```
/*************************************************************
* Class:  CSC-415-01 Spring 2021
* Team Name: Survivors...
* Team Members: Annie Liao, Vivian Kuang, Elyssa Tapawan, Joanne Wong
* Student ID: 918266744, 918403595, 918459248, 918441685
* Project: Basic File System
*
* File: file_system_writeup
*
* Description: This is the writeup for our file system project.
*
*************************************************************/
```

The github link for your group submission:
https://github.com/CSC415-Spring2021/filesystemproject-anliao2

A description of your file system:
VCB, stores information needed about volume. There is a init_VCB_blk that will first check if the volume already exists by calling getval. In getval, we check to see if the volume is there and if the volume has our signature. If it does, then we LBAread and set the data to the structs we need. If not, it returns to the init_VCB_blk to execute the rest of the initialization process, setting values in the vcb struct, initializing bitmap, and initializing the directory array.

bitmap, store 0's and 1's. 0's are free spaces and 1's are taken. It's more of a char array than a map. Each char is 1 byte which is 8 bits so to determine the block number, the index of the char array is multiplied by 8 then it's added with the inner index of the bytes which goes from 0 to 7 because 8 bits. Affected by b_open, b_write, b_close. When blocks are freed it's the same process but the bit is unset starting at the index given for the count given in the parameter.

dea (directory entry array), stores directory entries. Affected by b_open, b_write, b_close. The first function for the initialization of the directory array in createDir. It is allocated 10 blocks and there the dea array is made and the child and parent is set. Any other directory entries made goes through makede. It takes in a name, index, size, file or directory indicator, and the name of the current directory. If the ford (file or directory) is 0 it's a file, if it's a 1 it's a directory. There is a loop that goes through the dea array to find an open spot to create a directory entry and the values of the directory entry are set.

Issues you had

Issue	Solution

How to get struct values to travel between c files?	Tried to create a function that would return all the values needed like a set and get for each variable. It took too many lines, thus in the end used extern from https://stackoverflow.com/questions/31284425/how-do-i-share-a-dynamically-allocated-array-between-programs-in-c/31288047
How to check if our volume already exists if startPartition returns errors and not if the volume exists?	Tried to mess with fsLow. Made a function within fsLow that would return a number to let us know it exists, but then remembered we shouldn't mess with the fsLow so created a function in VCB that would be called in the init_VCB_blk in the beginning so that if it already exists, we don't need to init values, we would LBAread the blocks and set the structs to those values from the existing volume. In the end, we did not change anything fsLow. We did have some printf statements but removed those.
Malloc/free issues	Was freeing things that shouldn't be freed and mallocing things that would hold the same memory as another. Tried to decrease the use of malloc and only use malloc when it was necessary and kept track of when and how long we needed the information and when we would not need it to free it.
Adding more blocks to a file	Multiply the file block size by 2 each time it is encountered before LBAwrite that there is not enough blocks. Then in b_close, unused blocks will be freed.
Math issues	Doing math/calculations wrong, which affected indexes of blocks.

Detail of how your driver program works
Starts in the main of fsshell.c. We take the name, volume size, and block size through the parameters of main. Our initialized volume size is 10,000,000. From there, we call startPartitionSystem from fsLow and after that we call init_VCB_blk to either init the VCB if it doesn't exist or to set the values if it does exist.
After the setup, we start taking commands. Once a command is given, we execute that command until it's finished before taking the next command.

When the command is exit, we call close_vcb in VCB and closePartitionSystem to free the malloced memory used for the structs.

The commands of the fsshell calls on functions in the mfs and b_io.

Fsshell.c:
cmd_mv, moves a file/directory to source [dest]. The usage command would be: mv src dest. For the mv command, we can move a directory or file into a directory. It is not possible to move a

directory for example into an existing text file. First I check if the src exists. We can't move anything if the src doesn't exist, so return -1 if it does not exist. Then, I check if dest is a file and if it exists in the current working directory. If it is a file (like text file) and it exists, we return -1. If it is not a file and the dest name does not exist in the current directory, we can move the src to the dest. The destination does not need to exist. For example, we have newDir1 and I want to do mv newDir1 newDir2 but newDir2 does not exist in the current working directory. All we have to do really is rename newDir1 to newDir2. We are basically moving newDir1 as newDir2. Now, let's say the current working directory has 2 directories newDir3 and newDir4. We do mv newDir3 newDir4 and this would move the directory newDir3 into newDir4. This of course also works with files. So if we have a text file test.txt, we do mv test.txt newDir3 which moves test.txt to newDir3. Test.txt will no longer be in the current working directory as it will be inside newDir3.

b_io.c:
b_open, finds if the file already exists. If it does, then the struct in the b_io.c is filled. If it doesn't exist, then space would have to be allocated and the dea and bitmap has to be updated then those values will be set to the b_io.c struct. The fd returned will be the index of the b_io.c struct array.

b_read, using the fd in parameter, get the values that were set in b_open. Loop to copy the data from volume to the parameter buffer. LBAread the data of that file at the saved didx, and the index of the dea, to get the size of the file. It will do LBAread when there is no data in the buffer and when there are still blocks to read from. It will continue to copy until either the count requested has been reached or the blk_count and file_length are equal and/or bufferdata is zero, that means that we are at the end of the file.

b_write, using the fd in parameter, get the values that were set in b_open. Loop to copy the data from the parameter buffer to the struct in b_io.c buffer. Once the count has been reached it will exit the loop. If the bufferdata has reached capacity, then it will be written using LBAwrite, but before that happens, we must check if there are enough blocks in the file to write to. If there are not enough blocks, then the file_length will be doubled and that value has to be updated in the dea, b_io.c struct and the bitmap. There is a condition if b_write is called with count 0, that means that b_close is calling b_write and it means that there is still stuff in the buffer that hasn't been written yet so it needs to be written with LBAwrite.

b_close, using the fd, we free the buffer in the struct at the fd index of the array, but before we do that, we check to see if there is data that needs to be written and check if there are unused blocks that need to be freed.

b_seek, using the fd, we determine which directive whence is passed in to perform the proper offset operation. If whence is SEEK_SET, we set the file's file index to be that of the offset. If whence was SEEK_CUR, we set the file's file index to be the current position plus the offset. If whence was SEEK_END, we set the file's file index to be the size of the file plus the offset. We return the file's file index to get the resulting offset location.

mfs.c

fs_mkdir, goes through dea to find if a directory entry has the same name already. If not, then the directory entry will be created.

fs_rmdir, In fs_rmdir, we are removing a directory and inside that directory, there can also be multiple directories so we also have to remove those directories as well. I made a new char variable called currentDir which stands for the current working directory (pwd). Everytime a directory is made, we save the current working directory into the directory entry struct so in order to remove directories within a directory, I just need to check the currentDir variable. For example, our current working directory (pwd) is . and inside ., we have a directory newDir which I want to remove. Inside newDir, there is newDir1 and inside newDir1, there is newDir2. We want to remove all of those directories inside newDir. The currentDir for newDir is . and the currentDir for newDir1 is ./newDir and currentDir for newDir2 is ./newDir/newDir1. In order to remove newDir1 and newDir2, I just make 2 substrings ./newDir and ./newDir/ in which I will use to compare all the currentDir saved in every member of the directory entry struct. newDir2 has currentDir as ./newDir/newDir1 but my substring is ./newDir/ so I just need to cut every currentDir and see if it matches with ./newDir/ and remove it from that directory.

fs_opendir, In fs_opendir, we are going through the directory entry array (struct) to search for directory entries that match the currentDir name. The currentDir variable means the current working directory (pwd). When a directory or file is made, we save the current working directory name which the directory was made into a char variable currentDir. This helps us easily track what files or directories are currently in the currently working directory in which we want to see for the ls command.

fs_readdir, takes the fdDir from the parameter and sets values to fs_diriteminfo struct to be returned. We are currently listing all the directory entry names (files and directories) that are in the current working directory

fs_closedir, free the fdDir in parameter and set to NULL.

fs_getcwd, returns the currentDir variable, the current working directory

fs_setcwd, fs_setcwd is used for cd and when we use cd, we change the currentDir variable to the current working directory that we just changed to. First, we have to check if the directory exists or not and it can not be like a text file. We can only change to directories. For example, the current working directory is ./newDir and inside ./newDir, we have newDir1. We want to cd newDir1. I simply check if the directory exists in ./newDir and then if it exists, we change currentDir variable to ./newDir/newDir1. That's all I need to do. When we do cd .., we want to go back to the previous current working directory which is ./newDir. In order to do that, I just need to change currentDir to ./newDir by removing /newDir1. I used strtok for this and removed anything after the last dash.

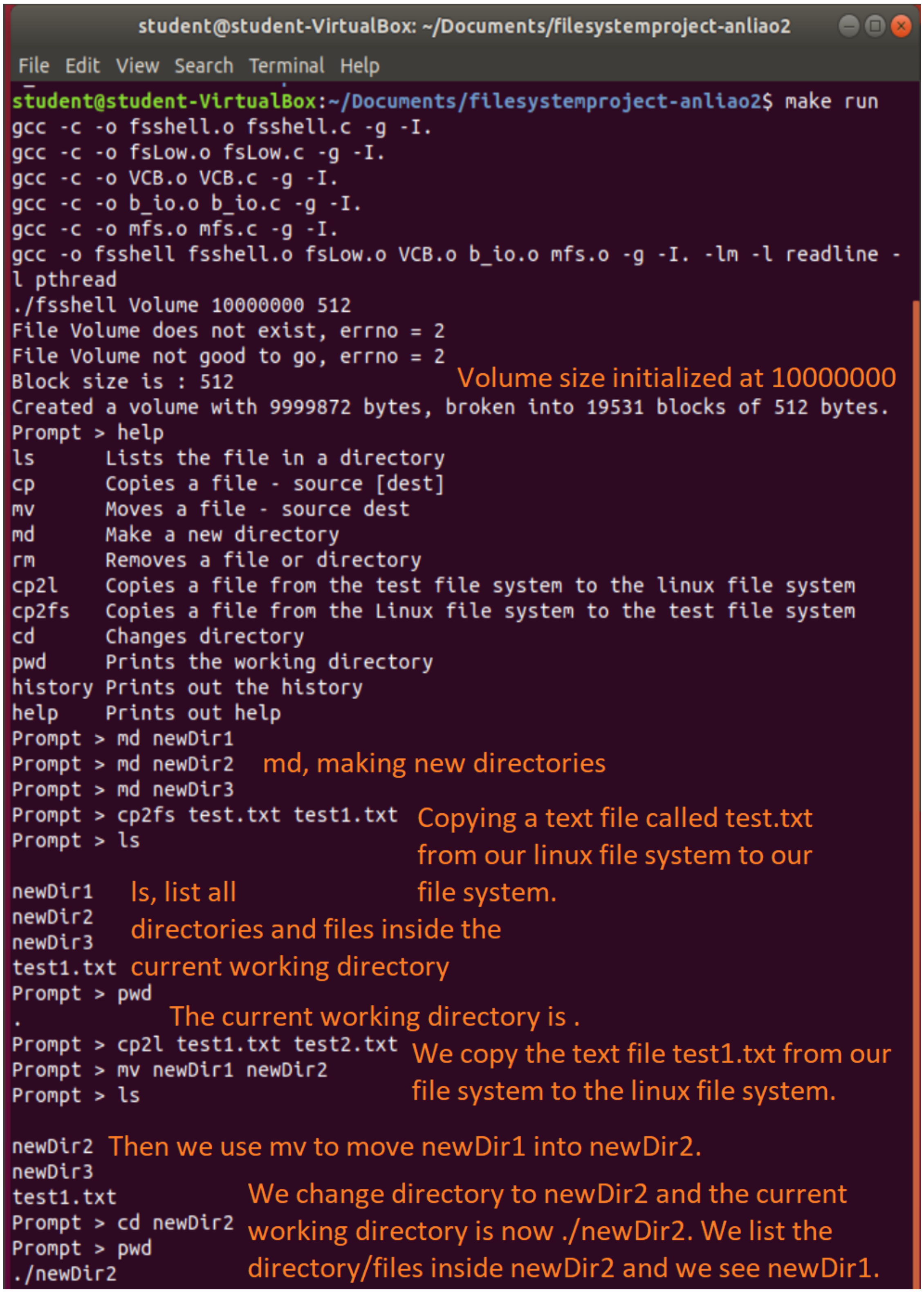
fs_isFile, goes through dea to find a matching name. Checks ford, if it's 0 then it is a file and returns 1.

fs_isDir, goes through dea to find a matching name. Checks ford, if it's 1 then it is a directory and returns 1.

fs_delete, goes through dea to find a matching name. Once found, space used is released and values set to 0, size sets to -1, and name become blank. LBAwrite changes.

fs_stat, goes through dea to find a matching name. Once found, we save the file's information to the corresponding attributes of the fs_stat struct that was passed in as a parameter.

Screenshots showing each of the commands listed in the readme



student@student-VirtualBox: ~/Documents/filesystemproject-anliao2

File Edit View Search Terminal Help

```
student@student-VirtualBox:~/Documents/filesystemproject-anliao2$ make run
gcc -c -o fsshell.o fsshell.c -g -I.
gcc -c -o fsLow.o fsLow.c -g -I.
gcc -c -o VCB.o VCB.c -g -I.
gcc -c -o b_io.o b_io.c -g -I.
gcc -c -o mfs.o mfs.c -g -I.
gcc -o fsshell fsshell.o fsLow.o VCB.o b_io.o mfs.o -g -I. -lm -l readline -
l pthread
./fsshell Volume 10000000 512
File Volume does not exist, errno = 2
File Volume not good to go, errno = 2
Block size is : 512
Created a volume with 9999872 bytes, broken into 19531 blocks of 512 bytes.
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 > md newDir1
Prompt > md newDir2
Prompt > md newDir3
Prompt > cp2fs test.txt test1.txt
Prompt > ls

newDir1
newDir2
newDir3
test1.txt
Prompt > pwd
.
Prompt > cp2l test1.txt test2.txt
Prompt > mv newDir1 newDir2
Prompt > ls

newDir2
newDir3
test1.txt
Prompt > cd newDir2
Prompt > pwd
./newDir2
```

Volume size initialized at 10000000

md, making new directories

Copying a text file called test.txt from our linux file system to our file system.

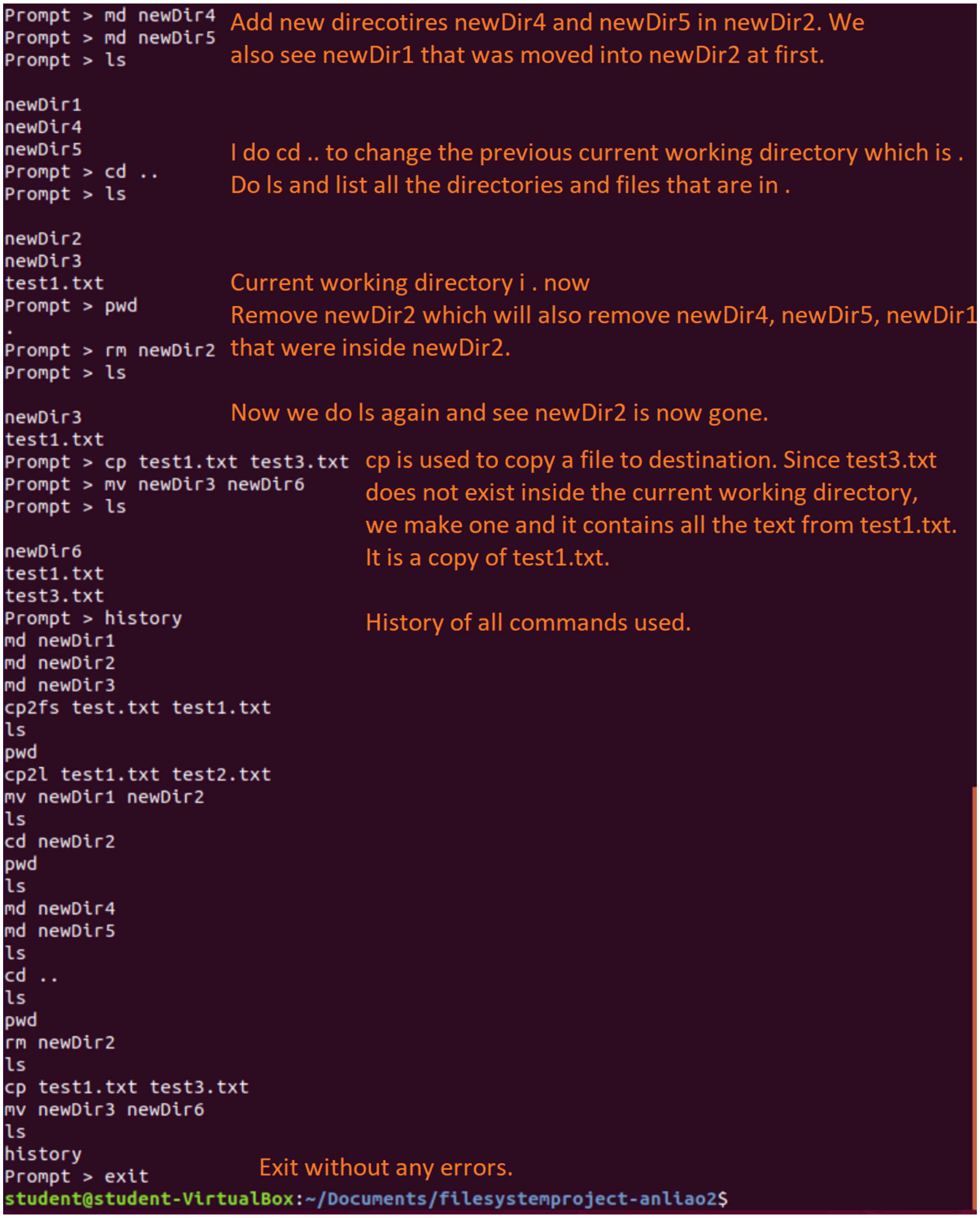
ls, list all directories and files inside the current working directory

The current working directory is .

We copy the text file test1.txt from our file system to the linux file system.

Then we use mv to move newDir1 into newDir2.

We change directory to newDir2 and the current working directory is now ./newDir2. We list the directory/files inside newDir2 and we see newDir1.



```
Prompt > md newDir4
Prompt > md newDir5
Prompt > ls

newDir1
newDir4
newDir5
Prompt > cd ..
Prompt > ls

newDir2
newDir3
test1.txt
Prompt > pwd
.
Prompt > rm newDir2
Prompt > ls

newDir3
test1.txt
Prompt > cp test1.txt test3.txt
Prompt > mv newDir3 newDir6
Prompt > ls

newDir6
test1.txt
test3.txt
Prompt > history
md newDir1
md newDir2
md newDir3
cp2fs test.txt test1.txt
ls
pwd
cp2l test1.txt test2.txt
mv newDir1 newDir2
ls
cd newDir2
pwd
ls
md newDir4
md newDir5
ls
cd ..
ls
pwd
rm newDir2
ls
cp test1.txt test3.txt
mv newDir3 newDir6
ls
history
Prompt > exit
student@student-VirtualBox:~/Documents/filesystemproject-anliao2$
```

Add new direcotires newDir4 and newDir5 in newDir2. We also see newDir1 that was moved into newDir2 at first.

I do cd .. to change the previous current working directory which is . Do ls and list all the directories and files that are in .

Current working directory i . now

Remove newDir2 which will also remove newDir4, newDir5, newDir1 that were inside newDir2.

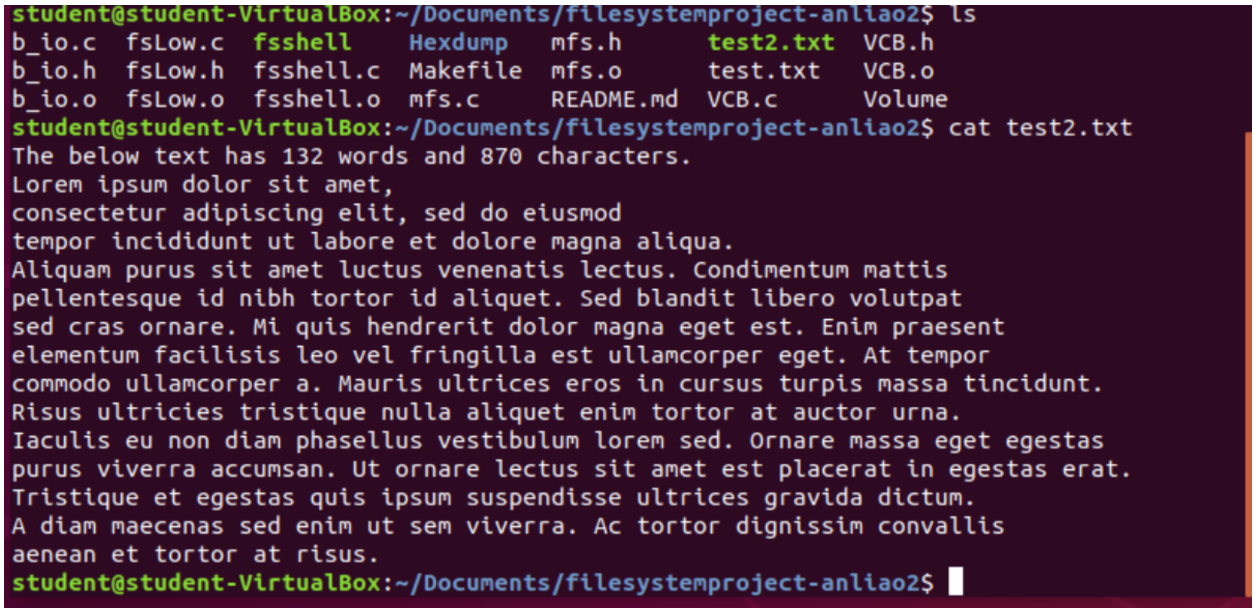
Now we do ls again and see newDir2 is now gone.

cp is used to copy a file to destination. Since test3.txt does not exist inside the current working directory, we make one and it contains all the text from test1.txt. It is a copy of test1.txt.

History of all commands used.

Exit without any errors.

This is the text file when we used cp2ls test1.txt test2.txt. We copied the text file from our file system to the linux file system.



```
student@student-VirtualBox:~/Documents/filesystemproject-anliao2$ ls
b_io.c   fsLow.c   fsshell     Hexdump    mfs.h      test2.txt  VCB.h
b_io.h   fsLow.h   fsshell.c   Makefile   mfs.o      test.txt   VCB.o
b_io.o   fsLow.o   fsshell.o   mfs.c      README.md  VCB.c      Volume
student@student-VirtualBox:~/Documents/filesystemproject-anliao2$ cat test2.txt
The below text has 132 words and 870 characters.
Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua.
Aliquam purus sit amet luctus venenatis lectus. Condimentum mattis
pellentesque id nibh tortor id aliquet. Sed blandit libero volutpat
sed cras ornare. Mi quis hendrerit dolor magna eget est. Enim praesent
elementum facilisis leo vel fringilla est ullamcorper eget. At tempor
commodo ullamcorper a. Mauris ultrices eros in cursus turpis massa tincidunt.
Risus ultricies tristique nulla aliquet enim tortor at auctor urna.
Iaculis eu non diam phasellus vestibulum lorem sed. Ornare massa eget egestas
purus viverra accumsan. Ut ornare lectus sit amet est placerat in egestas erat.
Tristique et egestas quis ipsum suspendisse ultrices gravida dictum.
A diam maecenas sed enim ut sem viverra. Ac tortor dignissim convallis
aenean et tortor at risus.
student@student-VirtualBox:~/Documents/filesystemproject-anliao2$
```