing, and one to work with files from the command line. I will refer to these as terminal one and terminal two.

In terminal one run the program: `python a2fuse1.py source mount`

In terminal two do:

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

For all of the questions put the answers (and requested output) into a file called `A2.txt`.

Question 1

Explain the output you have just seen in terminal two. What did you see and why was it like that?

[2 marks]

Question 2

For each of the following commands you perform in terminal two, copy the output generated by the user space file system in terminal one into your answer file and explain each method called. You can get some information from the Python documentation and more using `man`.

I have done the first one for you.

```
cd mount
```

```
DEBUG:fuse.log-mixin:-> getattr / (None,)
DEBUG:fuse.log-mixin:<- getattr {'st_ctime': 1504480206.8870952,
'st_mtime': 1504480206.8870952, 'st_nlink': 2, 'st_mode': 16877,
'st_size': 4096, 'st_gid': 1000, 'st_uid': 1000, 'st_atime':
1504480217.6225388}
DEBUG:fuse.log-mixin:-> access / (1,)
DEBUG:fuse.log-mixin:<- access None
```

`getattr / (None,)` - gets the file attributes associated with `/` which is the `mount` directory. The output is a dictionary. `st_ctime` is the creation time, `st_mtime` is the modified time, `st_nlink` is the number of hard links, `st_mode` is the file access mode, `st_size` is the size in bytes, `st_gid` is the group id, `st_uid` is the user id, `st_atime` is the last accessed time.

`access / (1,)` - checks the accessibility of the `mount` directory. Comes back with `None` which means ok (see `man access`).

Do these commands in the same way:

```
cat oneten
```

```
cat > newfile
hello world
^D          (this is control-D)
```

```
ls
```

(What does the command "`ls -l ../source`" show you? You don't need to answer this in your submission.)

```
rm newfile
```

[8 marks]

Then shut the user space file system down by moving up a directory (out of the user file system directory) and executing the command: `fusermount -u mount`

Check the contents of the `source` and `mount` directories.

Part 2

The `Operations` class in `fuse.py` is the one which does the work we are interested in. This is subclassed in both `passthrough.py` and `memory.py`.

As we have seen `passthrough.py` provides a copy of one directory mounted in a different location and passes all requests back to the original directory. Whereas `memory.py` implements an entirely separate file system in memory. This means when the file system is shut down those files are lost.

Question 3

Answer these questions about `memory.py`.

For the following list of methods in the `Memory` class explain exactly what each method does. Include a statement by statement explanation.

```
__init__, getattr, readdir
open, create, unlink
write, read
```

[8 marks]

Part 3

You now have to create your own user space file system. It works a little bit like a combination of `passthrough.py` and `memory.py`. Call your program `a2fuse2.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.

Run your program with `python a2fuse2.py source mount`

Use the same files in `source` as in Part 1. Make sure that `mount` is initially empty.

The file system works very much like `memory.py` but it starts with some real files from the `source` directory. So the file system has two classes of files which are in the `mount` directory. One consists of real files from the `source` directory and the other of files which only exist in memory.

Any changes which happen to files in the `mount` directory which have been created only in memory (including creating or deleting files, or writing to files) only happen in the `mount` directory. However if the file was initially in the `source` directory then changes get passed back to that directory just as with `passthrough.py`. See the following example.

Start the user space file system in terminal one.

```
python a2fuse2.py source mount
```

Start in the same directory in terminal two.

```
robert@ubuntu:Part2$ cd mount
robert@ubuntu:mount$ ls -l ../source
total 700
-rw-r--r-- 1 robert robert 700001 Aug 29 15:23 hundredthousand
-rw-r--r-- 1 robert robert     31 Sep  3 16:06 oneten
-rw-r--r-- 1 robert robert   3001 Aug 29 15:23 onethousand
-rw-r--r-- 1 robert robert   6001 Aug 29 15:23 twothousand
robert@ubuntu:mount$ ls -l
total 0
-rw-r--r-- 1 robert robert 700001 Aug 29 15:23 hundredthousand
-rw-r--r-- 1 robert robert     31 Sep  3 16:06 oneten
-rw-r--r-- 1 robert robert   3001 Aug 29 15:23 onethousand
-rw-r--r-- 1 robert robert   6001 Aug 29 15:23 twothousand
robert@ubuntu:mount$ cat onethousand twothousand > threethousand
robert@ubuntu:mount$ ls -l ../source
total 700
-rw-r--r-- 1 robert robert 700001 Aug 29 15:23 hundredthousand
-rw-r--r-- 1 robert robert     31 Sep  3 16:06 oneten
-rw-r--r-- 1 robert robert   3001 Aug 29 15:23 onethousand
-rw-r--r-- 1 robert robert   6001 Aug 29 15:23 twothousand
robert@ubuntu:mount$ ls -l
total 0
-rw-r--r-- 1 robert robert 700001 Aug 29 15:23 hundredthousand
-rw-r--r-- 1 robert robert     31 Sep  3 16:06 oneten
-rw-r--r-- 1 robert robert   3001 Aug 29 15:23 onethousand
-rw-rw-r-- 1 robert robert   9002 Sep  4 15:19 threethousand
-rw-r--r-- 1 robert robert   6001 Aug 29 15:23 twothousand
robert@ubuntu:mount$ rm threethousand
robert@ubuntu:mount$ ls -l
total 0
-rw-r--r-- 1 robert robert 700001 Aug 29 15:23 hundredthousand
-rw-r--r-- 1 robert robert     31 Sep  3 16:06 oneten
-rw-r--r-- 1 robert robert   3001 Aug 29 15:23 onethousand
-rw-r--r-- 1 robert robert   6001 Aug 29 15:23 twothousand
robert@ubuntu:mount$ rm oneten
```

```
robert@ubuntu:mount$ ls -l ../source
total 700
-rw-r--r-- 1 robert robert 700001 Aug 29 15:23 hundredthousand
-rw-r--r-- 1 robert robert   3001 Aug 29 15:23 onethousand
-rw-r--r-- 1 robert robert   6001 Aug 29 15:23 twothousand
robert@ubuntu:mount$ ls -l
total 0
-rw-r--r-- 1 robert robert 700001 Aug 29 15:23 hundredthousand
-rw-r--r-- 1 robert robert   3001 Aug 29 15:23 onethousand
-rw-r--r-- 1 robert robert   6001 Aug 29 15:23 twothousand
robert@ubuntu:mount$ cd ..
robert@ubuntu:Part2$ fusermount -u mount
robert@ubuntu:Part2$ ls -l source
total 700
-rw-r--r-- 1 robert robert 700001 Aug 29 15:23 hundredthousand
-rw-r--r-- 1 robert robert   3001 Aug 29 15:23 onethousand
-rw-r--r-- 1 robert robert   6001 Aug 29 15:23 twothousand
robert@ubuntu:Part2$ ls -l mount
total 0
```

For working correctly with `cat`, `ls`, `rm` on a variety of files. Files in the `source` directory can be modified but any new files created only exist in the `mount` directory (in memory).

[10 marks]

Submission

Use the Canvas submission system to submit your assignment. Zip together `A2.txt` and `a2fuse2.py`.

Extra marks

1 mark for including your name and login in both files.

1 mark for any files created by the file system having the correct user and group ids.

[2 marks]

Hints

To help with debugging you can put logging output directly into your code, as long as you model your code on that in `a2fuse1.py`. e.g.

```
logging.debug("whatever you want to print")
```

this will then appear as output in terminal one.

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.

e.g. You do not need to worry about files not existing, or having the wrong privileges. You do not need to worry about symbolic links. You do not need to worry about sparse files. You do not need to consider nested directories.

For those of you who have never programmed in Python, feel free to come for help or ask on Piazza. The language itself is simple, but learning the libraries (or modules as they are called in Python) requires time. Google and StackOverflow are really helpful here and you will eventually become confident with the Python documentation https://docs.python.org/2/.

Look up the Python documentation on:

`getattr` https://docs.python.org/2/library/functions.html?highlight=getattr#getattr (the Python function not the method with the same name in `fuse.py`). In Java terms *attributes* are like methods and instance variables.

the `os` module https://docs.python.org/2/library/os.html (this has lots of the methods the assignment relies on)

N.B. All submitted work must be your work alone. You may discuss assignments with others but by submitting any work you are claiming you did that work without the contributions of others (except for work you clearly identify as being from another source).