Class: CSC-415-01 Fall 2020

Team Name: A++

Student Name-ID:
Roberto Herman (918009734),
Cindy Fung Chan (920832364),
Phyoe Thu (918656575),
Aryanna Brown (920188955)

GitHub UserID:
mecosteas,
cny257,
phyoethu100,
Aryannayazmin

Github Link:
https://github.com/CSC415-Fall2020/group-term-assignment-file-system-mecosteas.git

Group Term Assignment - File System

Description:

Our file system uses contiguous memory allocation and a bitmap to manage free space. It can keep track of up to 20 opened files through the implementation of a File Control Table with File Control Blocks. When the program starts, a volume of memory is reserved in the form of a file. This volume is then formatted with a Master Boot Record that lets us know know the Logical Block Array's (LBA's) block size, total number of blocks in our volume, what the LBA's starting index is, where the free space map and root directory are located, and other things such as the name of the volume and the size of our directories. The current working directory is then initialized to start at the root, and the user is now ready to start using our file system.

The main files of our project are mfs, fsshell, dirEntry, bitmap, b_io, and fsFormat, with fsshell as our driver.

The mfs file contains methods that create, remove, edit, open, and close files and directories. It also contains helper methods such as initDir, pathToArray, displayCWD, parsePath, initChildrenNames, and getDirLen. These methods are commonly used throughout our program to help facilitate other functions.

The dirEntry file contains multiple methods such as initDE that initializes and writes a directory entry in the directory given. This file also contains the methods getFreeDirIndex, findDE, getDE, createRootDir, displayDE, and clearDE. createRootDir creates a root directory for our file system and each of these methods play a role in creating and modifying directory entries.

The bitmap file contains methods that allocates, creates, and clears free space, as well as methods that set bits in our bitmap. The methods it contains are searchBit, setBit, clearBit, initFreeSpace, allocateFreeSpace, clearFreeSpace, countFreeSpace, and findFreeSpace. The findFreeSpace method is a helper method that returns the next available free space with the given block counts. initFreeSpace initializes the bitmap array to all 0’s for freespace. A 0 is freespace and 1 is for allocated space.

The b_io file contains methods for how we read and write files to our system from Linux. The methods are initFCB, getFCB, displayFCB, clearFCB, b_open, b_read, b_write, b_close, and b_seek. The FCB methods initialize our file control block with data from the directory entry and returns the index of a free file control block that will serve as the file descriptor. Then there are methods to display and clear what is in the file control block. The b_open method returns a file descriptor of -1 if unable to open the file. b_read copies the requested bytes to the caller’s buffer or directly from the file depending on the size, it then returns the number of bytes requested or remaining. b_write passes the other b_io functions to fsshell through LBAwrite and returns the number of transfer bytes to the buffer.

The fsFormat file contains methods startFileSystem and closeFileSystem which allocates our master boot record, freespace, and root directory for the file system. closeFileSystem frees the buffers in the file system.

Issues we had:

At the beginning, knowing where to start was difficult as we were still trying to wrap our heads around how a file system even works. The idea of memory allocation is very abstract, and discerning the differences between files, directories, the LBA, and directory entries was a great challenge of its own. We would run into problems when we thought we knew something, but then would realize that what we knew was actually incorrect. For example, we thought a DE was supposed to take an entire block of memory in the LBA, but multiple DEs can actually fit inside one block. Before that we thought that directories held the actual files. We didn't know they were supposed to contain only DEs which pointed to a different place where the actual file contents are saved.

We also had issues due to our inexperience with the C programming language. We spent hours debugging code because we didn't know the right way to pass pointers around in functions and how to allocate them to other variables correctly so that they wouldn't point to strange, garbage data. Segmentation faults were very common, and we ended up learning lessons the hard way.

Other types of issues we had were communicating properly. Sometimes code would be pushed to GitHub without letting other teammates know, or the code that was pushed had little documentation to help understand how it worked. During meetings, some teammates would speak very softly and it was hard to hear what they were saying, or sometimes they wouldn't be able to talk because of too much background noise or not having a microphone. As time went on we did get better at communicating though. We began updating each other more often and became more comfortable talking with each other during meetings and through slack.

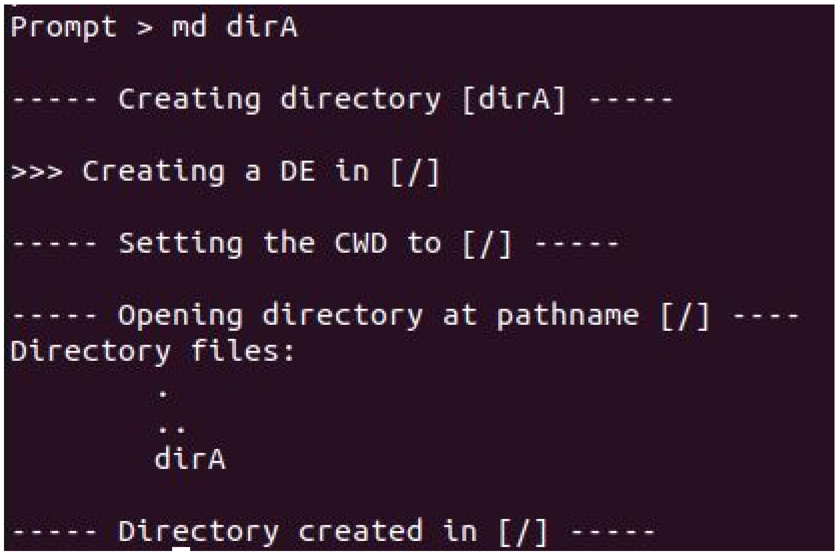
Details of how our driver program works:

Doing a make run will run the driver program for our file system. The driver opens up a shell where the user can enter the following file system commands:

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
exit, exits the shell and the program.

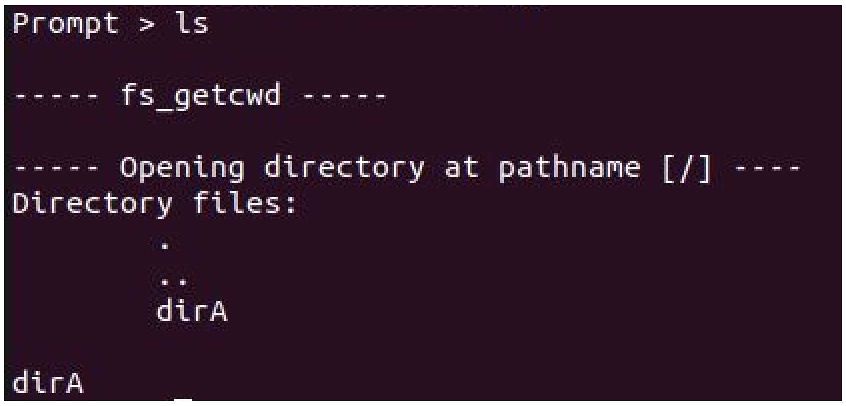
Screenshots of the commands:

md

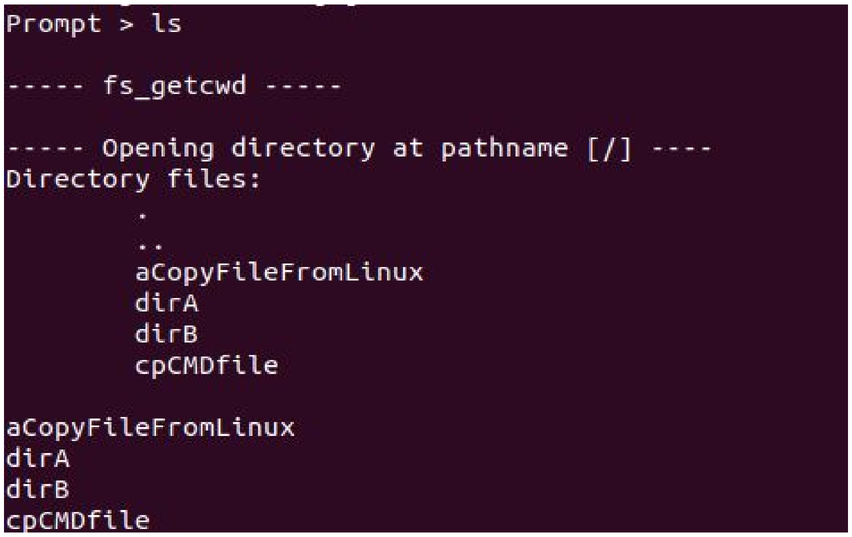


```
Prompt > md dirA

----- Creating directory [dirA] -----

>>> Creating a DE in [/]

----- Setting the CWD to [/] -----

----- Opening directory at pathname [/] ----
Directory files:
        .
        ..
        dirA

----- Directory created in [/] -----
```

ls

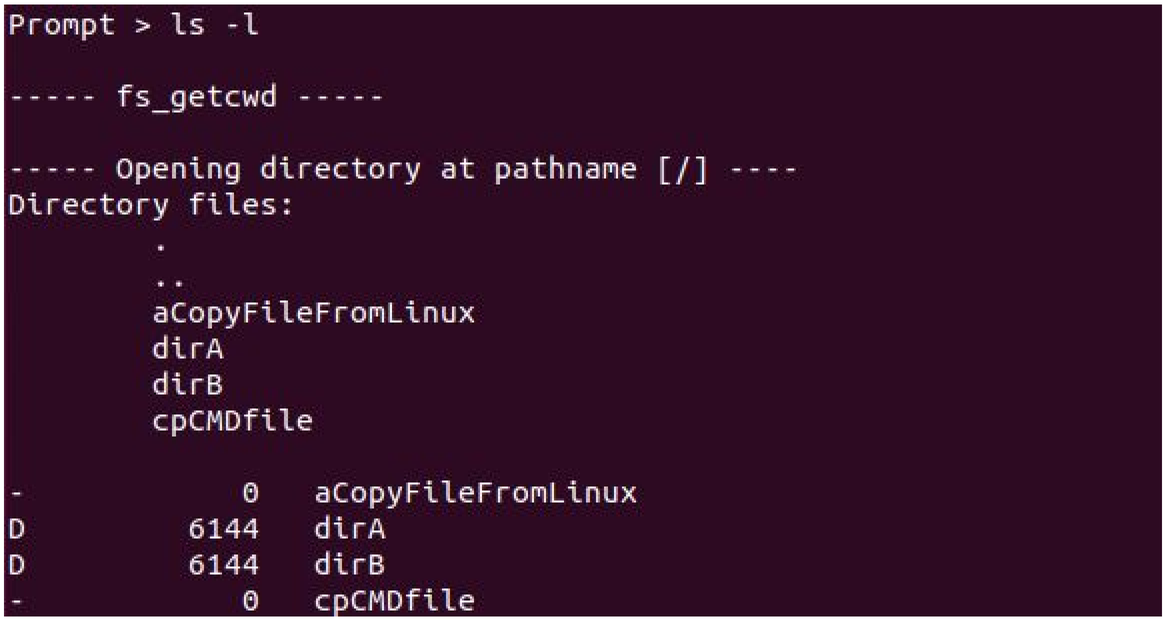


```
Prompt > ls

----- fs_getcwd -----

----- Opening directory at pathname [/] ----
Directory files:
        .
        ..
        dirA

dirA
```



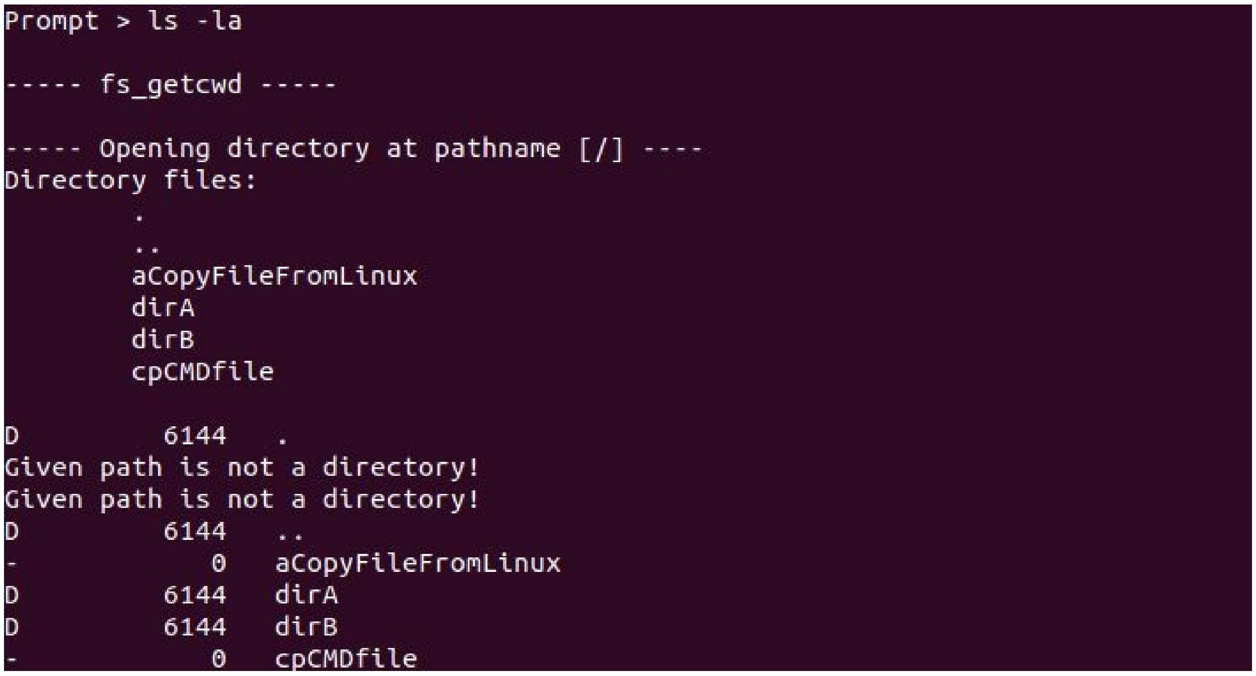
```
Prompt > ls

----- fs_getcwd -----

----- Opening directory at pathname [/] ----
Directory files:
        .
        ..
        aCopyFileFromLinux
        dirA
        dirB
        cpCMDfile

aCopyFileFromLinux
dirA
dirB
cpCMDfile
```

ls -l



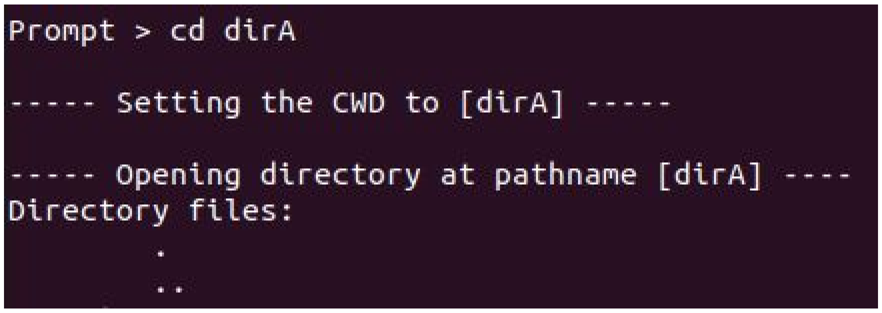
```
Prompt > ls -l

----- fs_getcwd -----

----- Opening directory at pathname [/] ----
Directory files:
        .
        ..
        aCopyFileFromLinux
        dirA
        dirB
        cpCMDfile

-             0   aCopyFileFromLinux
D          6144   dirA
D          6144   dirB
-             0   cpCMDfile
```

ls -la



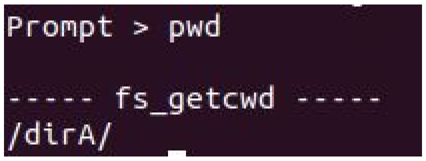
```
Prompt > ls -la

----- fs_getcwd -----

----- Opening directory at pathname [/] ----
Directory files:
        .
        ..
        aCopyFileFromLinux
        dirA
        dirB
        cpCMDfile

D          6144   .
Given path is not a directory!
Given path is not a directory!
D          6144   ..
-             0   aCopyFileFromLinux
D          6144   dirA
D          6144   dirB
-             0   cpCMDfile
```

cd



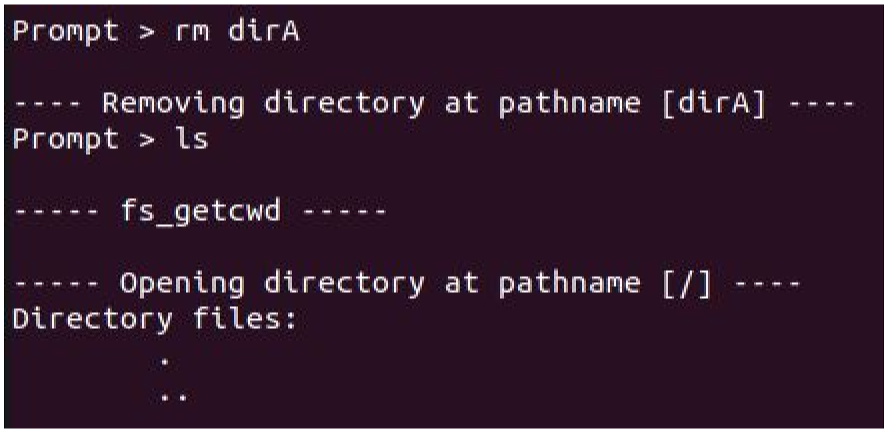
```
Prompt > cd dirA

----- Setting the CWD to [dirA] -----

----- Opening directory at pathname [dirA] ----
Directory files:
        .
        ..
```

pwd



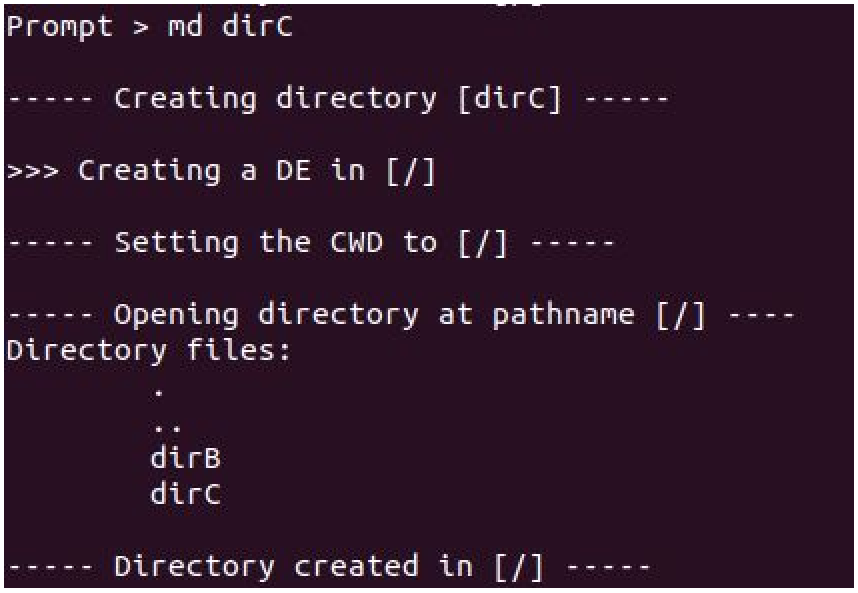
```
Prompt > pwd

----- fs_getcwd -----
/dirA/
```

rm

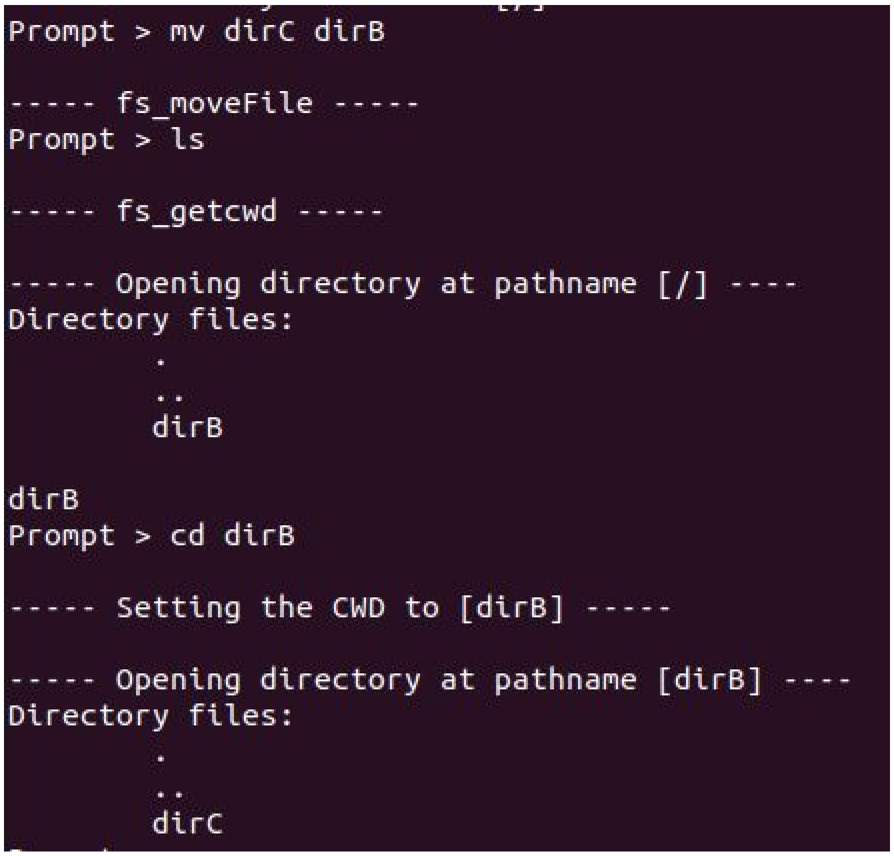


```
Prompt > rm dirA

---- Removing directory at pathname [dirA] ----
Prompt > ls

----- fs_getcwd -----

----- Opening directory at pathname [/] ----
Directory files:
        .
        ..
```

mv

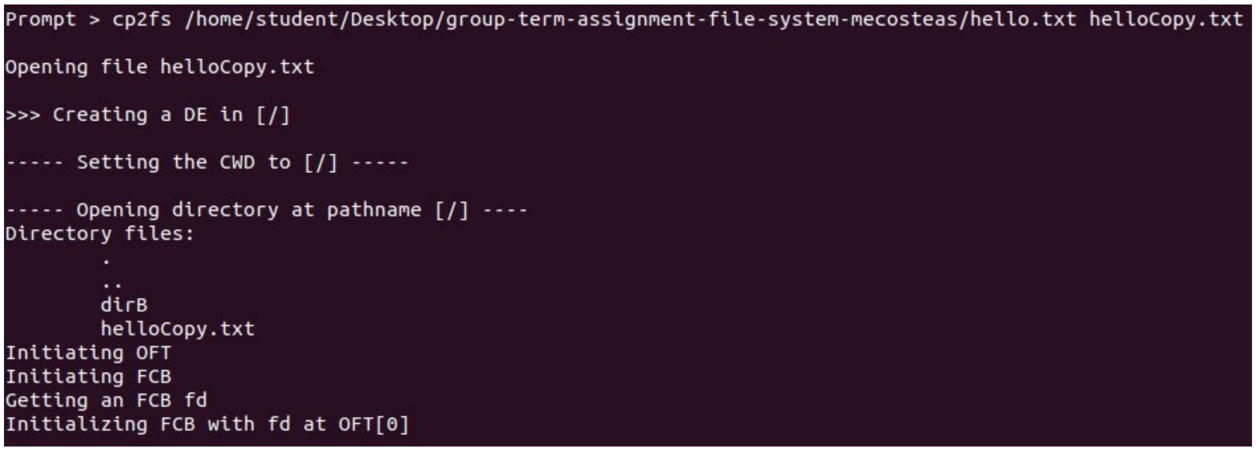


```
Prompt > md dirC

----- Creating directory [dirC] -----

>>> Creating a DE in [/]

----- Setting the CWD to [/] -----

----- Opening directory at pathname [/] ----
Directory files:
        .
        ..
        dirB
        dirC

----- Directory created in [/] -----
```

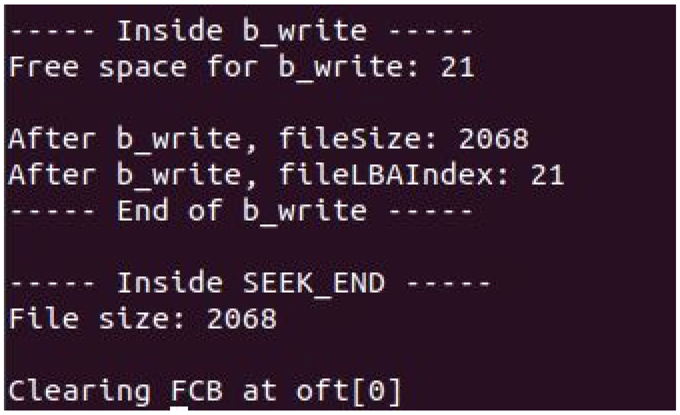


```
Prompt > mv dirC dirB

----- fs_moveFile -----
Prompt > ls

----- fs_getcwd -----

----- Opening directory at pathname [/] ----
Directory files:
        .
        ..
        dirB

dirB
Prompt > cd dirB

----- Setting the CWD to [dirB] -----

----- Opening directory at pathname [dirB] ----
Directory files:
        .
        ..
        dirC
```

cp2fs

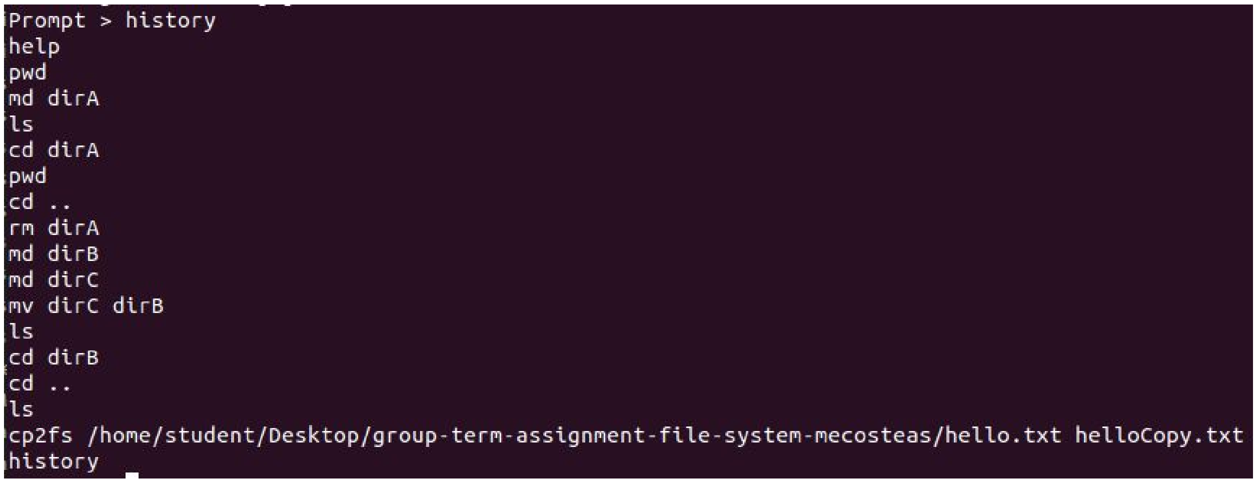


```
Prompt > cp2fs /home/student/Desktop/group-term-assignment-file-system-mecosteas/hello.txt helloCopy.txt

Opening file helloCopy.txt

>>> Creating a DE in [/]

----- Setting the CWD to [/] -----

----- Opening directory at pathname [/] ----
Directory files:
        .
        ..
        dirB
        helloCopy.txt
Initiating OFT
Initiating FCB
Getting an FCB fd
Initializing FCB with fd at OFT[0]
```



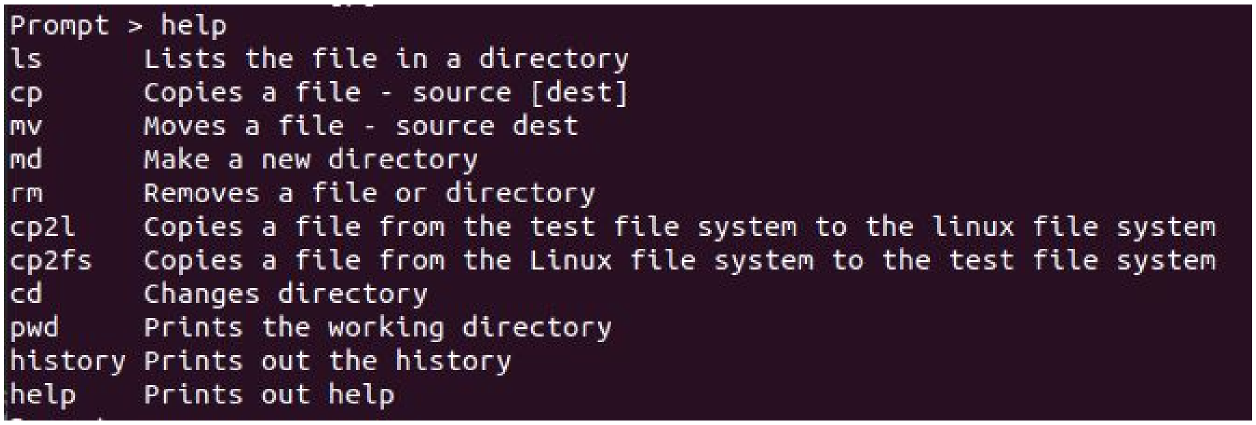
```
----- Inside b_write -----
Free space for b_write: 21

After b_write, fileSize: 2068
After b_write, fileLBAIndex: 21
----- End of b_write -----

----- Inside SEEK_END -----
File size: 2068

Clearing FCB at oft[0]
```

history



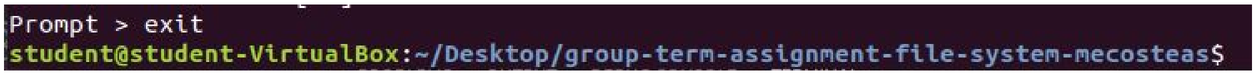
```
Prompt > history
help
pwd
md dirA
ls
cd dirA
pwd
cd ..
rm dirA
md dirB
md dirC
mv dirC dirB
ls
cd dirB
cd ..
ls
cp2fs /home/student/Desktop/group-term-assignment-file-system-mecosteas/hello.txt helloCopy.txt
history
```

help



```
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
```

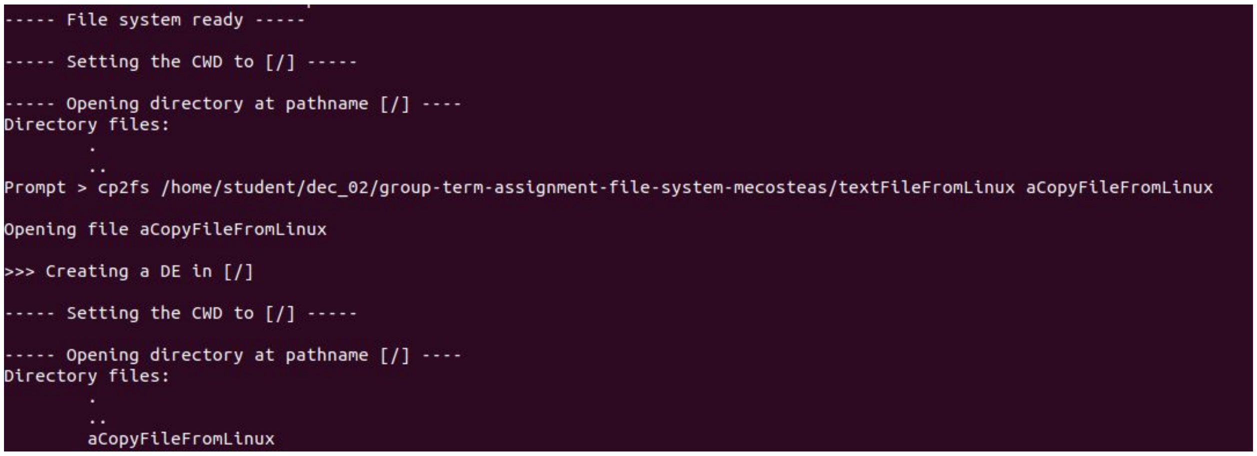
exit



```
Prompt > exit
student@student-VirtualBox:~/Desktop/group-term-assignment-file-system-mecosteas$
```

Commands that have issues:

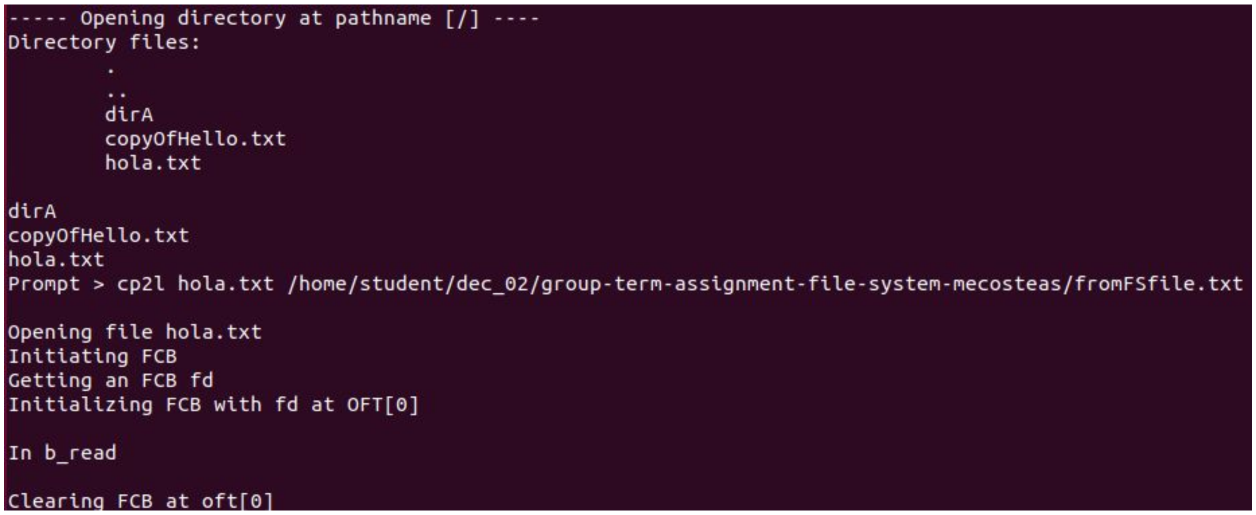
cp2fs



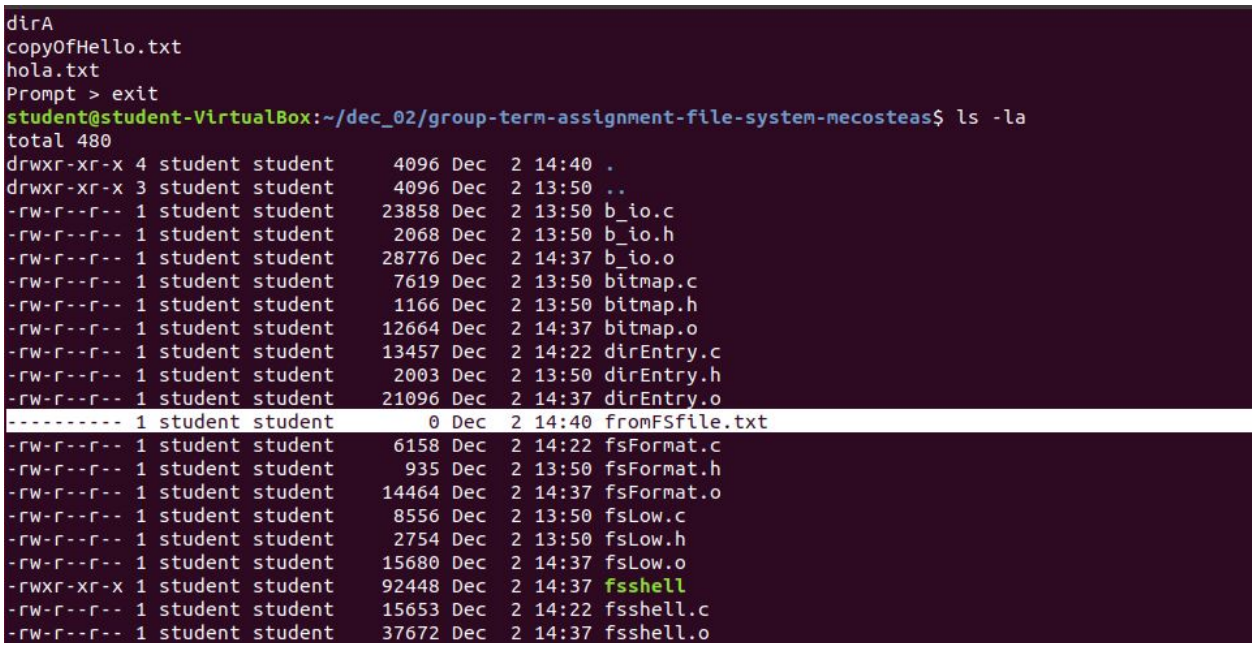
```
----- File system ready -----

----- Setting the CWD to [/] -----

----- Opening directory at pathname [/] ----
Directory files:
        .
        ..
Prompt > cp2fs /home/student/dec_02/group-term-assignment-file-system-mecosteas/textFileFromLinux aCopyFileFromLinux

Opening file aCopyFileFromLinux

>>> Creating a DE in [/]

----- Setting the CWD to [/] -----

----- Opening directory at pathname [/] ----
Directory files:
        .
        ..
        aCopyFileFromLinux
```

This command successfully creates a new DE for the file to be copied, however, it doesn't appear to write anything to it. There are probably some issues with our b_io functions and/or the file control block structure.

cp2l



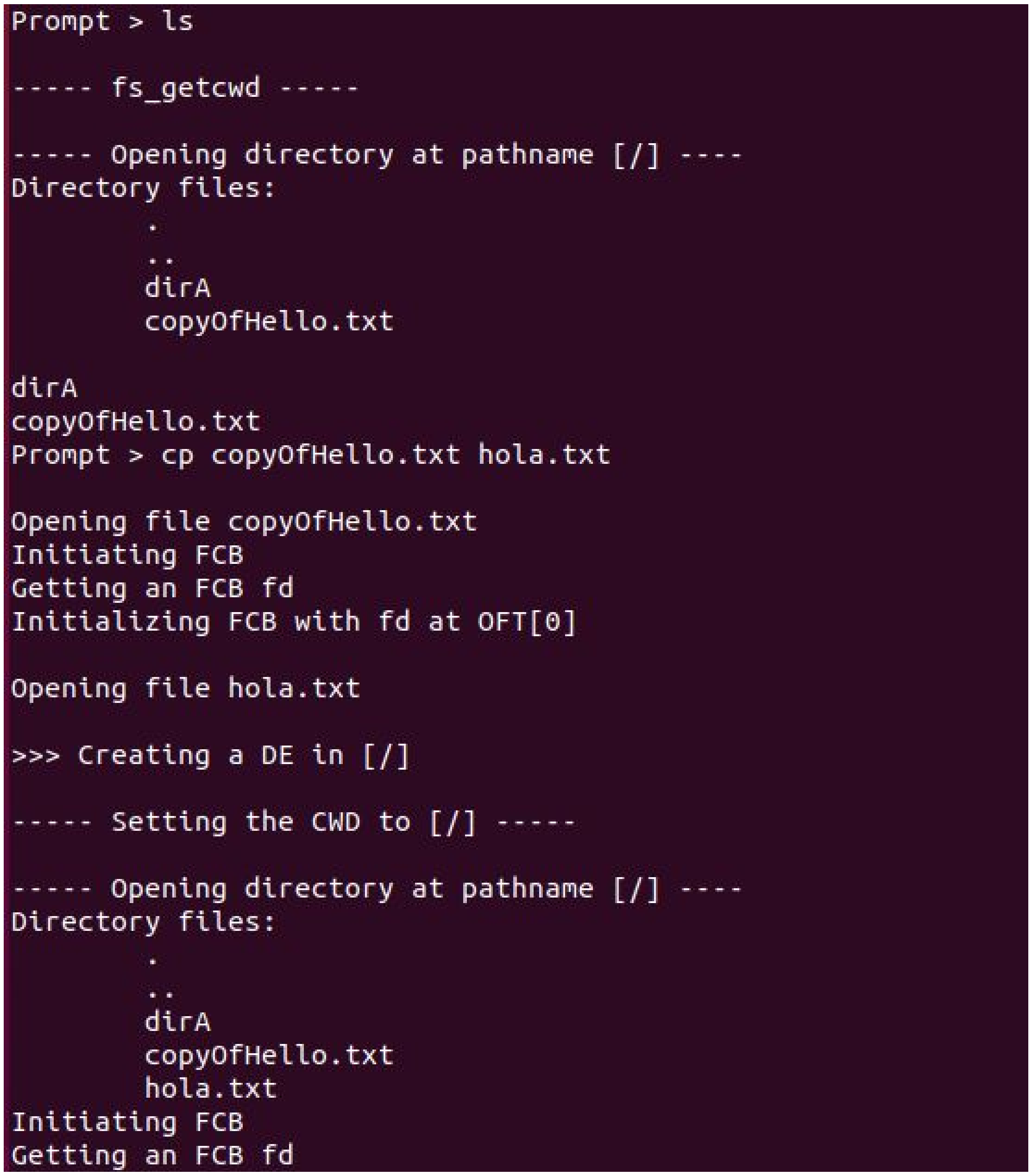
```
----- Opening directory at pathname [/] ----
Directory files:
        .
        ..
        dirA
        copyOfHello.txt
        hola.txt

dirA
copyOfHello.txt
hola.txt
Prompt > cp2l hola.txt /home/student/dec_02/group-term-assignment-file-system-mecosteas/fromFSfile.txt

Opening file hola.txt
Initiating FCB
Getting an FCB fd
Initializing FCB with fd at OFT[0]

In b_read

Clearing FCB at oft[0]
```



```
dirA
copyOfHello.txt
hola.txt
Prompt > exit
student@student-VirtualBox:~/dec_02/group-term-assignment-file-system-mecosteas$ ls -la
total 480
drwxr-xr-x 4 student student     4096 Dec  2 14:40 .
drwxr-xr-x 3 student student     4096 Dec  2 13:50 ..
-rw-r--r-- 1 student student    23858 Dec  2 13:50 b_io.c
-rw-r--r-- 1 student student     2068 Dec  2 13:50 b_io.h
-rw-r--r-- 1 student student    28776 Dec  2 14:37 b_io.o
-rw-r--r-- 1 student student     7619 Dec  2 13:50 bitmap.c
-rw-r--r-- 1 student student     1166 Dec  2 13:50 bitmap.h
-rw-r--r-- 1 student student    12664 Dec  2 14:37 bitmap.o
-rw-r--r-- 1 student student    13457 Dec  2 14:22 dirEntry.c
-rw-r--r-- 1 student student     2003 Dec  2 13:50 dirEntry.h
-rw-r--r-- 1 student student    21096 Dec  2 14:37 dirEntry.o
---------- 1 student student        0 Dec  2 14:40 fromFSfile.txt
-rw-r--r-- 1 student student     6158 Dec  2 14:22 fsFormat.c
-rw-r--r-- 1 student student      935 Dec  2 13:50 fsFormat.h
-rw-r--r-- 1 student student    14464 Dec  2 14:37 fsFormat.o
-rw-r--r-- 1 student student     8556 Dec  2 13:50 fsLow.c
-rw-r--r-- 1 student student     2754 Dec  2 13:50 fsLow.h
-rw-r--r-- 1 student student    15680 Dec  2 14:37 fsLow.o
-rwxr-xr-x 1 student student    92448 Dec  2 14:37 fsshell
-rw-r--r-- 1 student student    15653 Dec  2 14:22 fsshell.c
-rw-r--r-- 1 student student    37672 Dec  2 14:37 fsshell.o
```

This command successfully creates a copy of the file from our file system. However, it is of file size 0 since there are some issues related with the b_io functions.

cp

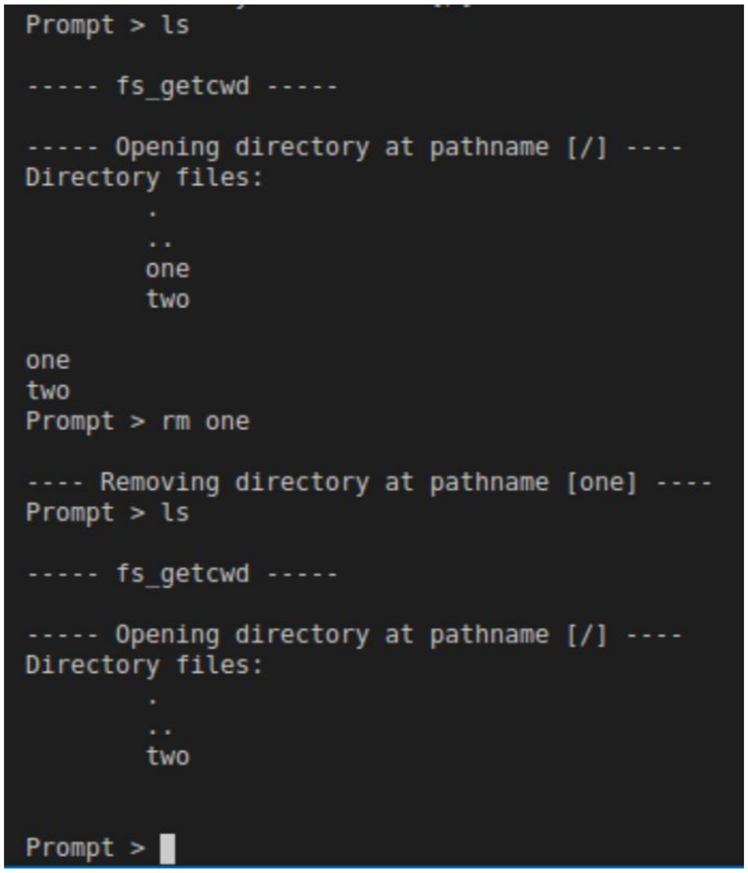


```
Prompt > ls

----- fs_getcwd -----

----- Opening directory at pathname [/] ----
Directory files:
        .
        ..
        dirA
        copyOfHello.txt

dirA
copyOfHello.txt
Prompt > cp copyOfHello.txt hola.txt

Opening file copyOfHello.txt
Initiating FCB
Getting an FCB fd
Initializing FCB with fd at OFT[0]

Opening file hola.txt

>>> Creating a DE in [/]

----- Setting the CWD to [/] -----

----- Opening directory at pathname [/] ----
Directory files:
        .
        ..
        dirA
        copyOfHello.txt
        hola.txt
Initiating FCB
Getting an FCB fd
```

This command creates a new file in our file system. However, we are not sure if it is working properly due to issues with b_io functions.

rm / ls

There’s a bug where if you remove the second directory before the first, the first directory won’t show up on ls either. This is why I left the “Directory files:” display. It gives the true representation of the directory's children since even though the first file isn't being displayed, it can still be accessed.



```
Prompt > ls

----- fs_getcwd -----

----- Opening directory at pathname [/] ----
Directory files:
        .
        ..
        one
        two

one
two
Prompt > rm one

---- Removing directory at pathname [one] ----
Prompt > ls

----- fs_getcwd -----

----- Opening directory at pathname [/] ----
Directory files:
        .
        ..
        two


Prompt > 
```

As you can see, I only removed directory one, but both disappeared. However, two still exists and can still be accessed. It's just not showing up with the ls command anymore for some reason.

Things we'd do differently

I think we should've tried to do the extense type of file system instead of contiguous because having to deal with an unpredictable file size is difficult with contiguous. In extense, we wouldn't be too concerned with that.

I think we should have focused on the directory and its mfs functions before doing the bio functions, because it wasn't until we understood how the directory structure and file system structure worked that we were able to completely understand what needed to happen in the bio functions in regards to file size, file size, file control blocks, LBAindexes, and finding freespace.

We should have found periods of time where we could all code at the same time together. We only did that once or twice, and it’s helpful to have each other immediately available to help other with questions.

What we learned

Operating systems are very intricate pieces of software, and its hardware as well. It takes a lot of planning, thinking, and problem solving. Coding in C is on a different level than Java and other languages where you don’t need to worry about memory allocation. We’ve found a new level of respect for embedded system programmers. Even though this project was long and arduous, we know we’ve barely scratched the surface of what it takes to completely make a robust file system. We learned that through hard work, time, focus, and collaboration, useful programs like a file system can be built for the world to take advantage of and make everyday tasks easier to manage.