Operating Systems - Final assignment

Supporting Files

1 About FUSE

You will need to install FUSE on your ubuntu **OR** use trottier machine in order to run your tests. Note that FUSE is not available on **linux.cs.mcgill.ca**, but you should be okay on **ubuntu.cs.mcgill.ca** and **mimi.cs.mcgill.ca**. I would recommend to install FUSE directly on your machine. You can do this by running

```
sudo apt-get update
sudo apt-get install fuse
```

2 How to use test files

You will need to open and edit your Makefile. You "uncomment" one of the tests provided.
Say you want to use the first test, this is what your Makefile will look like.

```
CFLAGS = -c -g -Wall -std=gnu99 ‘pkg-config fuse --cflags --libs‘

LDFLAGS = ‘pkg-config fuse --cflags --libs‘

# Uncomment on of the following three lines to compile
SOURCES= disk_emu.c sfs_api.c sfs_test.c sfs_api.h <------THIS CHANGED------
#SOURCES= disk_emu.c sfs_api.c sfs_test2.c sfs_api.h
#SOURCES= disk_emu.c sfs_api.c fuse_wrappers.c sfs_api.h

#if you wish to create your own test - you can do it using this
#SOURCES= disk_emu.c sfs_api.c sfs_mytest.c sfs_api.h

OBJECTS=$(SOURCES:.c=.o)
EXECUTABLE= ID_LASTNAME_FIRSTNAME

all: $(SOURCES) $(HEADERS) $(EXECUTABLE)

$(EXECUTABLE): $(OBJECTS)
   gcc $(OBJECTS) $(LDFLAGS) -o $@

.c.o:
   gcc $(CFLAGS) $< -o $@

clean:
   rm -rf *.o *~ $(EXECUTABLE)
```

Please change the **EXECUTABLE** to reflect on your McGill info. This will be the name of the executable that you will run to test your file.

Now to compile your test file along with changes in your **sfs_api.c** and **sfs_api.h** files, just execute the command

```
make
```

Depending on your linux version, you might have to do this instead

```
make -f Makefile
```

After you have completed the above steps, you can see that there will be an executable file called **ID_LASTNAME_FIRSTNAME** (reflecting on your info). Just run it normally

```
./ID_LASTNAME_FIRSTNAME
```

After you run a test, you can switch to the other test by commenting the previous one and uncommenting the next one. Make sure you run clean first. You clean using

```
make clean
```

3 Running the third test (FUSE)

FUSE will create a mounted disk on your computer. Just like the first two tests, you will need to uncomment FUSE and comment the other two tests.

To see how FUSE works, we have provided you with a working FUSE program (that passes the test). It is working SFS file system. You need to first create a temporary directory (e.g., mytemp). Now run the command

```
./sfs mytemp
```

Run the **ls** command on mytemp and you will see nothing it is an empty directory. That is the file system is empty. Now you copy some files over there or launch an editor like vi or emacs and create some files. You should be able to do most of your normal file writing and reading on the mounted virtual disk you have created. Think of this as the oppositie of assignment 1. In assignment 1 you created a terminal replica, in this assignment, you create a file system replica and use you terminal to run "ls", "cd", etc.. commands.

You will see **sfs_disk.disk** file in the folder where you ran the sfs from. This file is your file system. The data in the files you copied or created are stored over here. To check the contents of the file, load it into an editor that will let you examine binary data (e.g., emacs).
So for your work you will have to run

```
./ID_LASTNAME_FIRSTNAME mytemp
```

To un-mount the file system, find the process that is running the file system and kill it or call the following function on it

```
fusermount -u mytemp
```

Alternatively you can delete your "mytemp" directory and delete the "sfs_disk.disk" file.
Finally, we have provided you with a video showing how to use the provided file system and FUSE test.

4 Seeing your performance in the tests

When you run the tests as mentioned above, you will get an output indicating how your code is doing. It will count how many errors you have. If you still have some buffer overflowing or problems but your number of errors is still 0, you will still get the full mark. The lower the number of errors the better you are doing with the code. Ideally if your program passes test 1 and 2, it shouldn't have problems passing FUSE test. The output of test 1 and 2 will look like the following (if it passes everything).

```
File you requested to open doesn't exist.
Creating a new one..
The inode table is currently full
This file has already been closed..
Cannot find next file..
Test program exiting with 0 errors
```

The last line matters the most, previous lines are just indications of what the test has been doing. As for the FUSE test output - there aren't any. We test your mounted disk by preforming normal terminal writing and reading commands.