Team: On the Deadlock

Team Members:

Mato Ramic	**# 918794492**
Tyler Heslop	**# 920974129**
Christopher Humphrey	**# 922256813**
Jocelyn Guzman	**# 915040482**

Github: Mato-Ramic

File System Report

https://github.com/CSC415-Summer21/csc415-filesystem-Mato-Ramic

PLEASE WATCH THIS YOUTUBE VIDEO

https://www.youtube.com/watch?v=7QcdAxCbxH4

Issues Faced

- Directory files are set to 2 blocks in size.
- The data contained in files read into the filesystem or copied is set to up to 10 blocks.
- Blocks of a file must be sequential.
- Our project works very well, but is very picky about what you type, so ensure that you do not type incorrectly, otherwise you may introduce funky behavior without realizing it at first. **We made a youtube video to demonstrate exactly how to use our filesystem** at https://www.youtube.com/watch?v=7QcdAxCbxH4
- "cp" command does not work very well on directory ("md") files. We occasionally get unexpected behavior. It works fine for other files, such as those which you may copy using cp2fs.
- Sometimes (about 25% of the time) we get unexpected behavior when using . and ./ However, this does not apply to double dots, ".." or "../"
- "ls -all" is not implemented, and thus many of the fields in "fs_stat" such as creation time were ignored. This is based on Chris visiting you during office hours in which Chris was told "not to worry" about "ls -all."

How Program Works

- Upon doing a "make run" our project will initialize the VCB at block 0, and the root directory at block 1, if they do not already exist. The VCB contains the integer index of the root (1) and the integer index of the bitmap. The bitmap will take up the last X blocks of the volume, depending on how many total blocks there are. The bitmap is initialized to 0 (for free), with the exception of the first three indexes being initialized to 1 (because of the VCB and root directory). We used a #define to set directories to be 2 blocks in size, and other files data to be 10 blocks in size.

- Directory files (such as root) are essentially a structure (called "File_Entry_Struct"), which contains some information, such as name, type, data block location (if applicable), etc… along with an Int array. The 0 index of the Int array holds the block location of the instance itself (equivalent to "."), and the 1 index of the array holds the integer block location of wherever its parent is stored (equivalent to ".."). Additional entries may be appended to this array as files are added. In essence, we are building a tree-like structure, with the Int array of integer block locations serving as an array of pseudo pointers.

- As files are created, implement instances of "File_Entry_Struct" in different ways depending on if the file is a directory or a "normal" non-directory file. We write these instances of "File_Entry_Struct" to our volume using LBAwrite.

- Three unique functions of special importance are b_open, b_write, and b_read. b_open is fed a pathname to a file, if the file exists b_open adds relevant information about that file to a b_fcb structure, which is saved in an array. b_open returns the relevant index in the fcbArray[] array, which allows lookup of relevant file information, such as which block to read, or where the index is located in the process of reading the file. b_open MUST use a mutex lock to protect "fd".

- b_read then opens the relevant index of the fcbArray[] array to get information from a b_fcb struct, such as which block to begin reading from. It reads data to a buffer of size 512 using LBAread(...). b_write does something very similar in reverse. After utilization, the file is closed, and the information stored in the fcbArray[] is freed.

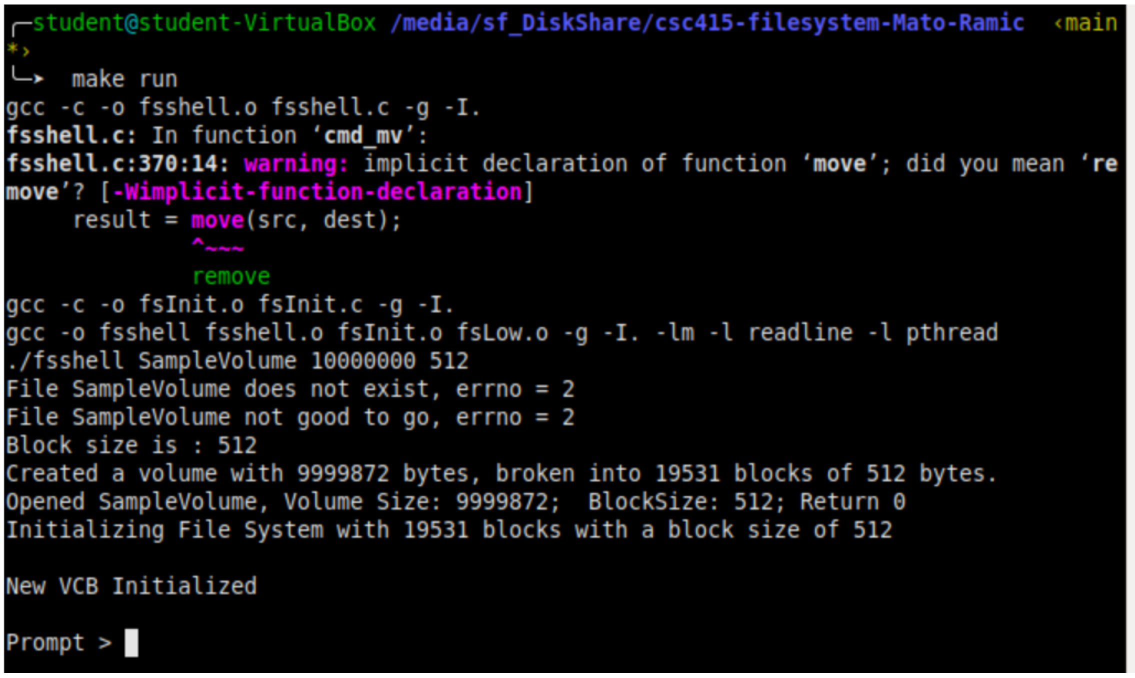
- The "ls" command activates a block of code of our implementation which reads data about the current working directory (or any other directory inputted) into an fdDir structure, which keeps track of the number of entries in the directory, and an index of which entry is being read. Then the fs_readdir(..) methodx is run, which returns (among other things) a fs_diriteminfo structure, which contains the name of said entry. This data is then displayed via the displayFiles(...) method in the shell.

- ○ The “rm” command is highly complex. It uses recursion to find all files in a directory (if you are removing a directory) and all files in each of those files. It then removes pseudo-pointers to said files from their parents File_Entry_Struct entry array. It then sets the relevant index of said files to 0 in the bitmap array.

- ○ The “mv” command merely changes the (“..”) pseudo-pointer from the old parent to a new parent, and removes the pseudo-pointer to said file from their parents’ File_Entry_Struct entry array.
- ○ As an easter egg, the volume named “VolumeWithStarWarsSpeech” contains a speech.

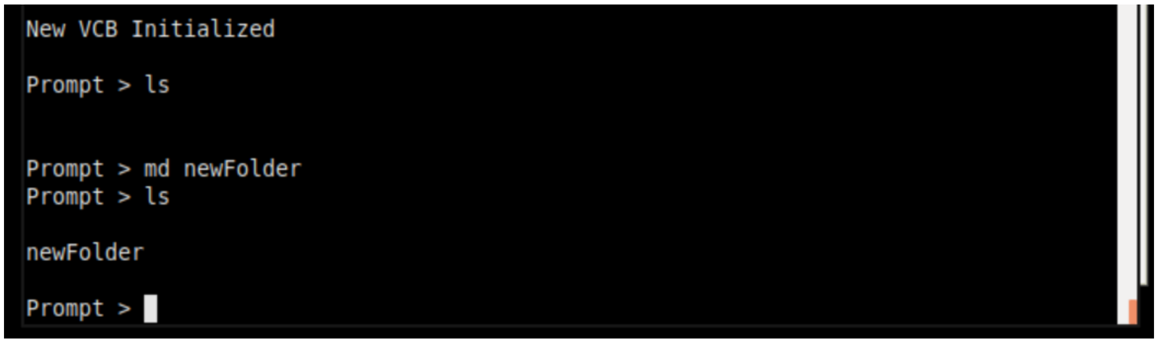
Screenshots of Commands

“make run”



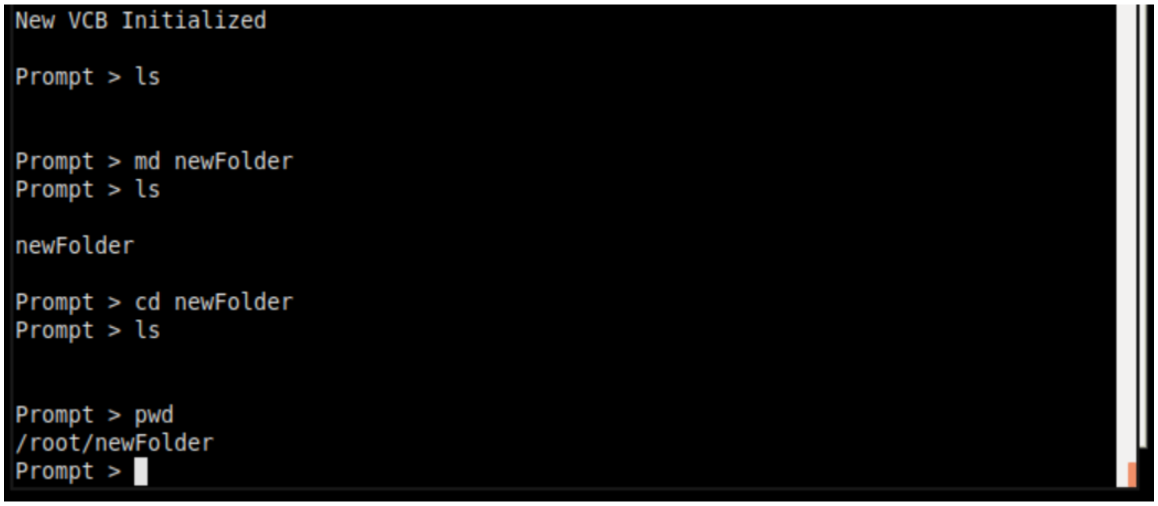
```
┌─student@student-VirtualBox /media/sf_DiskShare/csc415-filesystem-Mato-Ramic  ‹main
*›
└─➤  make run
gcc -c -o fsshell.o fsshell.c -g -I.
fsshell.c: In function ‘cmd_mv’:
fsshell.c:370:14: warning: implicit declaration of function ‘move’; did you mean ‘re
move’? [-Wimplicit-function-declaration]
     result = move(src, dest);
              ^~~~
              remove
gcc -c -o fsInit.o fsInit.c -g -I.
gcc -o fsshell fsshell.o fsInit.o fsLow.o -g -I. -lm -l readline -l pthread
./fsshell SampleVolume 10000000 512
File SampleVolume does not exist, errno = 2
File SampleVolume not good to go, errno = 2
Block size is : 512
Created a volume with 9999872 bytes, broken into 19531 blocks of 512 bytes.
Opened SampleVolume, Volume Size: 9999872;  BlockSize: 512; Return 0
Initializing File System with 19531 blocks with a block size of 512

New VCB Initialized

Prompt > 
```

“ls”



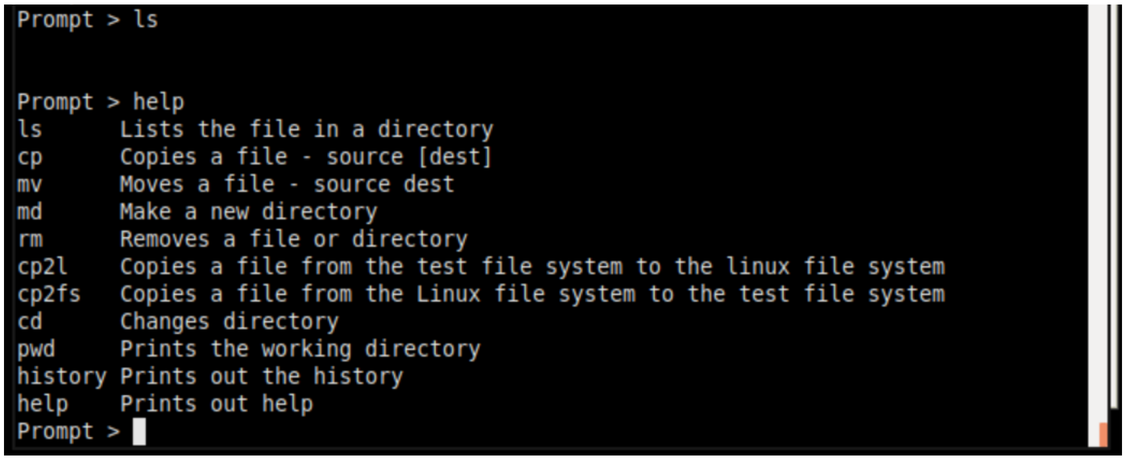
```
New VCB Initialized

Prompt > ls


Prompt > md newFolder
Prompt > ls

newFolder

Prompt >
```

“cd” and “pwd”



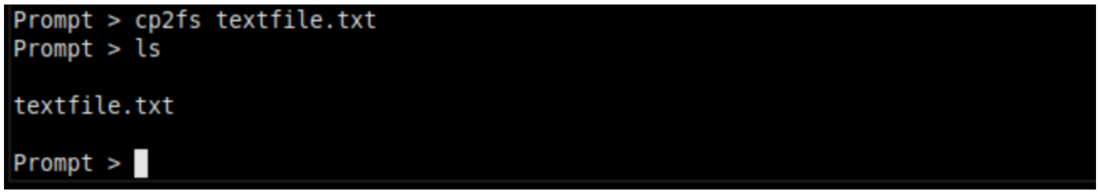
```
New VCB Initialized

Prompt > ls


Prompt > md newFolder
Prompt > ls

newFolder

Prompt > cd newFolder
Prompt > ls


Prompt > pwd
/root/newFolder
Prompt >
```

“Help”



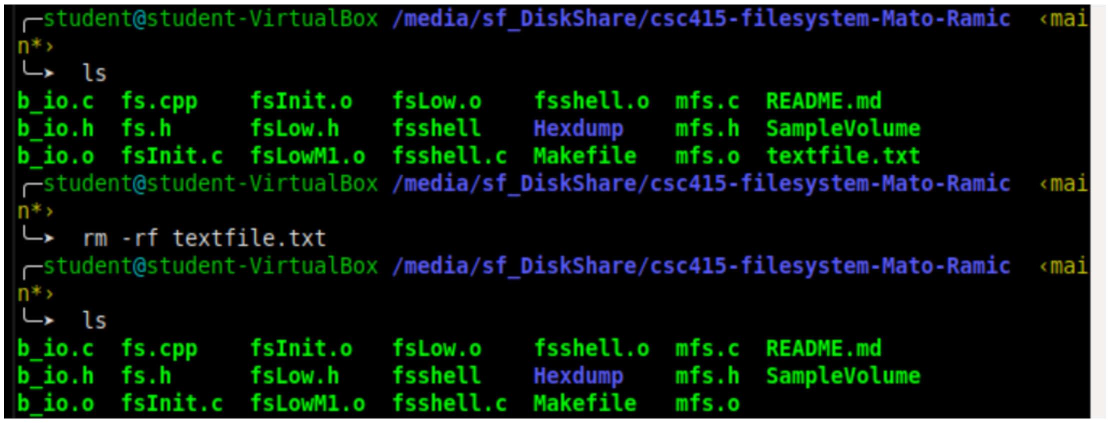
```
Prompt > ls


Prompt > help
ls      Lists the file in a directory
cp      Copies a file - source [dest]
mv      Moves a file - source dest
md      Make a new directory
rm      Removes a file or directory
cp2l    Copies a file from the test file system to the linux file system
cp2fs   Copies a file from the Linux file system to the test file system
cd      Changes directory
pwd     Prints the working directory
history Prints out the history
help    Prints out help
Prompt >
```

“Cp2fs”

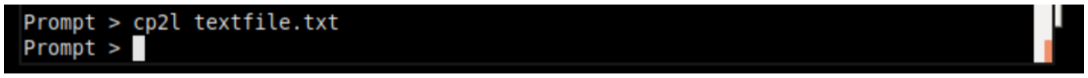


```
Prompt > cp2fs textfile.txt
Prompt > ls

textfile.txt

Prompt >
```

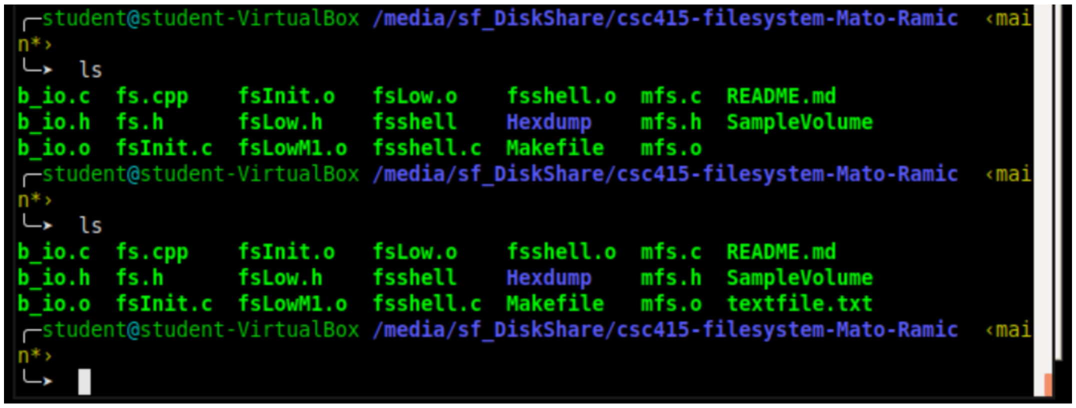
“Cp2l”



```
┌student@student-VirtualBox /media/sf_DiskShare/csc415-filesystem-Mato-Ramic  ‹mai
n*›
└➤ ls
b_io.c  fs.cpp    fsInit.o   fsLow.o    fsshell.o  mfs.c  README.md
b_io.h  fs.h      fsLow.h    fsshell    Hexdump    mfs.h  SampleVolume
b_io.o  fsInit.c  fsLowM1.o  fsshell.c  Makefile   mfs.o  textfile.txt
┌student@student-VirtualBox /media/sf_DiskShare/csc415-filesystem-Mato-Ramic  ‹mai
n*›
└➤ rm -rf textfile.txt
┌student@student-VirtualBox /media/sf_DiskShare/csc415-filesystem-Mato-Ramic  ‹mai
n*›
└➤ ls
b_io.c  fs.cpp    fsInit.o   fsLow.o    fsshell.o  mfs.c  README.md
b_io.h  fs.h      fsLow.h    fsshell    Hexdump    mfs.h  SampleVolume
b_io.o  fsInit.c  fsLowM1.o  fsshell.c  Makefile   mfs.o
```

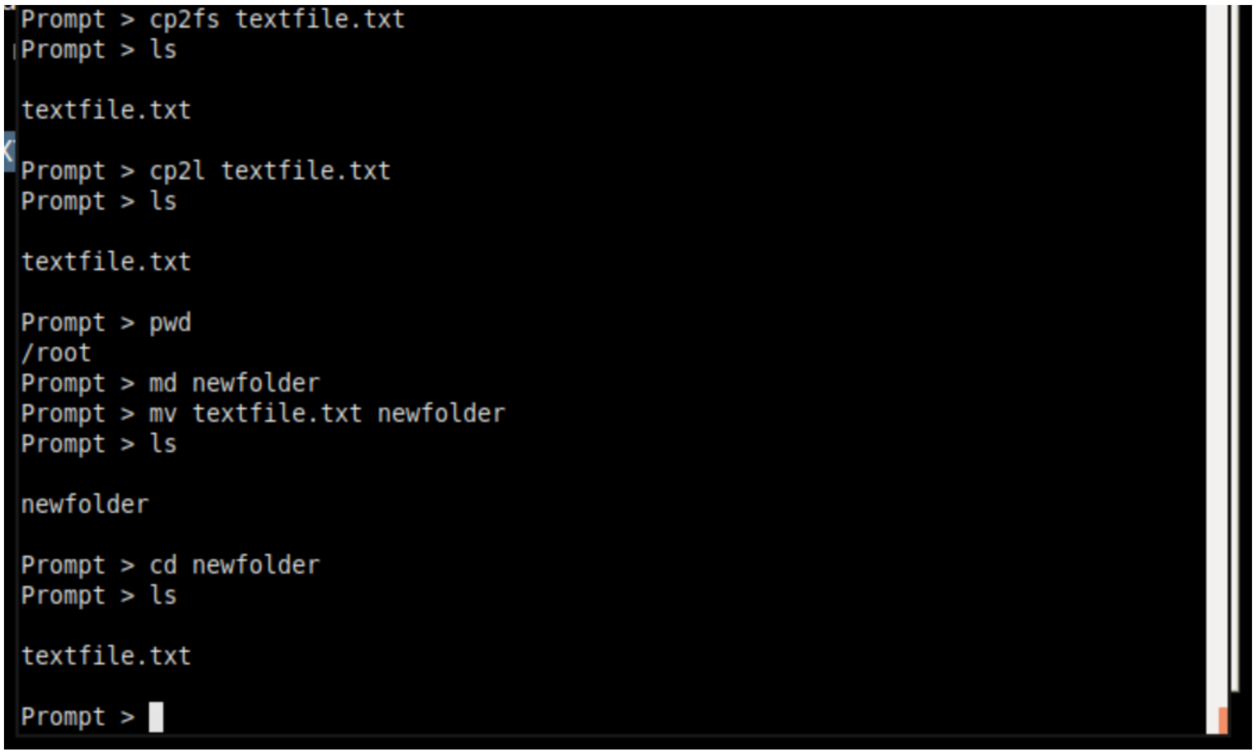


```
Prompt > cp2l textfile.txt
Prompt > █
```



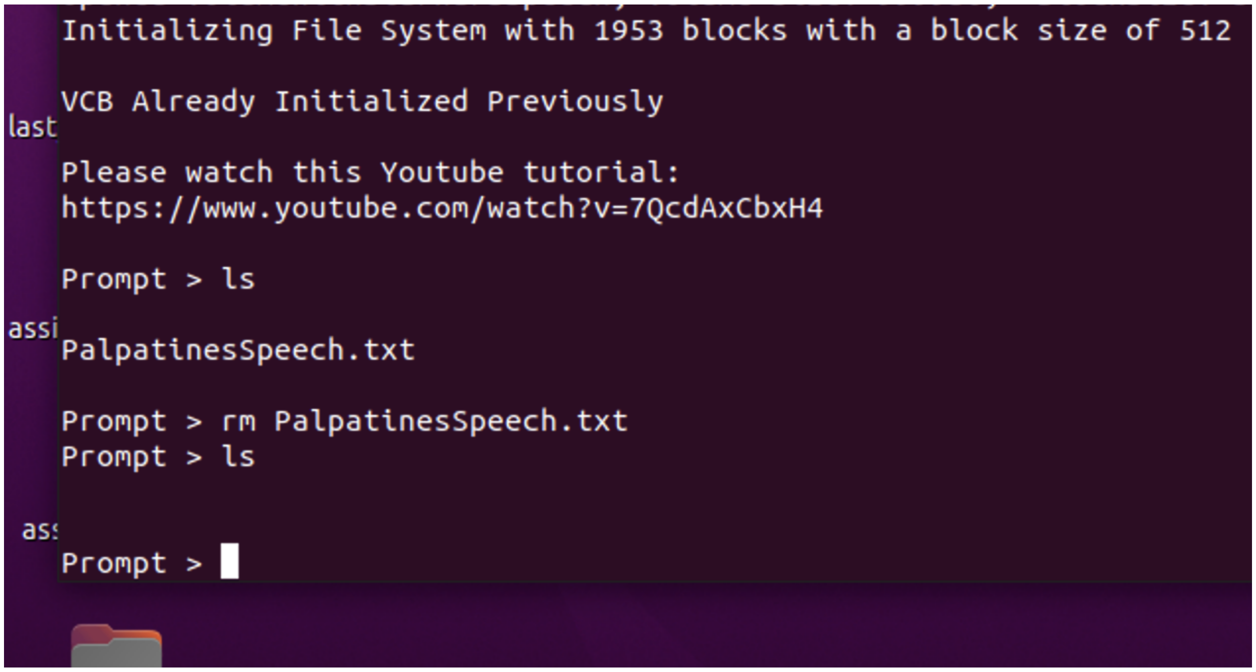
```
┌student@student-VirtualBox /media/sf_DiskShare/csc415-filesystem-Mato-Ramic  ‹mai
n*›
└➤ ls
b_io.c  fs.cpp    fsInit.o   fsLow.o    fsshell.o  mfs.c  README.md
b_io.h  fs.h      fsLow.h    fsshell    Hexdump    mfs.h  SampleVolume
b_io.o  fsInit.c  fsLowM1.o  fsshell.c  Makefile   mfs.o
┌student@student-VirtualBox /media/sf_DiskShare/csc415-filesystem-Mato-Ramic  ‹mai
n*›
└➤ ls
b_io.c  fs.cpp    fsInit.o   fsLow.o    fsshell.o  mfs.c  README.md
b_io.h  fs.h      fsLow.h    fsshell    Hexdump    mfs.h  SampleVolume
b_io.o  fsInit.c  fsLowM1.o  fsshell.c  Makefile   mfs.o  textfile.txt
┌student@student-VirtualBox /media/sf_DiskShare/csc415-filesystem-Mato-Ramic  ‹mai
n*›
└➤ █
```

“mv”



```
Prompt > cp2fs textfile.txt
Prompt > ls

textfile.txt

Prompt > cp2l textfile.txt
Prompt > ls

textfile.txt

Prompt > pwd
/root
Prompt > md newfolder
Prompt > mv textfile.txt newfolder
Prompt > ls

newfolder

Prompt > cd newfolder
Prompt > ls

textfile.txt

Prompt > █
```

“rm”



```
Initializing File System with 1953 blocks with a block size of 512

VCB Already Initialized Previously

Please watch this Youtube tutorial:
https://www.youtube.com/watch?v=7QcdAxCbxH4

Prompt > ls

PalpatinesSpeech.txt

Prompt > rm PalpatinesSpeech.txt
Prompt > ls


Prompt > 
```