Names: Zhuozhuo Liu, Jinjian Tan, Yunhao Wang (Mike on iLearn), Chu Cheng Situ

Student ID: 921410045 (Zhuozhuo), 921383408 (Jinjian), 921458509 (Yunhao), 921409278 (Chu Cheng)

GitHub Names: liuzz10 (Zhuozhuo), KenOasis (Jinjian), mikeyhwang (Yunhao), chuchengsitu (Chu Cheng)

Group Name: return 0

Class: CSC 415 Spring 2021- Section 2

GitHub Repository Link: https://github.com/CSC415-Spring2021/filesystemproject-liuzz10.git

File System Volume Name (in github submission): VolumeZero

Group File System Project

Description:

The file system uses indexed allocation as the allocation method, and bit vector to manage the free space. The file system is divided into several main files to keep the file system working. They are named as b_io, dir, freeSpace, mfs, vcb, fsLow and fsshell. The fsshell is our driver program for the file system.

The b_io file is to initialize the b_open, b_close, b_read, b_write, b_seek functions. For the b_open function we use b_init and b_getFCB as the helper to help b_open to initialize the system and to get the free FCB element. Then we use the getFileLBA, getBlocks, getFileSize from the mfs to get the information for the file. For b_close we first check if there are some remaining to write. If not, we return to value and update all the information to the directory and close the file. For b_read and b_write, it is almost the same as other assignments. Instead of using linux read and write, we modify the code using the LBAread and LBAwrite in the functions. For b_seek, this function only moves the position if needed for the program.

The freeSpace file is our file to handle the space management system. Our free space is contiguous, using a bit vector to track one block per bit. The free space struct consists of information such as bit vector block count, bit vector index count, etc. The free space struct also

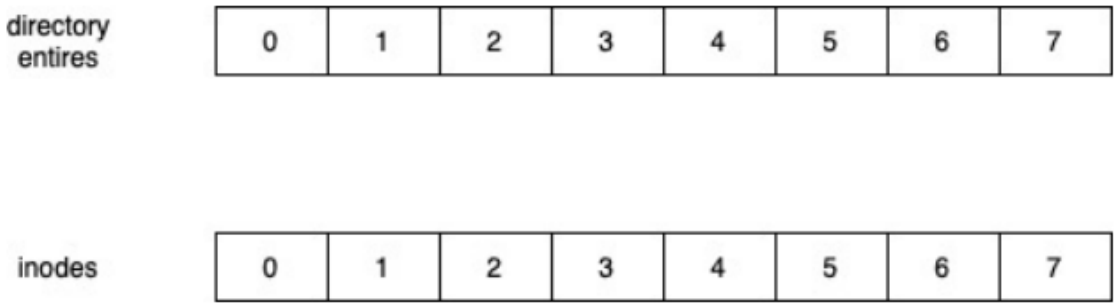
contains a pointer array that is the actual bit vector. The bit vector is an int array, where I use each bit from each int to store the freedom of each block. This is achieved with pointer arithmetic.

The mfs and dir file is our directory that In the directory system:

- directory entry(DE): which holds the logical structure (tree) and the name of the file.
- inode: the inner nodes which hold the metadata of the file. IF the file type is FILE (DT_REG), it still has the address to where to hold the file data.

Both directory entry(DE) and inodes are storing linearly in the disk(LBA) which means their relevant position to the start address (index of the array) is the identifier to locate them.

For example



DE #0 and inode #0 is set for root, the pointer in DE which points to the parent inode maintains the tree structure, when it changes, the logical structure is changed. For example, if you get the directory entries, then you could find their inode and parent inode easily. (You could find parent DE easily since DE # equals inode #)

The vcb files are how the volume control block is managed and contains the data for each block in the volume. It will get information from the free space and directory.

The fsLow file is the file that provides the ability to read and write Logical Blocks and is the main interface for the file system project.

Issues:

1. The main issue is that we ran into stemmed from having to work remotely. Due to the situation, we were not able to meet in person and it became difficult to coordinate debugging, coding, and testing. To solve the issue, we used Discord and Zoom to communicate with each other and keep updating our work. Luckily, we can meet once every two days to keep up the progress, and be able to divide up the work between four of us. In addition, there are some issues integrating our branches together. However, thanks to the communication with each other and searching the error, we were able to finish the project.
2. We had a lot of confusion on how to start the project. The first milestone told us the basis of vcb, free space, and directory, three branching points that we could divide among ourselves. However, the amount of work was daunting, and we were struggling to see how everything combined in the end. This got significantly better with our multiple booked office hours with the professor, and individually asking the professor on Slack about fast questions we had.
3. In terms of b_io, we struggled on the following questions:
    a. In terms of integrating with the free space manager, we were not sure who does the job to move files, like copy the old file and paste it to the new LBA. We asked the professor and decided to let the free space manager take this job so that the caller does not have to worry about it. Just like how realloc works.
    b. In terms of integrating with directory, we were not sure when to update directory about file metadata (e.g., file size, blocks allocated, starting LBA). We firstly implemented that only in b_close. However, considering if there's any crash happen, the directory will not be able to have fresh information about files. Therefore, we improved by updating it as long as there's any change. So we also notice the directory while calling b_write().
    c. The last issue we had is to enable b_write() on an existing file. We realized that cp2fs, cp, and cp2l doesn't test all use cases - they only make us open and trunk a new file of size 0. We want our system more robust - to be able to handle complex cases such as cp a document itself or write something at the end of a file. Therefore, we need to reload the buffer and reset index in fcb. Another example is

b_seek, we also need to reload a buffer from the LBA block that the cursor is currently at.

4. In terms of free space, I struggled on the pointer math that was involved in making a bit vector in C, as I hadn't used pointer arithmetic often at all. I also struggled to understand the functionality of LBAwrite/read and the concept of data persistence, which is not a concept I ever came upon before this project.
5. For the part of the directory system, I have to use a lot of pointer or even pointer to pointer for the dynamic management of the content and structure of the directory system. I have to do a lot of checking on the memory allocated and freed. I met a lot of memory related errors or exceptions at the early stage but I solved it at the end. Another challenge of this project is the relative path issues, I was struggling at the beginning and wrote tons of the codes to handle it but I finally figured it out that the stack could help me to simplify the logic of the code and improve the code structure.

How the Driver Program Works:

1. Clone the repository into Linux: https://github.com/KenOasis/CSC415-GroupProject-FileSystem.git
2. Type "make run" into the command line, now the file system is running.
3. Type commands when you see "prompt >", type exit to "shut down" file system.

ScreenShots of the System Works:

Compilation

```
student@student-VirtualBox:~/Documents/CSC415-GroupProject-FileSystem$ make run
gcc -c -o fsshell.o fsshell.c -g -I.
gcc -c -o fsLow.o fsLow.c -g -I.
gcc -c -o b_io.o b_io.c -g -I.
gcc -c -o freeSpace.o freeSpace.c -g -I.
gcc -c -o vcb.o vcb.c -g -I.
gcc -c -o dir.o dir.c -g -I.
gcc -c -o mfs.o mfs.c -g -I.
gcc -o fsshell fsshell.o fsLow.o b_io.o freeSpace.o vcb.o dir.o mfs.o -g -I. -lm -l readline -l pthread
./fsshell
File VolumeZero does not exist, errno = 2
File VolumeZero not good to go, errno = 2
Block size is : 512
Created a volume with 9999872 bytes, broken into 19531 blocks of 512 bytes.
Opened VolumeZero, Volume Size: 9999872;  BlockSize: 512; Return 0
Initializing VCB......
Prompt >
```

ls - Lists the file in a directory

- ls: list out all files in the current directory
- ls -l: list out the size of each file

```
Prompt > ls

new_folder
README.md
Prompt > ls -l

D          264   May 07 19:17    new_folder
-         7020   May 07 20:35    README.md
Prompt >
```

cp - Copies a file - source dest

- Example: copy and paste README.md in the same directory as README_copy.md

```
Prompt > ls

README.md
Prompt > cp README.md README_copy.md
Prompt > ls

README.md
README_copy.md
Prompt >
```

mv - Moves a file - source dest

- Example: move file README.md to new_folder

```
Prompt > ls

README.md
new_folder
Prompt > mv README.md new_folder
Prompt > ls

new_folder
Prompt > cd new_folder
Prompt > ls

README.md
Prompt >
```

md - Make a new directory

- Example: make a new directory “Documents”

```
Prompt > ls

Admin
Guest
Jimmy
Prompt > md Documents
Prompt > ls

Admin
Guest
Jimmy
Documents
Prompt >
```

rm - Removes a file or directory

- Example: remove README.md from the current directory

```
Prompt > ls

new_folder
README.md
Prompt > rm README.md
Prompt > ls

new_folder
Prompt >
```

cp2l - Copies a file from the test file system to the linux file system

- Example: copy README_copy.md from our file system to linux file system

```
Prompt > ls

README.md
README_copy.md
Prompt > cp2l README_copy.md
Prompt >
```

- The length of README_copy.md is the same with the length of the source file README.md

```
-rw-r--r-- 1 student student    25650 May  7 18:55  dir.c
-rw-r--r-- 1 student student     6868 May  7 18:55  dir.h
-rw-rw-r-- 1 student student    38400 May  7 18:57  dir.o
-rw-r--r-- 1 student student     4138 May  7 18:55  freeSpace.c
-rw-r--r-- 1 student student      749 May  7 18:55  freeSpace.h
-rw-rw-r-- 1 student student    11368 May  7 18:57  freeSpace.o
-rw-r--r-- 1 student student     8426 May  7 18:54  fsLow.c
-rw-r--r-- 1 student student     1690 May  7 18:54  fsLowDemo.c
-rw-r--r-- 1 student student     2534 May  7 18:54  fsLow.h
-rw-rw-r-- 1 student student    15624 May  7 18:57  fsLow.o
-rwxrwxr-x 1 student student    90496 May  7 18:57  fsshell
-rw-r--r-- 1 student student    15916 May  7 18:55  fsshell.c
-rw-rw-r-- 1 student student    36672 May  7 18:57  fsshell.o
drwxr-xr-x 2 student student     4096 May  7 18:54  Hexdump
-rw-r--r-- 1 student student     1802 May  7 18:55  Makefile
-rw-r--r-- 1 student student    20077 May  7 18:55  mfs.c
-rw-r--r-- 1 student student     7719 May  7 18:55  mfs.h
-rw-rw-r-- 1 student student    28960 May  7 18:57  mfs.o
---------- 1 student student     7020 May  7 19:05  README_copy.md
-rw-r--r-- 1 student student     7020 May  7 18:54  README.md
-rw-r--r-- 1 student student      586 May  7 18:55  spaceManagement.md
-rw-r--r-- 1 student student    27308 May  7 18:55 'struct of directory
system.pdf'
-rw-r--r-- 1 student student     1734 May  7 18:55  vcb.c
-rw-r--r-- 1 student student      890 May  7 18:55  vcb.h
-rw-rw-r-- 1 student student     9808 May  7 18:57  vcb.o
student@student-VirtualBox:~/Documents/CSC415-GroupProject-FileSystem$
```

- There’s no difference between 2 files

```
student@student-VirtualBox:~/Documents/CSC415-GroupProject-FileSy
stem$ chmod 0777 README_copy.md
student@student-VirtualBox:~/Documents/CSC415-GroupProject-FileSy
stem$ diff README.md README_copy.md
student@student-VirtualBox:~/Documents/CSC415-GroupProject-FileSy
stem$
```

cp2fs - Copies a file from the Linux file system to the test file system

- Example: copy README.md from the Linux file system to our file system - the length is the same with the original file.

```
Prompt > ls

Prompt > cp2fs README.md
Prompt > ls

README.md
Prompt > ls -l

-          7020    May 07 19:02     README.md
```

cd - Changes directory

- cd ..: Go back to last level

```
Prompt > cd Documents
Prompt > ls

Prompt > cd ..
Prompt > ls

Admin
Guest
Jimmy
Documents
Prompt >
```

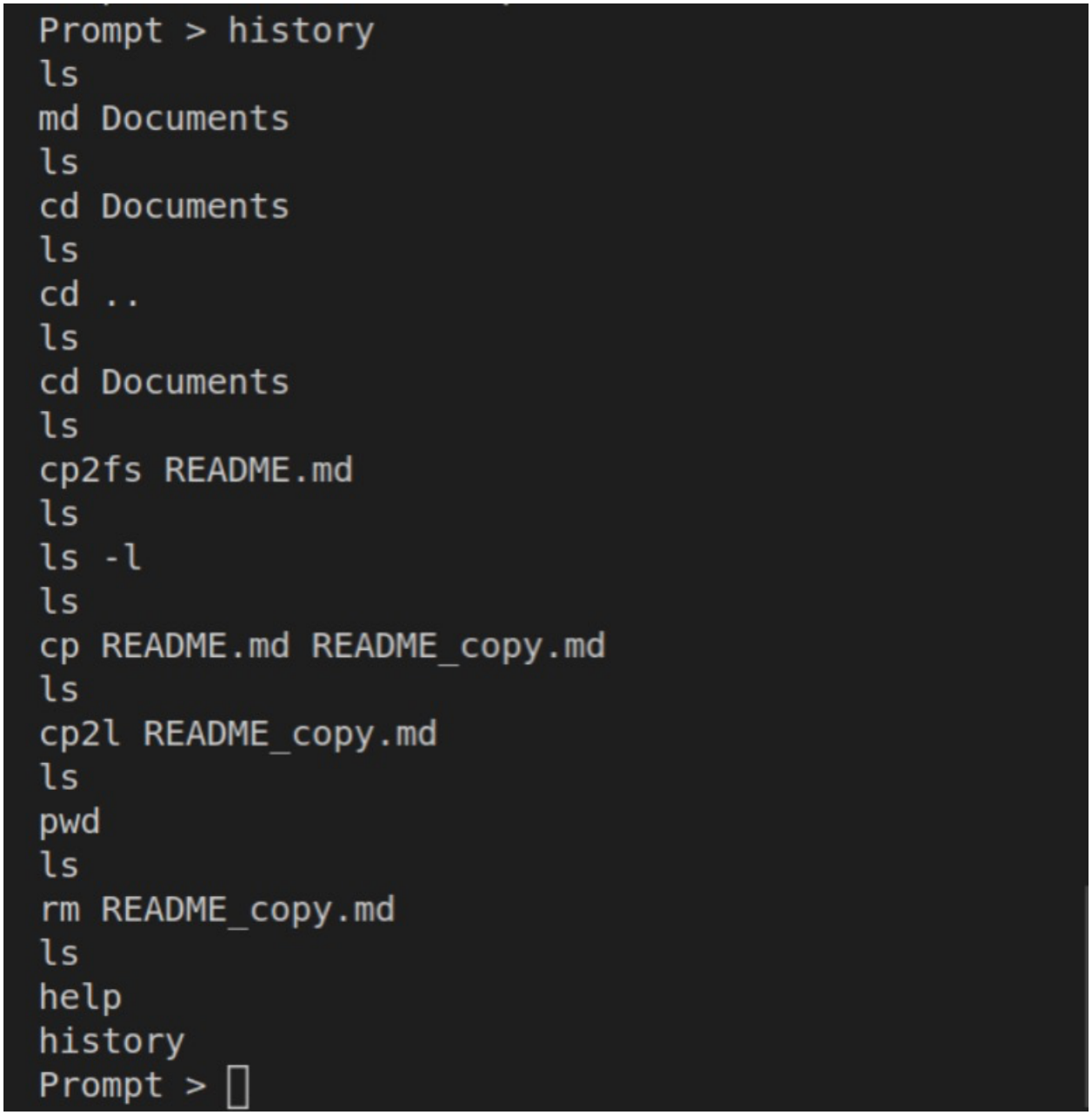
- cd /: Go back to the root directory
- cd + file name: Go to the next level file

```
Prompt > cd /
Prompt > pwd
root/
Prompt > cd Users
Prompt > pwd
root/Users
Prompt > cd Documents
Prompt > pwd
root/Users/Documents
Prompt >
```

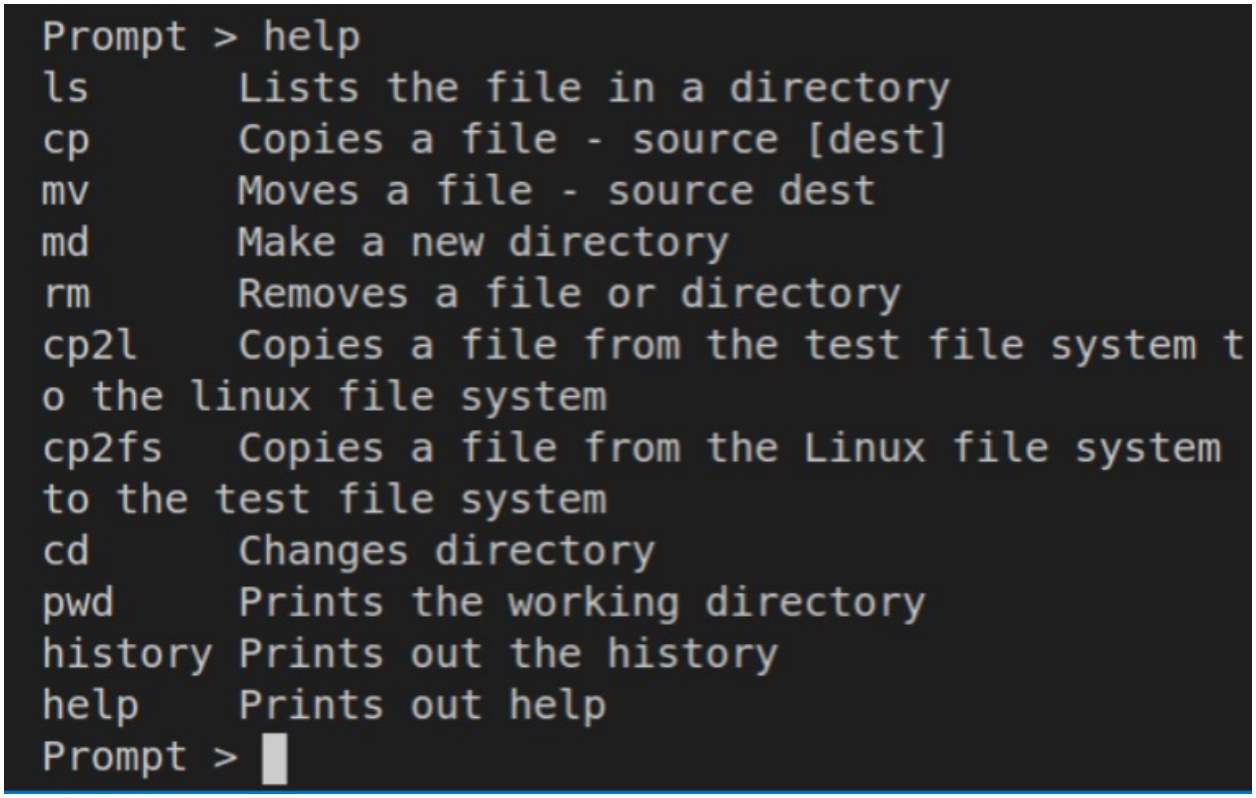
pwd - Prints the working directory

```
Prompt > pwd
root/Users/Documents
Prompt >
```

history - Prints out the history



help - Prints out 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 t
o 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 >