

File System Milestone One

James Dixon, Justin Hellweg, Devon Huang, Omar Thiongane

Student IDs: 922440659, 922692586, 916940666, 922164097

GitHub Names: JD499, Jus1927, Novedh, ot409

GitHub Link: <https://github.com/CSC415-2024-Spring/csc415-filessystem-Novedh>

Table of Contents

File System Milestone One.....	1
Table of Contents.....	2
Description of File System.....	3
<i>Description of the VCB Structure.....</i>	3
<i>Description of the Free Space Structure.....</i>	3
<i>Description of the Directory System.....</i>	3
Our Functions.....	3
<i>int initFreeSpaceMap.....</i>	3
<i>int allocateBlocks.....</i>	3
<i>int createDirectory.....</i>	4
Dump.....	4
Plan for Each Phase.....	6
<i>Phase One Plan.....</i>	6
Remaining Phases.....	6
Screenshots of the Shell Commands.....	8
Teamwork and Division of Tasks.....	8
<i>Teamwork Description.....</i>	9
Issues & Resolutions.....	10
<i>Issue & Resolution 1:.....</i>	10

Description of File System

Description of the VCB Structure

The VCB is the Volume control Block which contains information such as signature, the number of blocks in the volume, block size, when free space starts and when the root directory starts.

Description of the Free Space Structure

Responsible for managing and allocating free blocks in the file system. Using setBit, clearBit, and getBit to manipulate bits in the bitmap where each bit represents a block of 512 bytes.

Description of the Directory System

The Directory system contains information about a file or directory in the file system. It includes name of directory, directory entry, location on disk, size, and a flag indicating if it is a directory or not.

Our Functions

int initFreeSpaceMap

Responsible for initializing the free space map. Calculates the number of bytes needed to represent the free space. Allocates memory for the freeSpaceMap array and initializes it by setting all bits to 0. Then marks the blocks that were used for the free space map as used and writes it to storage.

int allocateBlocks

Allocates a specific number of contiguous blocks. It searches the free space map for free blocks and when one is found it increments the count of blocks to see if it has enough to match

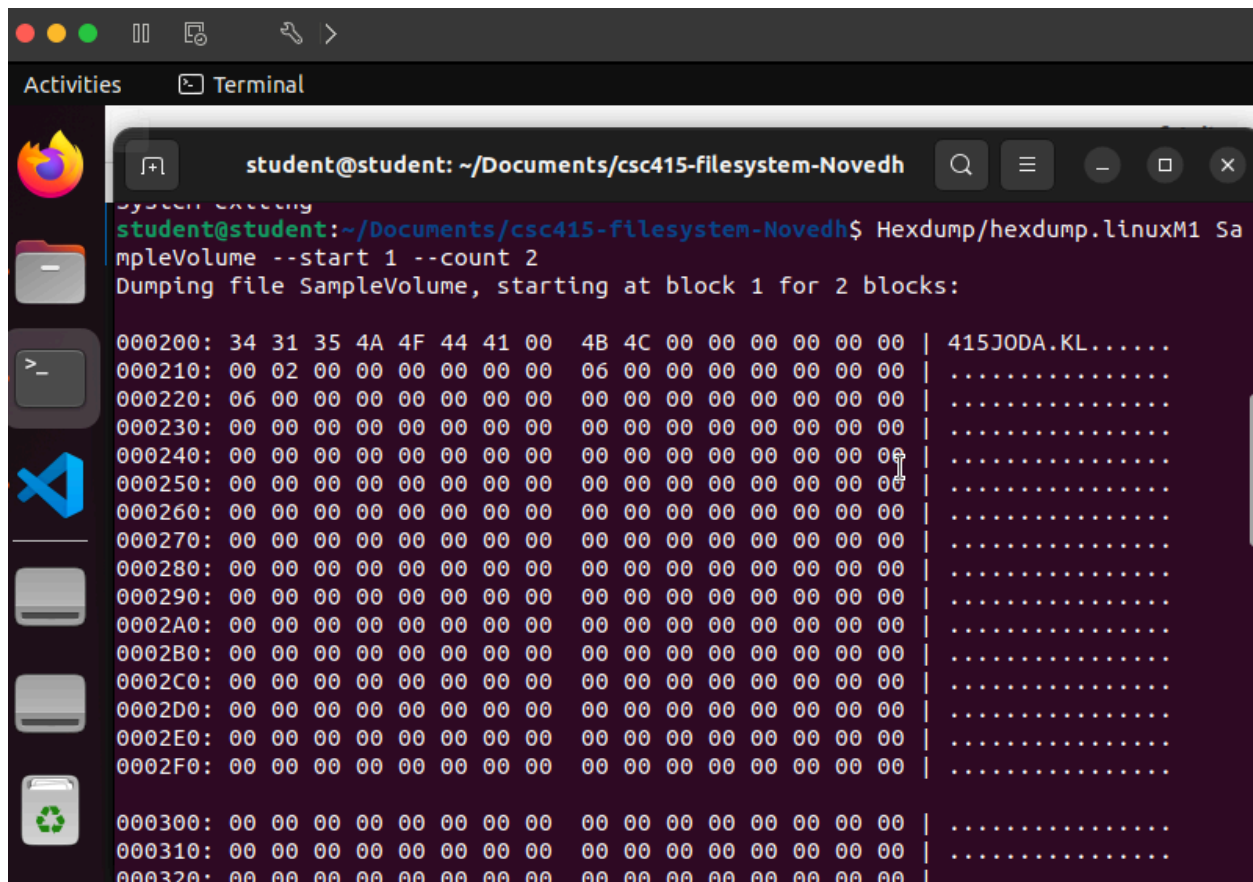
the requested number of blocks. If it does then it sets them as used and writes the updated free space map onto disk.

int createDirectory

Creates a new directory which contains the actual amount of entries that can fit in the blocks given the minimum number of entries given from the parameters. If there is no parent then it means that it is creating a new root directory and initializing it then writing it to disk.

Dump

//show a dump (using provided HexDump utility) that show VCB, Free Space, and complete root directory

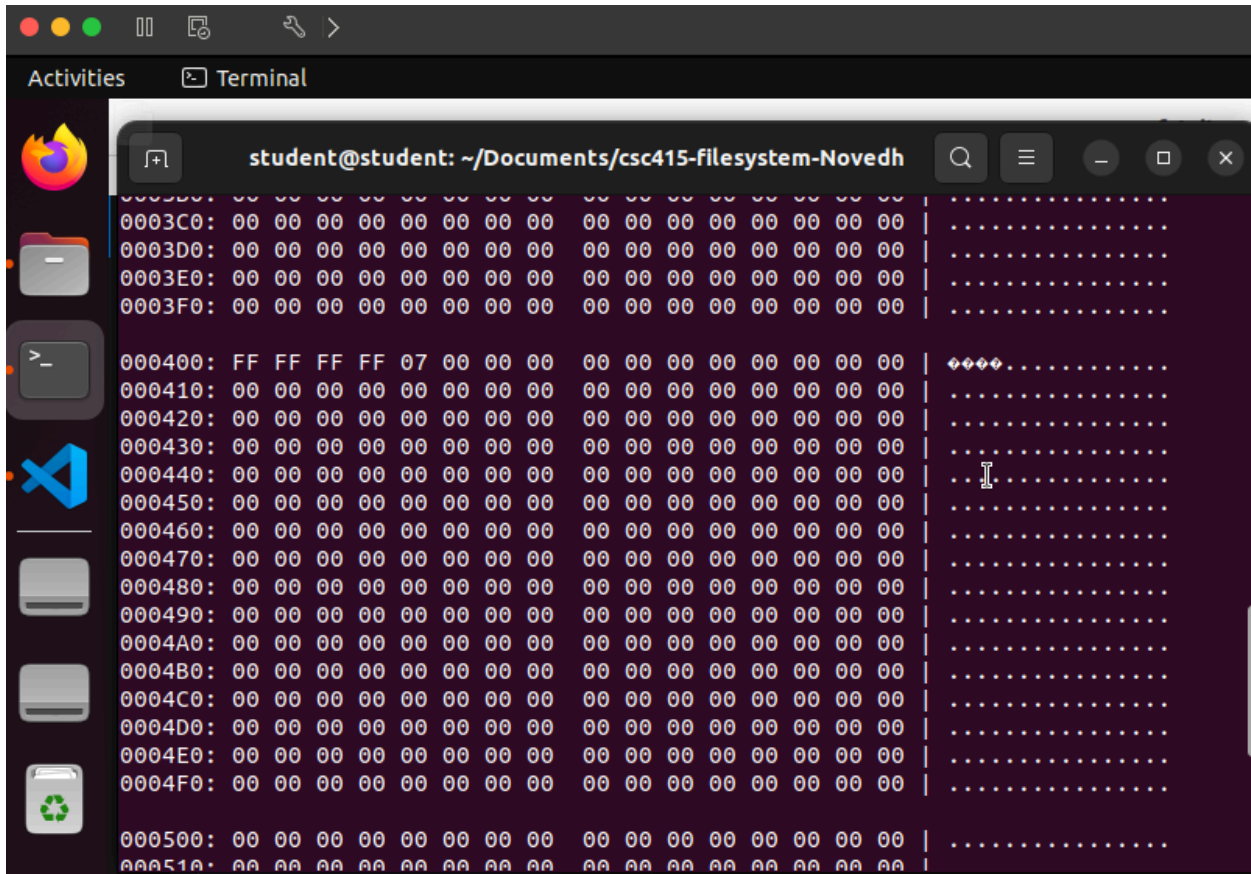


```
student@student: ~/Documents/csc415-filesystem-Novedh
student@student:~/Documents/csc415-filesystem-Novedh$ Hexdump/hexdump.linuxM1 Sa
mpleVolume --start 1 --count 2
Dumping file SampleVolume, starting at block 1 for 2 blocks:

000200: 34 31 35 4A 4F 44 41 00  4B 4C 00 00 00 00 00 00 | 415JODA.KL.....
000210: 00 02 00 00 00 00 00 00  06 00 00 00 00 00 00 00 | .....
000220: 06 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | .....
000230: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | .....
000240: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | .....
000250: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | .....
000260: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | .....
000270: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | .....
000280: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | .....
000290: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | .....
0002A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | .....
0002B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | .....
0002C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | .....
0002D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | .....
0002E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | .....
0002F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | .....

000300: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | .....
000310: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | .....
000320: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | .....
```

This Dump shows Block 1 which represents our VCB, 34 31 35 4A 4F 44 41 is our volume signature, which is 415JODA



This dump shows the bitmap when we convert the hex FF FF FF FF 07 to binary we get

```
11111111 11111111 11111111 11111111 00000111
```

Which shows the 35 blocks that are used

1 for the VCB

5 for the free space map

29 for the root directory

Our root directory is 29 blocks because our current DE size is 296 bytes and we chose to have 50 entries in our root directory,

$$296 * 50 = 14800$$

Then we see how many blocks we will need

$$(14800 + 511) / 512 = 29 \text{ blocks}$$

Plan for Each Phase

Phase One Plan

Following the Steps for milestone 1.pdf we started by looking at the fsInit.c file and writing down our structs for directory entries and the volume control block from our file system design group activity.

We first needed to see if the volume even needed to be formatted by reading block 0 and checking the signature. If it wasn't ours then we would have to initialize the volume.

Then we would have to initialize the free space map by finding out how many blocks we need for it and setting it to use on the free space map.

After we would have to initialize the root directory by using the same function used to create directories we can check if it has a parent if not then it would be the new root directory. We set up . and .. to point to itself.

Remaining Phases

For the remaining phases of our project, we plan to use similar strategies as we used to complete Milestone One. This includes communicating with group members on Discord, using Notion to track, breakdown, and organize the remaining tasks in the group project. The Notion page includes spaces to organize and break down tasks for every phase.



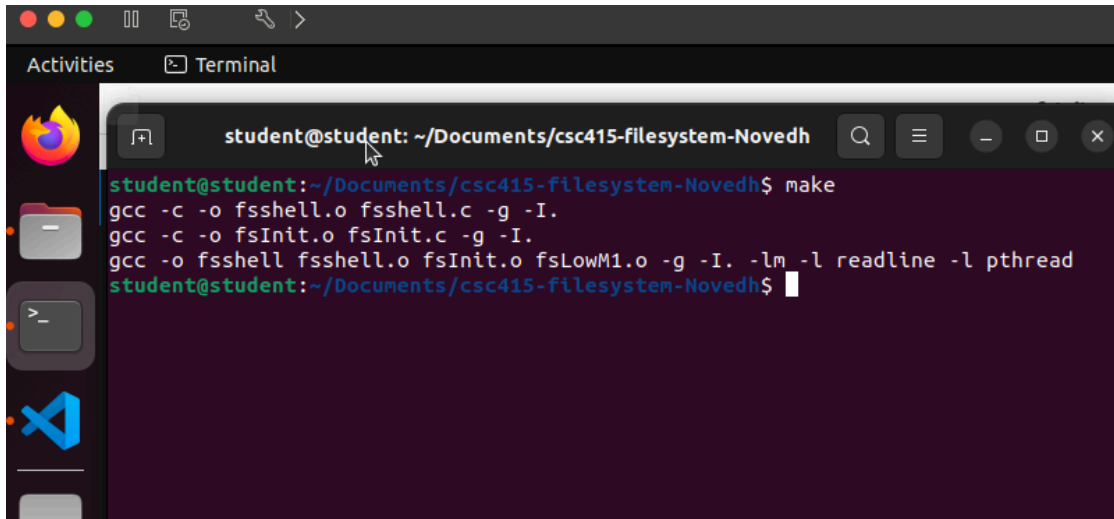
File System Group Project

- 1 Milestone One
- 2 Milestone/Phase 2: implementation of directory based functions
- ← END Final Phase/Milestone: implementation of file operations

So far, outside of a few live team meetings, we have worked mostly independently with the group Discord server available for us to reach out if we need help. Group members were good at responding and helping each other resolve our questions. Working independently includes where group members will voluntarily choose to complete certain tasks of the project as they have time, and then update other group members once it has been completed.

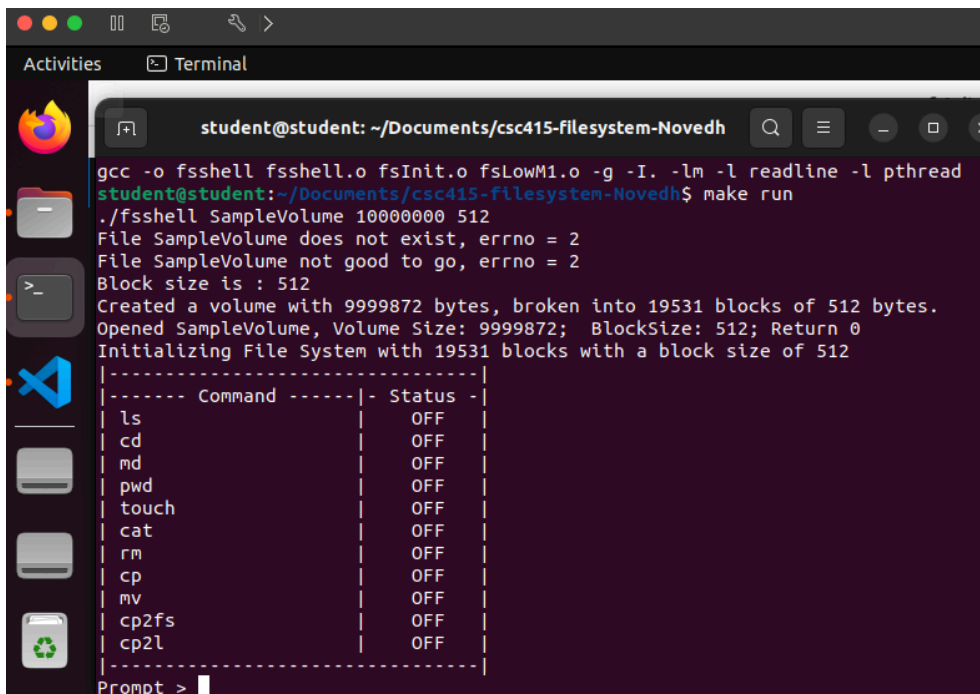
If necessary, we might see that we are running behind where would be ideal, we might need to decide on due dates to specific tasks as a group to make sure that we are able to complete the project in time. However, we will continue our current system of working on tasks when we have time, and updating the group once it's completed (or if we get stuck).

Screenshots of the Shell Commands



A terminal window titled "student@student: ~/Documents/csc415-filesystem-Novedh". The terminal shows the following commands and output:

```
student@student:~/Documents/csc415-filesystem-Novedh$ make
gcc -c -o fsshell.o fsshell.c -g -I.
gcc -c -o fsInit.o fsInit.c -g -I.
gcc -o fsshell fsshell.o fsInit.o fsLowM1.o -g -I. -lm -l readline -l pthread
student@student:~/Documents/csc415-filesystem-Novedh$
```



A terminal window titled "student@student: ~/Documents/csc415-filesystem-Novedh". The terminal shows the following commands and output:

```
gcc -o fsshell fsshell.o fsInit.o fsLowM1.o -g -I. -lm -l readline -l pthread
student@student:~/Documents/csc415-filesystem-Novedh$ 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
|----- Command -----| Status |
| ls                      | OFF   |
| cd                      | OFF   |
| md                      | OFF   |
| pwd                    | OFF   |
| touch                  | OFF   |
| cat                    | OFF   |
| rm                     | OFF   |
| cp                     | OFF   |
| mv                     | OFF   |
| cp2fs                  | OFF   |
| cp2l                   | OFF   |
|-----|-----|
Prompt >
```


Teamwork and Division of Tasks

Name	Description of Tasks Completed
James Dixon	Wrote the free space management system. Bitmap & block allocation
Justin Hellweg	Made PDF template & formatting, Notion Tracker to help organize work; wrote Teamwork Description, Remaining Phases, Issues & Resolutions section; updated headers; reviewed PDF and submitted on Canvas
Devon Huang	Wrote steps 1 and 4 of milestone 1, formatting volume and initializing root directory, Dump, Our Functions section, Phase One plan, Screenshots of Shell Commands
Omar Thiongane	

Teamwork Description

We had frequent asynchronous communication that was held primarily in a Discord Server; while the frequency of the communication varied it ranged from every other day to multiple times per day. The majority of the coding for the project was done separately, but the Discord server provided a way for team members to reach out for help with a particular step or feature that they were working on (or just ask a general question about the project). Live meetings and asynchronous chats on Discord as well as organizational tools like Notion were used to divide up the tasks.

We had several live meetings that also used Discord, so we could talk to each other in real time. These were mostly at the beginning of the project when we were organizing how we would divide up the work and made decisions about how certain features in our file system would be implemented as a group. We also had one live group meeting near the end of Milestone One to review our deliverables and make sure we were on the same page.

After working on the initial File System group submissions together, Devon took the lead in Milestone One in making the GitHub and implementing some steps of the project including the VCB and root directory. After sharing the GitHub repo with the group, James wrote the free space management system in the way that our group planned to manage the free space. Justin made a Notion page to help list the tasks that have been completed as well as any remaining tasks; a screenshot of the Notion page (with subtasks collapsed) is shown below. From there, team members could comment on which tasks they were currently working on and easily update each other on which tasks had been completed.




Milestone One

formatting the volume:

begin implementing the basic components of the file system

✓ PDF


-  [CSC 415: File System Milestone One](#)
- ▶ PDF Requirements

→ Deliverables

1. Team GitHub
2. PDF

✓ GitHub tasks

- ▶ What Needs to be Submitted
 - ▶ General Expectations for All Projects
 - ▶ Writing a File System
-
- ▶ ReadMe Notes

 3 comments

Issues & Resolutions

Issue & Resolution 1:

One issue that our team faced was a limited overlap in time availability. This made it difficult to plan live meetings where we would all be present. We had some live meetings in class, during the time class would have been, again over break, and before Milestone One.

However, this frequency of meeting would be insufficient to be able to divide up and complete the tasks on time. So to resolve this, we used a Discord server and Notion page to allow group members to pick up tasks, ask questions, respond to questions, and update group members when they had time available. Then, group members could check these chats and tools and see what had been completed and what was still remaining that needed to be done.

Our other issues were minor where people were either able to resolve the issue on their own or it was quickly resolved in the group Discord server.