Robert Swanson ID: 917106793
Kevin Islas Orgaz ID: 921260116
Tom Kirke ID:
Github: csc415-filesystem-SwansonSays CSC415 Operating Systems

Group Submission- Files System Project

Link: https://github.com/CSC415-2022-Summer/csc415-filesystem-SwansonSays

Description:
For our file system we decided to use a bitmap in order to keep track of the free space. The bitmap would mark whether a block was used or unused. We also used contiguous allocation in the file system. The shell program implements the functions required such as ls along with any other flags that might follow the command. In our volume control block we are checking for the size of the block, the total amount of blocks, the number of free blocks, the location of the bitmap, the total amount of blocks in a bitmap, location to our root directory and our custom signature to identify the volume control block. The signature inside our volume control block is a reference to our name. As for our directory entries we kept track of the time it was created, when it was modified, the size, the location, whether it is a directory, if it was in use and the name of the entry.

Issues and Resolutions:

The first issue was checking to see if the volume control block. In milestone one when we were initializing the VCB we ran into issues when doing the malloc for the volume control block. Initially we were using sizeOf the fdDir structure.
It was resolved it by removing sizeOf the fdDir structure and instead we used the block size to malloc the volume control block.

Another issue we had was a github issue. When pulling there was some conflicts which caused some things to get deleted or overwritten. Luckily, we had a few backups so it was squared away pretty fast.

An issue that took quite a bit of time to resolve was with the ls command. Whenever ls was called it would give us a segmentation fault. This was an issue caused by the readDir function. The problem seemed to be with the pointer to directory entry. It had not been initialized and was not filling the directory item info structure with anything. After doing a malloc for the pointer to the directory entry things seemed to work better.

There was an issue regarding the fs_rmdir function. We were using strtok but it was not keeping the string passed into together. Since this function was checking if what was being passed was a directory it was changing the path string. One solution that we thought might work was changing strtok to strtok but it was still doing the same thing. The solution we implemented was copying the path into a buffer and then pass that buffer into the parsedPath function.

One of the more extensive problems was related to parsedPath. At the beginning of writing, it there was some confusion as to how to load the root into the memory. In other to solve this, a function was created that would malloc a pointer to the dirEntry structure, perform and LBAread on the pointer and then return the pointer. Overall find the answers to these errors did end up taking a lot of time and a lot of print statements.

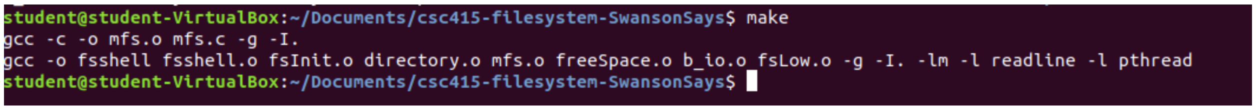
Driver program explanation:
The driver program is waiting for user input. Once the user input is read and matches one of the available commands it calls to our msf.c file. If the user enters ls, the call from the driver program would go to the fs_openDir where the path of the directory will be parsed, loaded, and will be checked to make sure the path is for a directory. Then the opendir function will create a pointer to the fdDir structure and malloc the size of it. A pointer to directory entry will also be malloced to the size of the directory. From there we initialize the things inside the fdDir structure such as the current directory, the index and whether or not it is in use. Then we return the pointer to all of this. The readdir fills in all the directory item information. It is called when a user enters ls. It is used for getting the information that is displayed. The touch command relies on our b_open. The file can be written into or if the file does not exist it can be created. The copy command also uses b_open and the source file can only be read while the destination file can be read, created, or truncated. It also requires access to b_read. The move command has its on function, so when a user enters mv, the fsshell.c will call moveDirEntry inside of mfs.c where a pointer to the filepath will be strcpy. The path will also be parsed and both the destination and the source will be added to the parsed path. Once it is done the pointer will be freed. After this is done the pointer will find an available spot in the parent directory of the destination and set the source files inside it. Following this the parent directory will be written to disk. The make directory has its own function that creates a new directory at specified path. The remove directory command assesses if the path provided is a directory or a file. Each option will lead to its own function where the directory will be removed or the file is deleted. The cd command sends the path to the set current working directory function where the path will be parsed and copied. Then the current work directory will be set using pointers to the parsed path structure.

Screen shot of compilation:

Make sure it is easily readable (i.e. do not cram lots of screen shots on a single age) and that it includes the command and the complete compilation output of gcc. There should be no warnings or errors.

This should be in the Linux Terminal window and not visual studio.
Example:

Robert Swanson ID: 917106793
Kevin Islas Orgaz ID: 921260116
Tom Kirke ID:
Github: csc415-filesystem-SwansonSays CSC415 Operating Systems

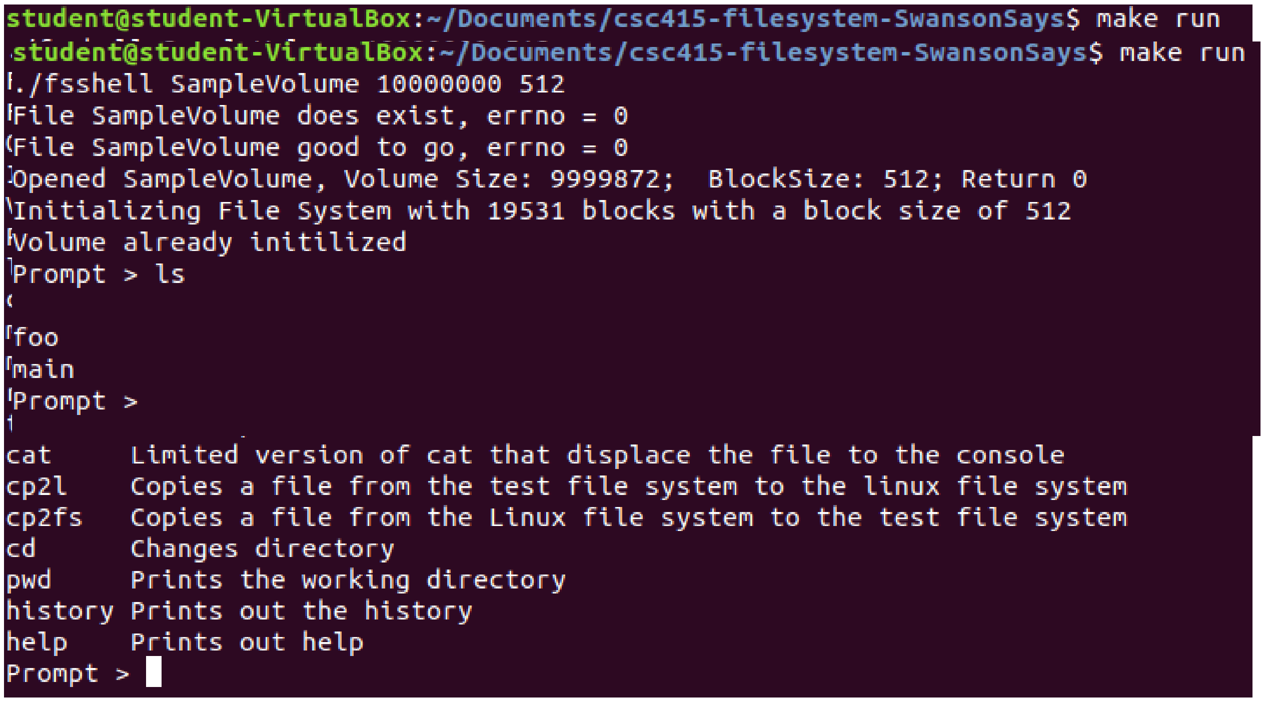


```
student@student-VirtualBox:~/Documents/csc415-filesystem-SwansonSays$ make
gcc -c -o mfs.o mfs.c -g -I.
gcc -o fsshell fsshell.o fsInit.o directory.o mfs.o freeSpace.o b_io.o fsLow.o -g -I. -lm -l readline -l pthread
student@student-VirtualBox:~/Documents/csc415-filesystem-SwansonSays$ 
```

Screen shot(s) of the execution of the program:

Show all necessary screen shots (some assignments require more than one).
These should be in the Linux Terminal window and not visual studio.

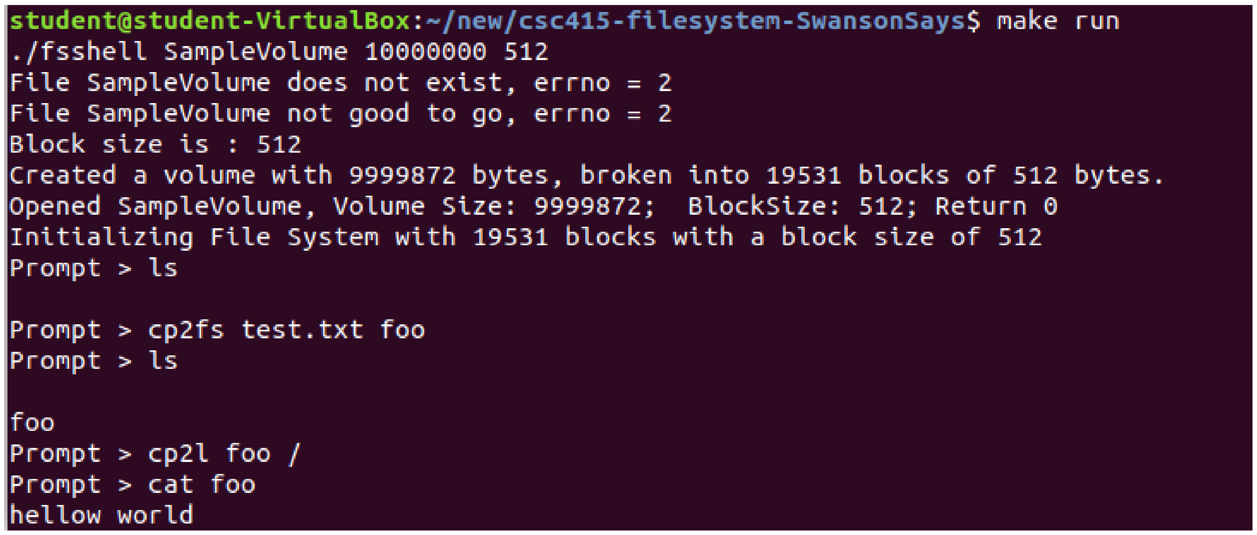
ls



```
student@student-VirtualBox:~/Documents/csc415-filesystem-SwansonSays$ make run
student@student-VirtualBox:~/Documents/csc415-filesystem-SwansonSays$ make run
./fsshell SampleVolume 10000000 512
File SampleVolume does exist, errno = 0
File SampleVolume good to go, errno = 0
Opened SampleVolume, Volume Size: 9999872;  BlockSize: 512; Return 0
Initializing File System with 19531 blocks with a block size of 512
Volume already initilized
Prompt > ls

foo
main
Prompt >

cat     Limited version of cat that displace the file to the console
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 > 
```

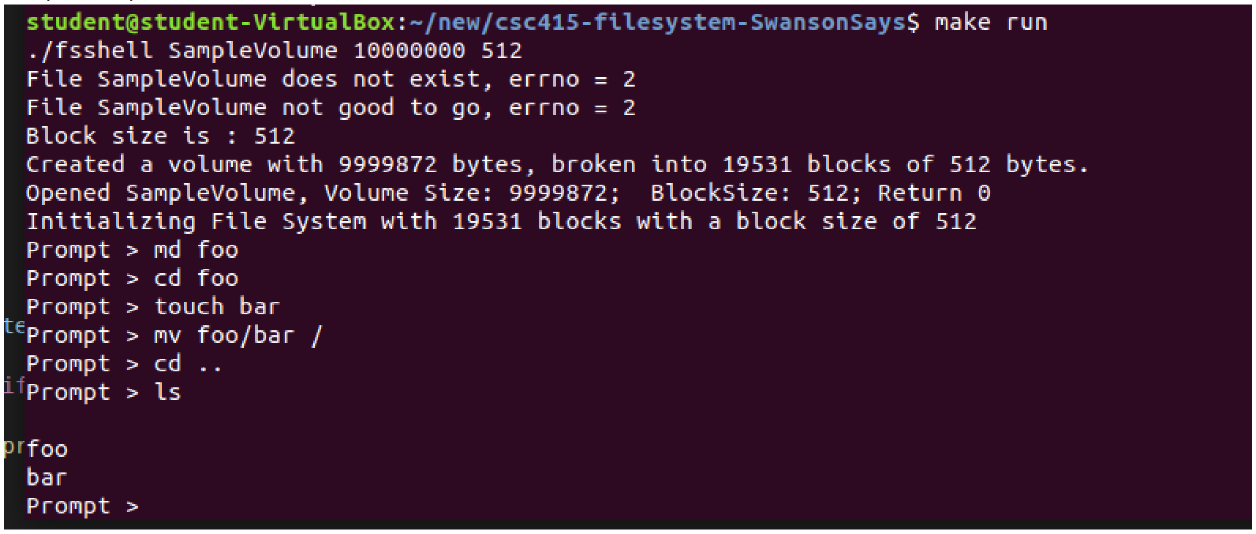
Robert Swanson ID: 917106793
Kevin Islas Orgaz ID: 921260116
Tom Kirke ID:
Github: csc415-filesystem-SwansonSays CSC415 Operating Systems

cp2fs,cp2l,cat



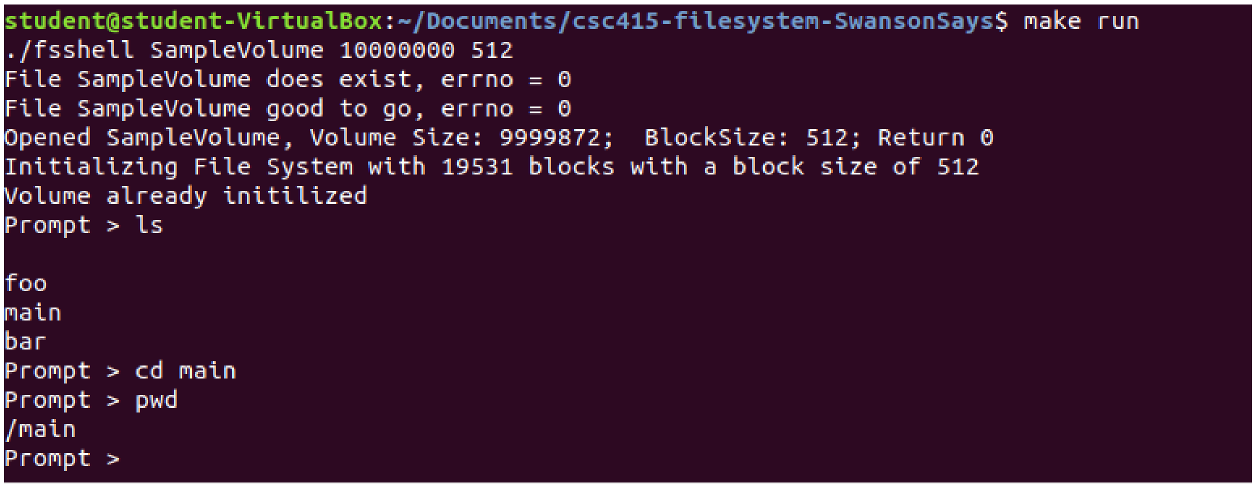
```
student@student-VirtualBox:~/new/csc415-filesystem-SwansonSays$ make run
./fsshell SampleVolume 10000000 512
File SampleVolume does not exist, errno = 2
File SampleVolume not good to go, errno = 2
Block size is : 512
Created a volume with 9999872 bytes, broken into 19531 blocks of 512 bytes.
Opened SampleVolume, Volume Size: 9999872;  BlockSize: 512; Return 0
Initializing File System with 19531 blocks with a block size of 512
Prompt > ls

Prompt > cp2fs test.txt foo
Prompt > ls

foo
Prompt > cp2l foo /
Prompt > cat foo
hellow world
```

md,touch,mv



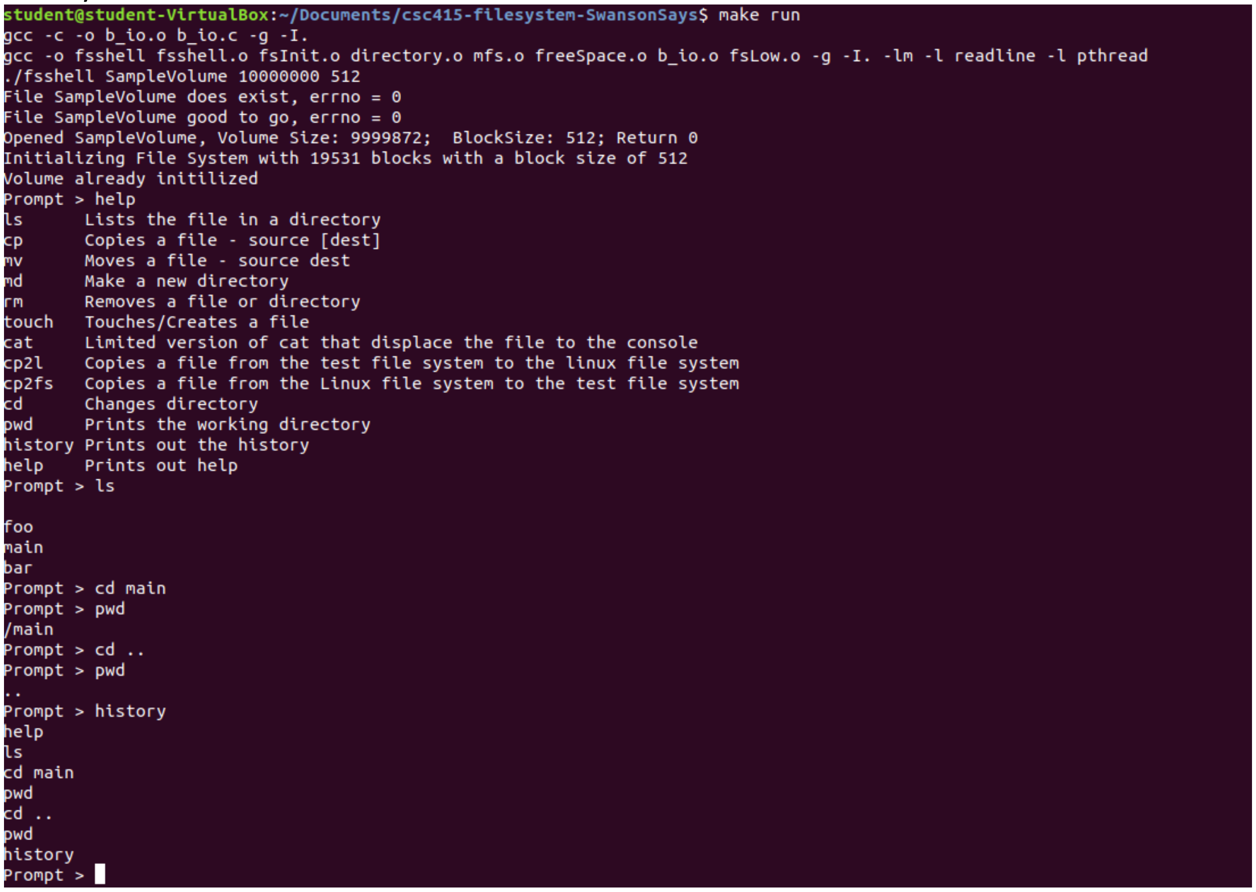
```
student@student-VirtualBox:~/new/csc415-filesystem-SwansonSays$ make run
./fsshell SampleVolume 10000000 512
File SampleVolume does not exist, errno = 2
File SampleVolume not good to go, errno = 2
Block size is : 512
Created a volume with 9999872 bytes, broken into 19531 blocks of 512 bytes.
Opened SampleVolume, Volume Size: 9999872;  BlockSize: 512; Return 0
Initializing File System with 19531 blocks with a block size of 512
Prompt > md foo
Prompt > cd foo
Prompt > touch bar
Prompt > mv foo/bar /
Prompt > cd ..
Prompt > ls

foo
bar
Prompt >
```

PWD



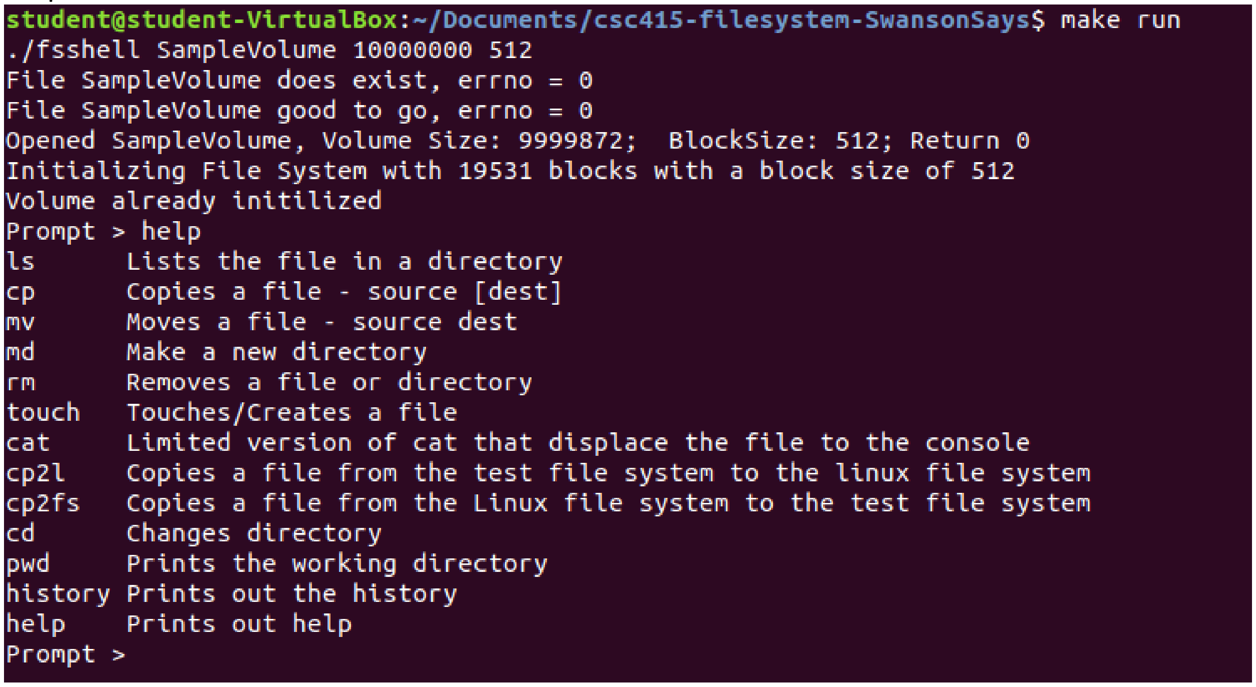
```
student@student-VirtualBox:~/Documents/csc415-filesystem-SwansonSays$ make run
./fsshell SampleVolume 10000000 512
File SampleVolume does exist, errno = 0
File SampleVolume good to go, errno = 0
Opened SampleVolume, Volume Size: 9999872;  BlockSize: 512; Return 0
Initializing File System with 19531 blocks with a block size of 512
Volume already initilized
Prompt > ls

foo
main
bar
Prompt > cd main
Prompt > pwd
/main
Prompt >
```

history



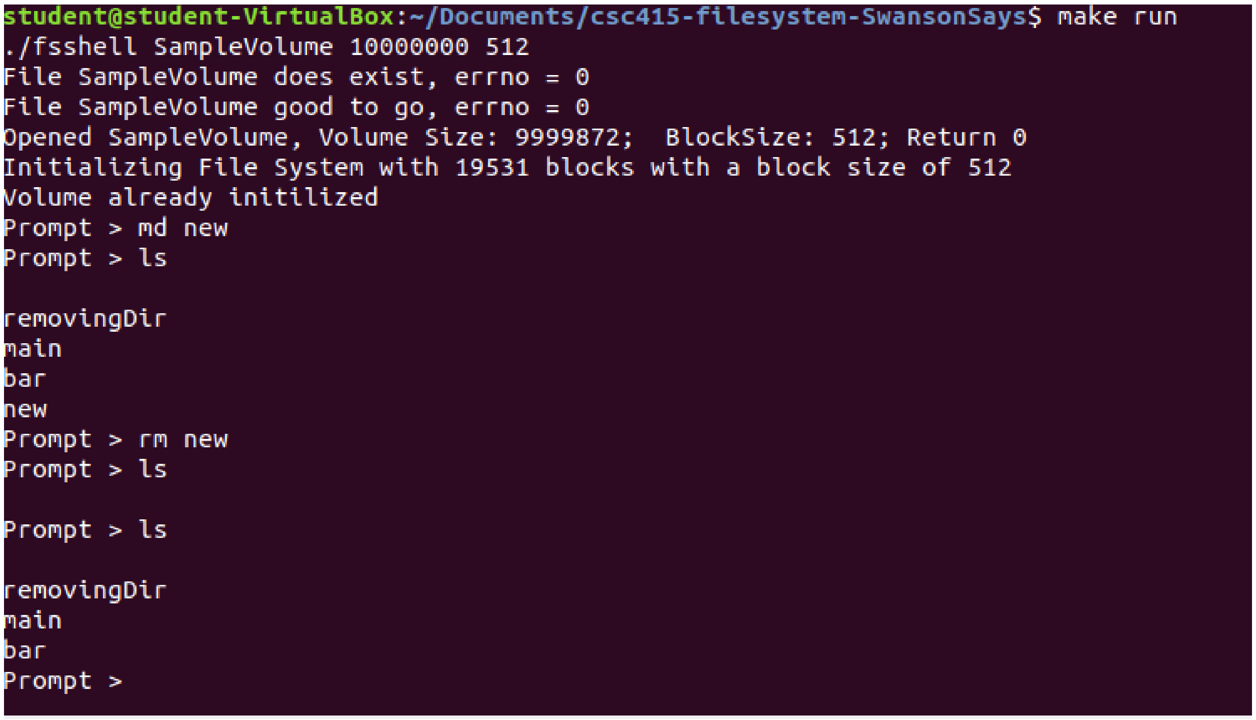
```
student@student-VirtualBox:~/Documents/csc415-filesystem-SwansonSays$ make run
gcc -c -o b_io.o b_io.c -g -I.
gcc -o fsshell fsshell.o fsInit.o directory.o mfs.o freeSpace.o b_io.o fsLow.o -g -I. -lm -l readline -l pthread
./fsshell SampleVolume 10000000 512
File SampleVolume does exist, errno = 0
File SampleVolume good to go, errno = 0
Opened SampleVolume, Volume Size: 9999872;  BlockSize: 512; Return 0
Initializing File System with 19531 blocks with a block size of 512
Volume already initilized
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
touch   Touches/Creates a file
cat     Limited version of cat that displace the file to the console
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 > ls

foo
main
bar
Prompt > cd main
Prompt > pwd
/main
Prompt > cd ..
Prompt > pwd
..
Prompt > history
help
ls
cd main
pwd
cd ..
pwd
history
Prompt >
```

Help



```
student@student-VirtualBox:~/Documents/csc415-filesystem-SwansonSays$ make run
./fsshell SampleVolume 10000000 512
File SampleVolume does exist, errno = 0
File SampleVolume good to go, errno = 0
Opened SampleVolume, Volume Size: 9999872;  BlockSize: 512; Return 0
Initializing File System with 19531 blocks with a block size of 512
Volume already initilized
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
touch   Touches/Creates a file
cat     Limited version of cat that displace the file to the console
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 >
```

rm



```
student@student-VirtualBox:~/Documents/csc415-filesystem-SwansonSays$ make run
./fsshell SampleVolume 10000000 512
File SampleVolume does exist, errno = 0
File SampleVolume good to go, errno = 0
Opened SampleVolume, Volume Size: 9999872;  BlockSize: 512; Return 0
Initializing File System with 19531 blocks with a block size of 512
Volume already initilized
Prompt > md new
Prompt > ls

removingDir
main
bar
new
Prompt > rm new
Prompt > ls

Prompt > ls

removingDir
main
bar
Prompt >
```