Names: Aaron Colmenares, Chandler Cruz, Wesley Xu, Chaoyi Ying

Student ID: 916913613, 917048657, 916260714, 918810235

Class: CSC-415 Fall 2020

GitHub Repository Link: https://github.com/chandlercruz/csc415-CCAW-filesystem

Group File System Project

Description

The file system utilizes an indexed allocation system that provides an advantage compared to linked allocation because all of the pointers are grouped together in an index block. This is so each entry doesn't have a pointer to the first disk block. This allows for files to be accessed directly and randomly, reducing the seek time.

The management system used for keeping track of the free space uses a linked list. This allows for the first block of the list to be cached in the main memory and it will contain the pointer to the next free block in it. It might be difficult to traverse the list but this has no impact on accessing files only when creating and deleting.

The file system is made to be initialized by using the make command and then using make run. Doing these commands in this order will format the volume and also print out all of the necessary information including: volume size, block size, and other important info. Available to the user are the fsshell commands that are used to navigate and populate the system. Creating directories and copying files into folders are possible as demonstrated in the images below.

The functionality of the system is accomplished by utilizing nodes in a linked list that contain the necessary data for using the system. There are a few files that maintain different functionality for different aspects of the system. The fsshell and mfs files are directly connected to user input and the commands that will be processed by the system. On the other hand, files like b_io deal directly with files and the functions to interact with them.

How the Driver Program Works

Before running the program, you do a "make clean" command to clean up the files, and then you do a "make" command to initialize the file system's volume. Then a "make run" will run the file system's shell that allows the user to interact with the file system.

Issues

The main issues that we ran into stemmed from having to work remotely. Due to the current situation, we weren't able to meet in person, and it subsequently became difficult to coordinate debugging, coding, and testing. As a result, we used Discord to communicate with each other. While the progress on the project was slow and stilted, we managed to keep up our communication and were able to divide up work between the four of us. GitHub was also initially tricky to use, however we were able to make use of multiple branches to ensure nobody was stepping over anyone else's work. In addition, we faced many challenges just due to the overall difficulty of the project. However, thanks to communicating with each other and researching the subject at hand, we were able to successfully develop a finished product.

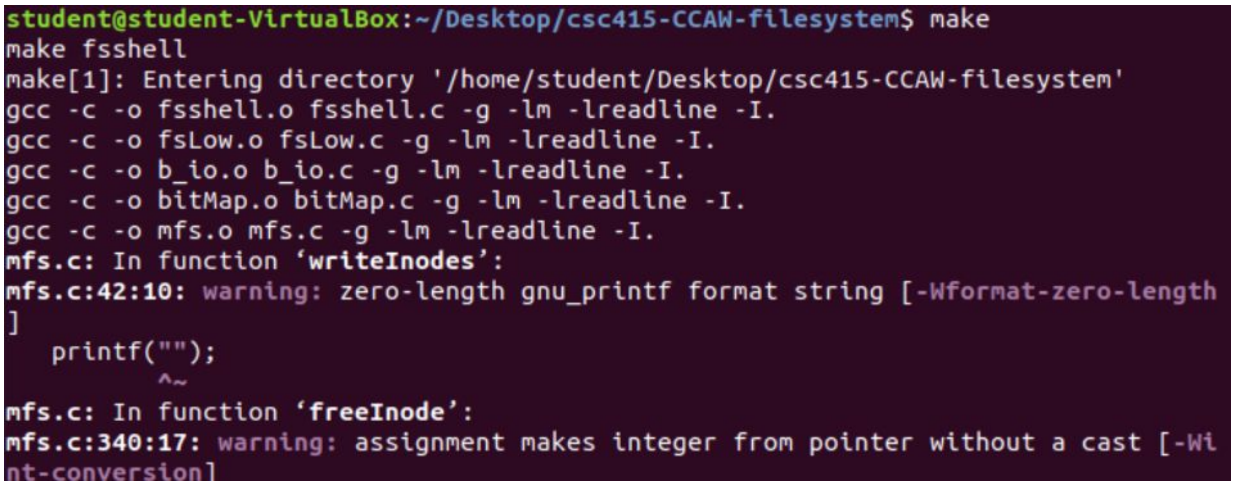
How to Run our File System:

1. Clone the repository onto your device
2. Open a terminal window in the file location of the repository
3. Enter "make" into the terminal
4. Enter "make run" into the terminal to start the shell
5. Use "help" to display the available commands, and use "exit" to exit the file system shell.

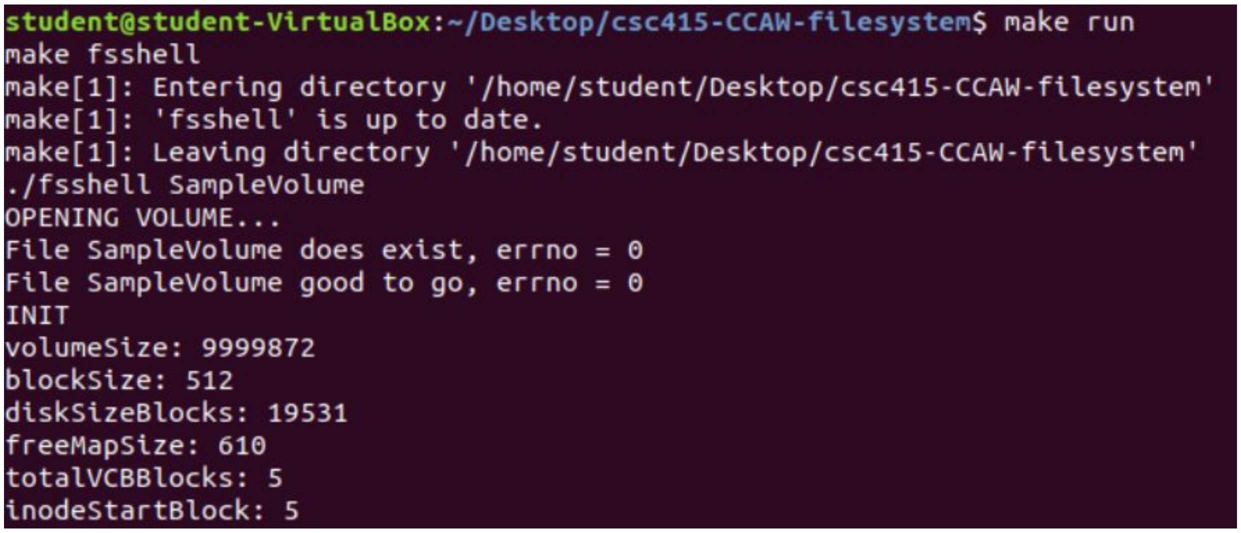
Screenshots detailing the make commands below.

<u>Screenshots of how our file system shell works:</u>

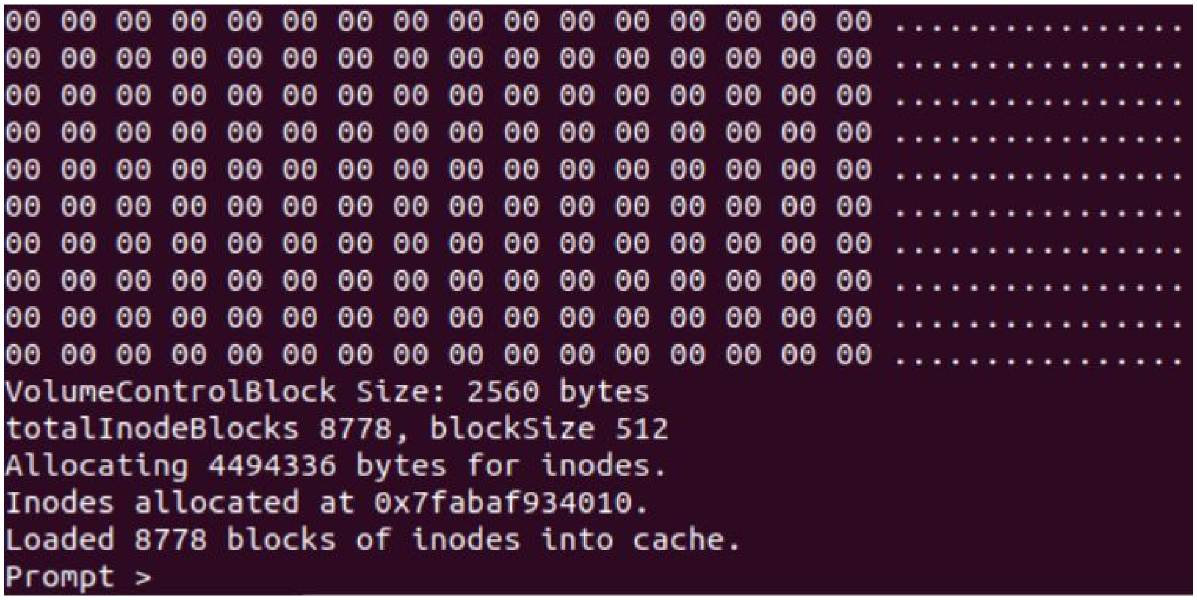
Using "make" and "make run" to create our volume to run the file system



```
student@student-VirtualBox:~/Desktop/csc415-CCAW-filesystem$ make
make fsshell
make[1]: Entering directory '/home/student/Desktop/csc415-CCAW-filesystem'
gcc -c -o fsshell.o fsshell.c -g -lm -lreadline -I.
gcc -c -o fsLow.o fsLow.c -g -lm -lreadline -I.
gcc -c -o b_io.o b_io.c -g -lm -lreadline -I.
gcc -c -o bitMap.o bitMap.c -g -lm -lreadline -I.
gcc -c -o mfs.o mfs.c -g -lm -lreadline -I.
mfs.c: In function ‘writeInodes’:
mfs.c:42:10: warning: zero-length gnu_printf format string [-Wformat-zero-length
]
   printf("");
          ^~
mfs.c: In function ‘freeInode’:
mfs.c:340:17: warning: assignment makes integer from pointer without a cast [-Wi
nt-conversion]
```

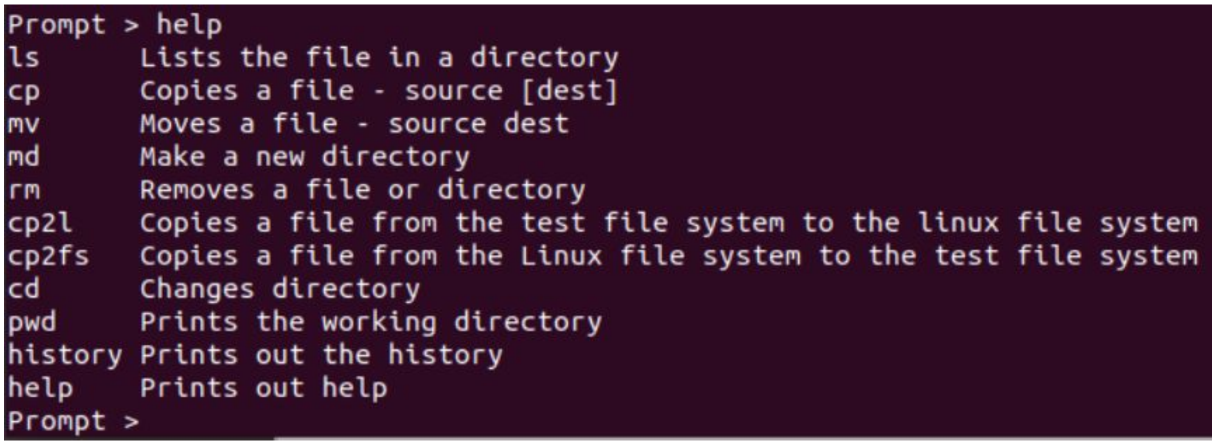


```
student@student-VirtualBox:~/Desktop/csc415-CCAW-filesystem$ make run
make fsshell
make[1]: Entering directory '/home/student/Desktop/csc415-CCAW-filesystem'
make[1]: 'fsshell' is up to date.
make[1]: Leaving directory '/home/student/Desktop/csc415-CCAW-filesystem'
./fsshell SampleVolume
OPENING VOLUME...
File SampleVolume does exist, errno = 0
File SampleVolume good to go, errno = 0
INIT
volumeSize: 9999872
blockSize: 512
diskSizeBlocks: 19531
freeMapSize: 610
totalVCBBlocks: 5
inodeStartBlock: 5
```



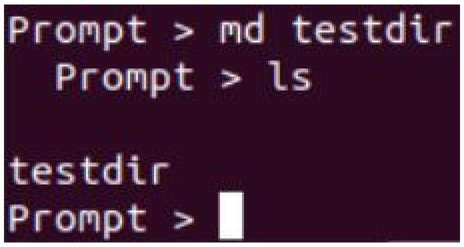
```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
VolumeControlBlock Size: 2560 bytes
totalInodeBlocks 8778, blockSize 512
Allocating 4494336 bytes for inodes.
Inodes allocated at 0x7fabaf934010.
Loaded 8778 blocks of inodes into cache.
Prompt >
```

Help Command:



```
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 >
```

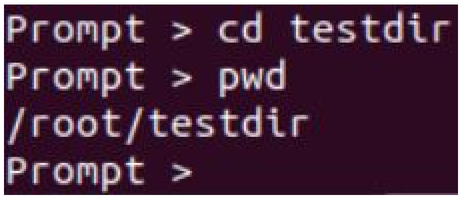
Make Directory (md) Command:



```
Prompt > md testdir
  Prompt > ls

testdir
Prompt > 
```

In here, we created a directory called “testdir” and used ls to view the newly created directory.

Using the Change Directory (cd) to change to the directory we created, and pwd to check the Present Working Directory:

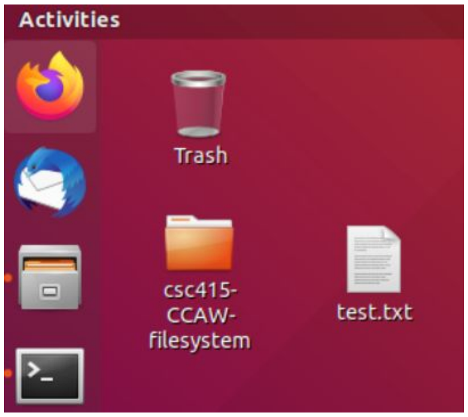


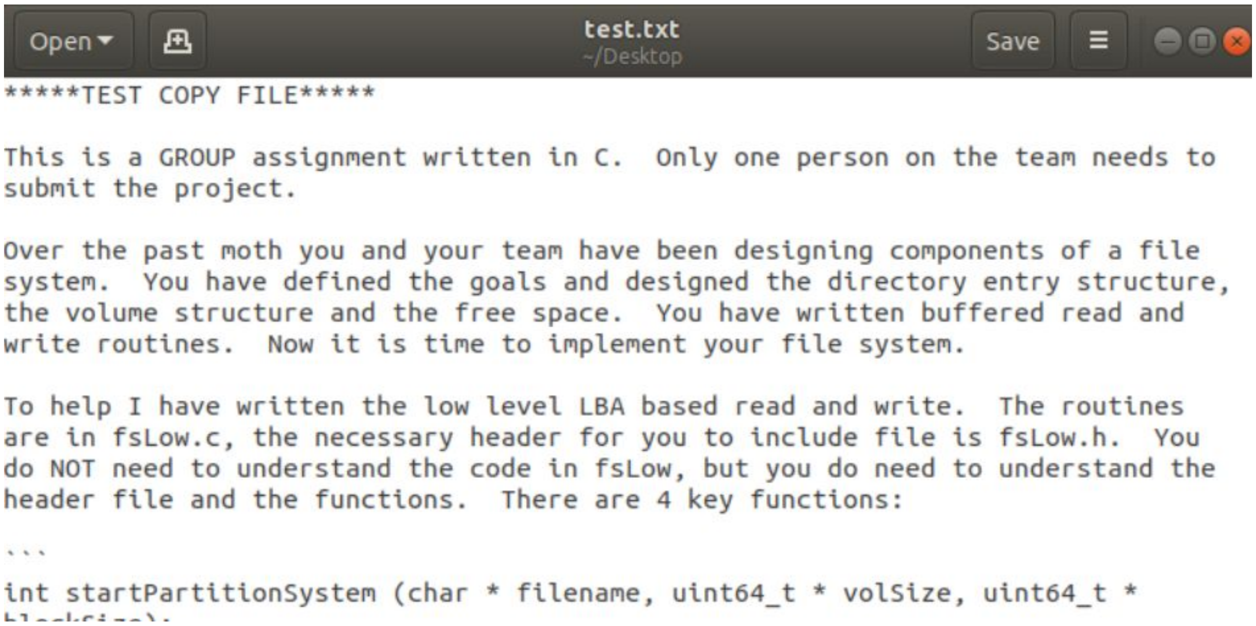
```
Prompt > cd testdir
Prompt > pwd
/root/testdir
Prompt >
```

In here, we used cd to change to our newly created “testdir” directory, and used pwd to check that we were in the correct directory.

Using the cp2fs command

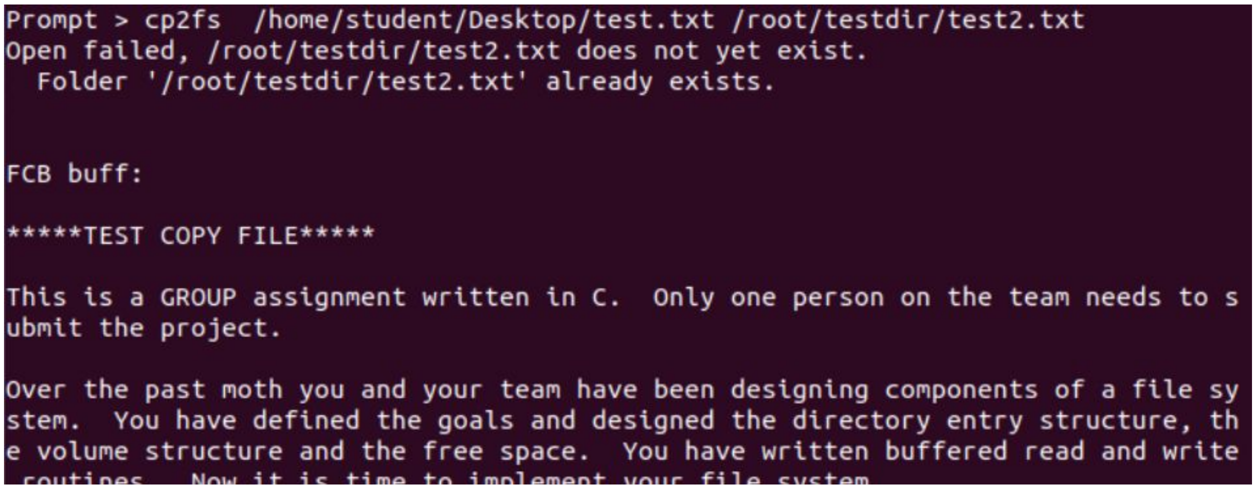
Here’s the test.txt file we’ll be copying



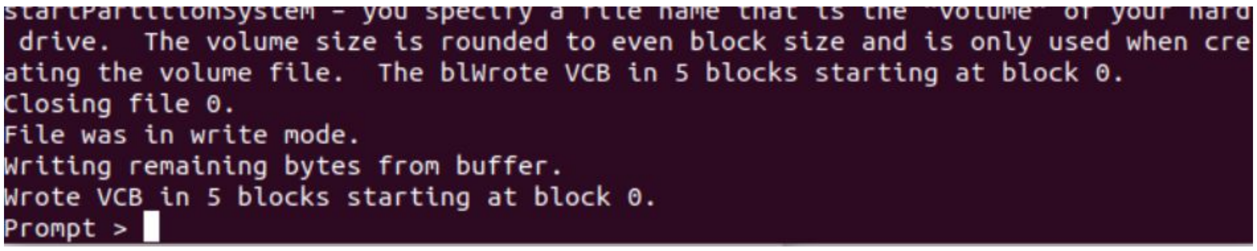


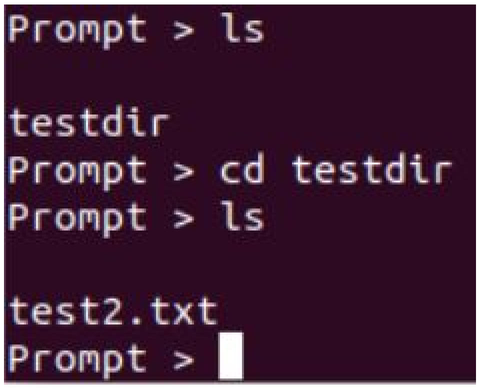
```
*****TEST COPY FILE*****

This is a GROUP assignment written in C.  Only one person on the team needs to
submit the project.

Over the past moth you and your team have been designing components of a file
system.  You have defined the goals and designed the directory entry structure,
the volume structure and the free space.  You have written buffered read and
write routines.  Now it is time to implement your file system.

To help I have written the low level LBA based read and write.  The routines
are in fsLow.c, the necessary header for you to include file is fsLow.h.  You
do NOT need to understand the code in fsLow, but you do need to understand the
header file and the functions.  There are 4 key functions:

```

```
int startPartitionSystem (char * filename, uint64_t * volSize, uint64_t *
```

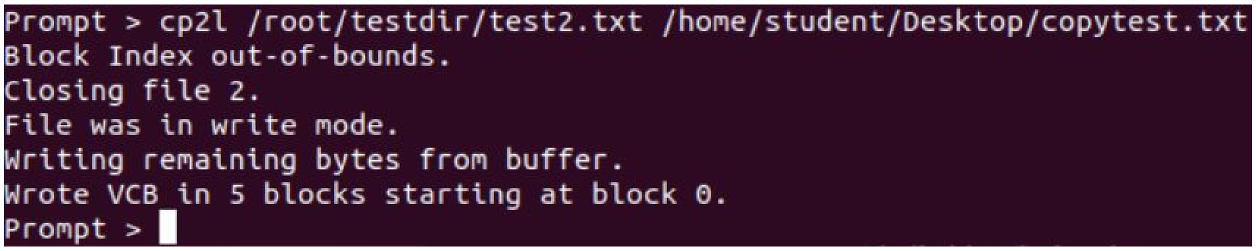


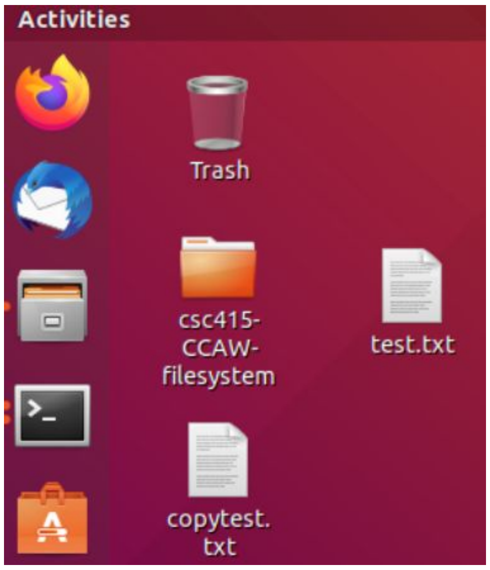
```
Prompt > cp2fs  /home/student/Desktop/test.txt /root/testdir/test2.txt
Open failed, /root/testdir/test2.txt does not yet exist.
  Folder '/root/testdir/test2.txt' already exists.


FCB buff:

*****TEST COPY FILE*****

This is a GROUP assignment written in C.  Only one person on the team needs to s
ubmit the project.

Over the past moth you and your team have been designing components of a file sy
stem.  You have defined the goals and designed the directory entry structure, th
e volume structure and the free space.  You have written buffered read and write
```

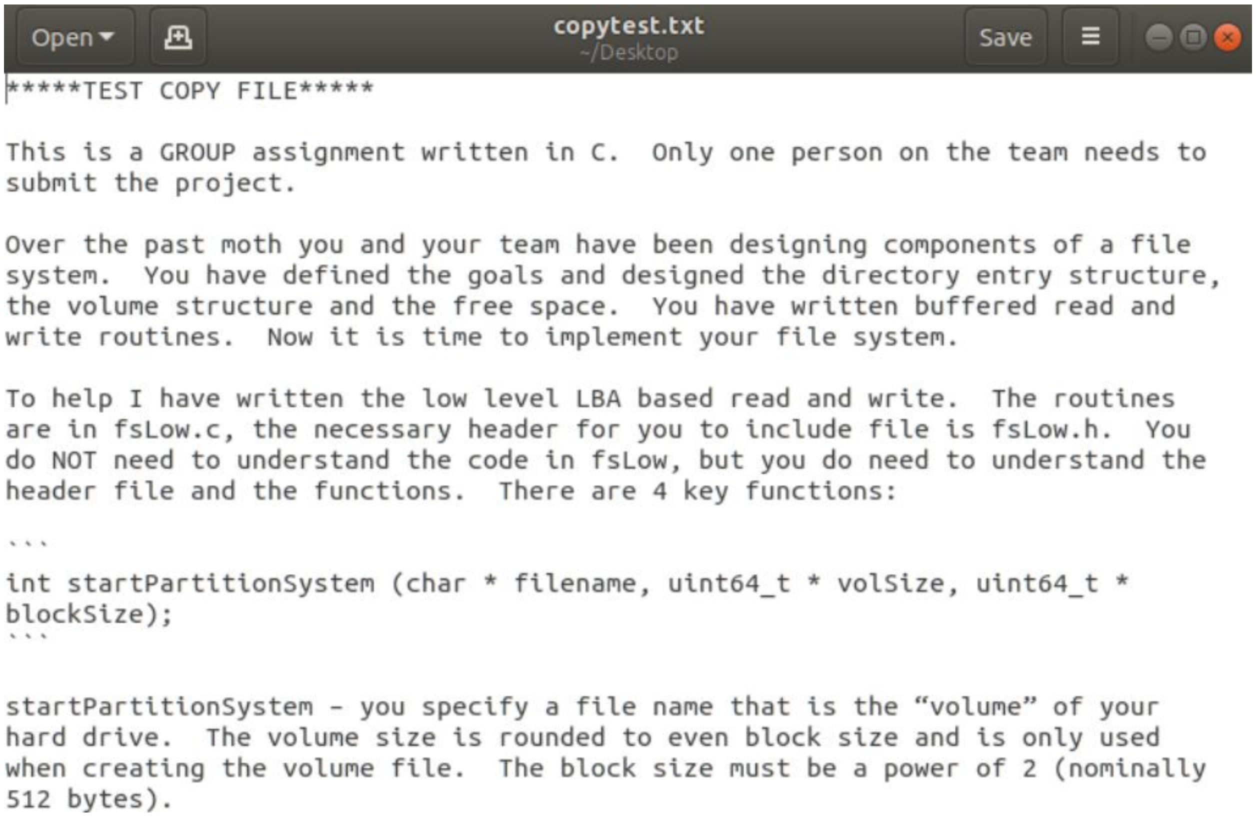

```
startPartitionSystem - you specify a file name that is the "volume" of your hard
 drive.  The volume size is rounded to even block size and is only used when cre
ating the volume file.  The blWrote VCB in 5 blocks starting at block 0.
Closing file 0.
File was in write mode.
Writing remaining bytes from buffer.
Wrote VCB in 5 blocks starting at block 0.
Prompt > 
```


```
Prompt > ls

testdir
Prompt > cd testdir
Prompt > ls

test2.txt
Prompt > 
```

And now the file has been copied into the file system. Now we can copy it back out onto the Linux system…


```
Prompt > cp2l /root/testdir/test2.txt /home/student/Desktop/copytest.txt
Block Index out-of-bounds.
Closing file 2.
File was in write mode.
Writing remaining bytes from buffer.
Wrote VCB in 5 blocks starting at block 0.
Prompt > 
```

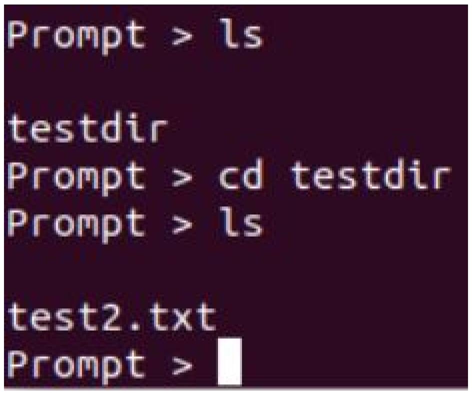




```
*****TEST COPY FILE*****

This is a GROUP assignment written in C.  Only one person on the team needs to
submit the project.

Over the past moth you and your team have been designing components of a file
system.  You have defined the goals and designed the directory entry structure,
the volume structure and the free space.  You have written buffered read and
write routines.  Now it is time to implement your file system.

To help I have written the low level LBA based read and write.  The routines
are in fsLow.c, the necessary header for you to include file is fsLow.h.  You
do NOT need to understand the code in fsLow, but you do need to understand the
header file and the functions.  There are 4 key functions:

```
int startPartitionSystem (char * filename, uint64_t * volSize, uint64_t *
blockSize);
```

startPartitionSystem - you specify a file name that is the “volume” of your
hard drive. The volume size is rounded to even block size and is only used
when creating the volume file. The block size must be a power of 2 (nominally
512 bytes).
```

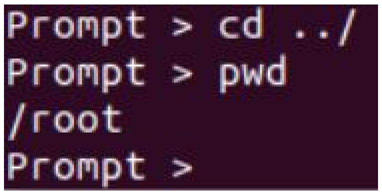
And the file has been copied back successfully.

Demonstrating ls



```
Prompt > ls

testdir
Prompt > cd testdir
Prompt > ls

test2.txt
Prompt > 
```

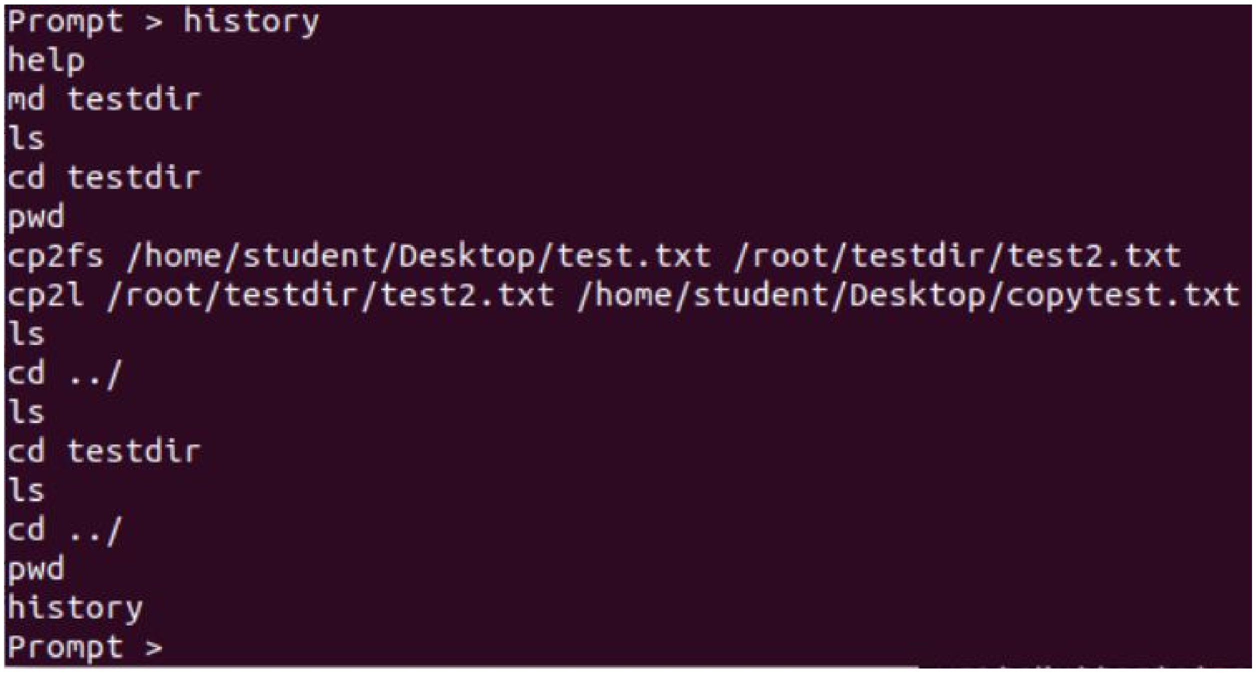
Here, we used ls to view the directories from root, which was just our testdir from earlier. When we cd into testdir, we can see the test2.txt file we copied in earlier.



```
Prompt > cd ../
Prompt > pwd
/root
Prompt >
```

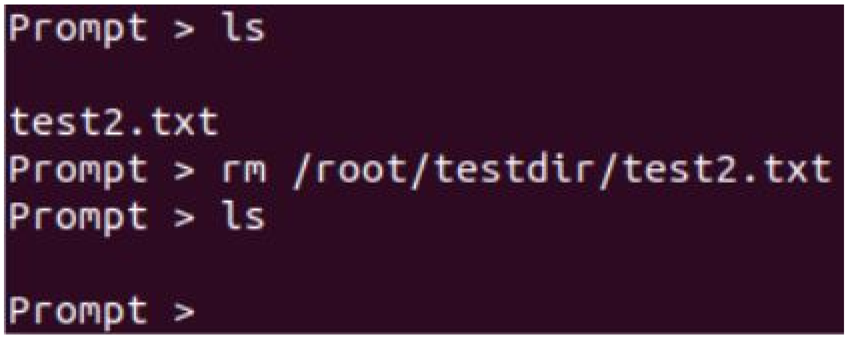
And then we can use cd to return to root.

Demonstrating history:



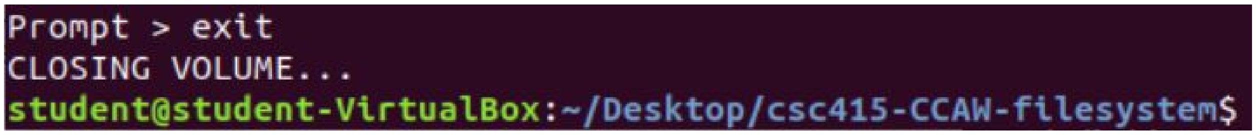
```
Prompt > history
help
md testdir
ls
cd testdir
pwd
cp2fs /home/student/Desktop/test.txt /root/testdir/test2.txt
cp2l /root/testdir/test2.txt /home/student/Desktop/copytest.txt
ls
cd ../
ls
cd testdir
ls
cd ../
pwd
history
Prompt >
```

Demonstrating remove:



```
Prompt > ls

test2.txt
Prompt > rm /root/testdir/test2.txt
Prompt > ls

Prompt >
```

Demonstrating exit:



```
Prompt > exit
CLOSING VOLUME...
student@student-VirtualBox:~/Desktop/csc415-CCAW-filesystem$
```