THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

SOFTENG 370
Operating Systems

Assignment 2 - User space file system

Worth 10%

due date 9:30pm Friday 4th of October, 2019

Introduction

In lectures the FUSE (file system in user space) library and eduFUSE were introduced and explained. In this assignment you need to write your own user space file system using these tools.

Note the due date change. I will apply my usual penalties for late submissions: 5% off your earned score for every day late up until and including 3 days late (15%).

Setup

Do the assignment either on Ubuntu in the Computer Science labs (unfortunately the image in the SE lab does not have the required packages) or on your own machine (virtual machines work too, but not the Windows subsystem for Linux version 1). The markers will use the CS lab image.

- Install the FUSE development package. (This step should already have been done on the CS lab machines.)

  ```
  sudo apt-get install libfuse-dev
  ```

- Install the maven package. (This step should also already have been done on the lab machines.)

  ```
  sudo apt-get install maven
  ```

- Download the file `MemoryFS.zip` from the `A2` files section of Canvas into a directory and uncompress it.

It is easiest if you use an IDE or editor like Visual Studio Code (VSC). The following instructions are for VSC. You can use other editors but you do need to create a project based on the `pom.xml` file you have unpacked.

- Add the Maven extension to VSC. Maven is a project management tool. You will probably want to install the `Java Extension Pack` and this includes the `Maven for Java` extension.
- Select `File / Open Folder...` in VSC. Navigate to the unzipped folder `MemoryFS`.
- In the `Explorer` panel you should see a section called `Maven Projects`.
- Right click on `MemoryFS` and select `Compile`.

This should compile without problems. You have to be connected to internet for this. You can ignore `WARNING`s.

- Run this by navigating to `src/main/java` in the `Explorer` panel and right clicking on `MemoryFS.java` and selecting `Run`. The program will crash. This is because you need to pass a directory as a command line argument.
- Make a directory called `memMount` somewhere in your home directory. This directory is the mount point for your user space file system.

  ```
  mkdir /home/whatever/memMount
  ```

The `memMount` directory will normally be empty, but when your file system is active it will contain files.

- Pass this directory name as a command line argument to the program. In VSC you open the `launch.json` file in the `.vscode` directory and add the following line inside `configurations`:

```
"args": "-d -f /fullpathnameTo/memMount"
```

Where `/fullpathnameTo/memMount` is the full path to the `memMount` directory. The markers will replace this with a path to a directory they have created.

When you run the program now it should open up the eduFUSE window. At the moment it won't show very much but you can still click on things in the eduFUSE window to see what is there. Examine the "`Charts`", "`Directory`", "`Logger`" and "`Inodes`" panels.

In the "`Charts`" panel you can select the system calls you want to chart, the main ones are already selected.

In the "`Directory`" panel you will eventually see the files created in your file system and be able to click on them in order to see information about them.

In the "`Logger`" panel you can see the information being sent to your file system and back to the operating system. There can be a lot of information here and so you can filter it using the options at the top of the panel. It also helps to "`Clear`" the information when you want to focus on particular events.

The "`Inodes`" panel is useful as you get the assignment going.

Closing the eduFUSE window does not stop your file system from running. You may have to explicitly unmount the file system with:

```
fusermount -u memMount
```

Further Documentation

There are alternative installation instructions for IntelliJ IDEA in the assignment 2 area of Canvas. The name of the file system in this document is `BlockFileSystem`, but you should use `MemoryFS`.

Documentation about eduFUSE including the C and Java user guides can be found at https://github.com/lukethompsxn/edufuse-student . This includes the user guide to the eduFUSE gui https://github.com/lukethompsxn/edufuse-student/tree/master/gui/ .

Other important documentation is in the following `man` pages (this is not a complete listing):

```
man 2 stat
man inode
man utimensat
```

Feel free to ask for help on Piazza for anything you are unsure of.

What you have to do

You need to complete `MemoryFS.java` so that in conjunction with `MemoryINode.java` and `MemoryInodeTable.java` it runs a memory based file system. There are versions of memory based file systems on the web but **we will be checking submissions** against these and these existing versions will not get you any marks.

While the file system is running you can store files in it and access them in all the ways files can be used.

You can use `MemoryINode.java` and `MemoryInodeTable.java` as they are. Read through these files and understand what they are doing. A real inode is more complicated than the `MemoryINode` class but in this case we are really storing the information about the file in a `FileStat` object. The other big simplification is that we can store all of the data for a file as the `content` byte array instance variable.

The `FileStat` objects will need to have values for most of their fields, but you can ignore `st_ino`, `st_dev`, `st_rdev`.

All steps below are worth two marks each.

Step 1

Complete the `readdir` method.

`readdir` gets information about all of the files in the directory.

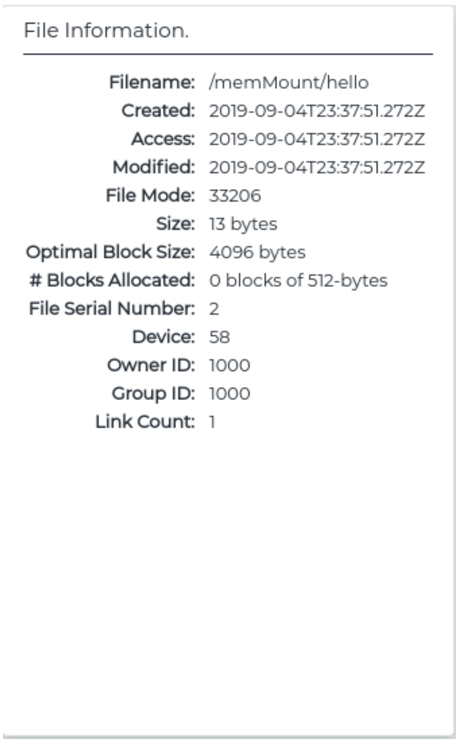
When this is working you should be able to see the `hello` file in the directory, but it won't have all necessary information.

Step 2

Complete the `gettatr` method. In all of the methods you can safely ignore any `FuseFileInfo` parameters.

This method returns all of the file attributes in the `stat` parameter, the first few are done for you. You should also fix up the `init` method so that all useful `FileStat` fields are filled in for the `hello` file.

Check that when you select the `hello` file in the directory panel you see something like this:



File Information.

Filename:	/memMount/hello
Created:	2019-09-04T23:37:51.272Z
Access:	2019-09-04T23:37:51.272Z
Modified:	2019-09-04T23:37:51.272Z
File Mode:	33206
Size:	13 bytes
Optimal Block Size:	4096 bytes
# Blocks Allocated:	0 blocks of 512-bytes
File Serial Number:	2
Device:	58
Owner ID:	1000
Group ID:	1000
Link Count:	1

The file mode will be different and of course the times will be different. The group and owner IDs should be yours. You can use the `unix` instance variable with `getUid` and `getGid` to retrieve these values.

In all that follows you need to make sure that the time fields are correctly updated when files are accessed and modified.

Step 3

Complete the `read` method so that you can read the `hello` file from the command line.

```
cat memMount/hello
```

Step 4

Complete the `write` method so that data can be written to the `hello` file from the command line.

```
echo "more" >> memMount/hello
cat memMount/hello
```

Step 5

Complete the `mknod` method so that files can be added to the directory from the command line.

```
touch memMount/newfile
```

Ensure that new files can be written to and read from.

Step 6

Complete the `unlink` method so that files can be deleted from the directory from the command line.

```
 rm memMount/newfile
```

Step 7

Complete the `link` method so that hard links can be made to files from the command line. Both the source and destination files will be inside your file system. You will probably have to modify the `iNode` classes to do this.

```
ln memMount/hello memMount/hi
```

When you modify the `hello` file you should see the changes in the `hi` file as they are different names for the same file. Make sure that removing files works correctly with hard links.

Step 8

By now you should have a file system which works in one directory. There are two more marks if you can add the functionality to have nested directories inside your memory file system.

The markers will go no deeper than directories inside your mounted directory. All of the preceding operations should work with multiple files, multiple directories and files up to any size (within reason, no files will be more than 100KiBs in size).

Submission, questions and marking guide

Information on how to submit your assignment along with a number of questions to complete and the accompanying marking guide will be on Canvas shortly.

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).