File System Milestone 1

Summer 2023

Team: Vancouver

Vijayraj Tolnoorkar, 922110069
Phillip Ma, 920556972
Janvi Patel, 917944864
Leo Wu, 917291133

Github Name: Vijayt2001

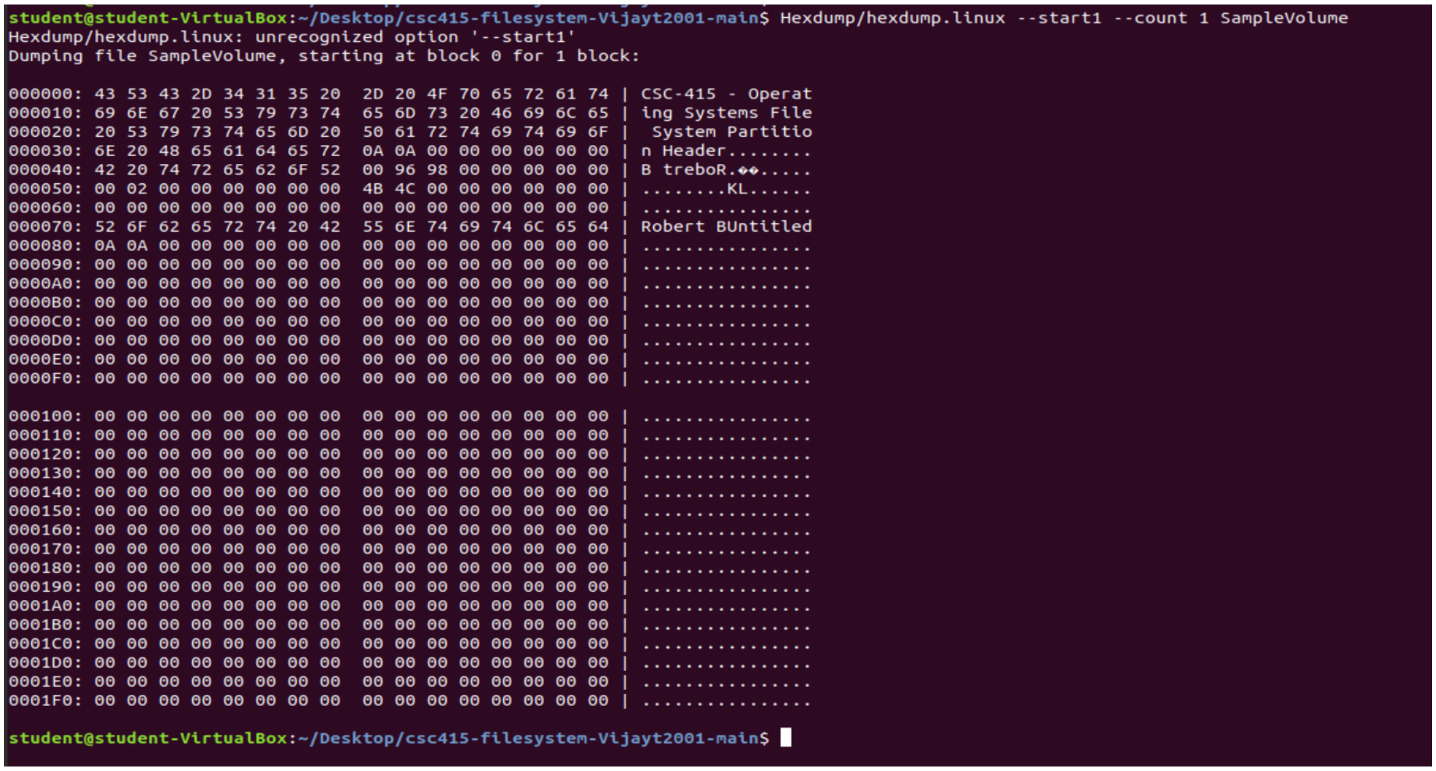
Overview:
This assignment is to implement the first phase of the file system in C formatting the volume by writing the Volume Control Block, free space management, and the root directory.

Approach:
The approach to implementing the VCB is dynamically allocating memory, in order to create a buffer to store the VCB that is read from the file system. Before initializing the values in the VCB structure, a condition must check whether it has already been initialized by determining whether the signature value is the expected value.

The approach to implementing free space structure was to use the bit-map-based approach instead of a grouping, linked list, and counting approach. Simply because it is easy to manipulate and access block allocation status using bitwise operations and it is also efficient.

The approach to implementing the directory entry is defining a structure which contains the necessary metadata values to access and manage the file system. These values are necessary to create and define two directory entries: "." and ".."

1. HEXDUMP



```
student@student-VirtualBox:~/Desktop/csc415-filesystem-Vijayt2001-main$ Hexdump/hexdump.linux --start1 --count 1 SampleVolume
Hexdump/hexdump.linux: unrecognized option '--start1'
Dumping file SampleVolume, starting at block 0 for 1 block:

000000: 43 53 43 2D 34 31 35 20  2D 20 4F 70 65 72 61 74 | CSC-415 - Operat
000010: 69 6E 67 20 53 79 73 74  65 6D 73 20 46 69 6C 65 | ing Systems File
000020: 20 53 79 73 74 65 6D 20  50 61 72 74 69 74 69 6F |  System Partitio
000030: 6E 20 48 65 61 64 65 72  0A 0A 00 00 00 00 00 00 | n Header........
000040: 42 20 74 72 65 62 6F 52  00 96 98 00 00 00 00 00 | B treboR.��.....
000050: 00 02 00 00 00 00 00 00  4B 4C 00 00 00 00 00 00 | ........KL......
000060: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000070: 52 6F 62 65 72 74 20 42  55 6E 74 69 74 6C 65 64 | Robert BUntitled
000080: 0A 0A 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000090: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0000A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0000B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0000C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0000D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0000E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0000F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................

000100: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000110: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000120: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000130: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000140: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000150: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000160: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000170: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000180: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000190: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0001A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0001B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0001C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0001D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0001E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0001F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................

student@student-VirtualBox:~/Desktop/csc415-filesystem-Vijayt2001-main$ █
```

Description of the VCB:

```
typedef struct VCB {
            uint64_t signature; //Determining whether a particular file system exists.
            uint64_t freeSpaceStartBlock; //Block number where free space starts.
} VCB;
```

The first step is defining a structure that is needed to initialize the VCB. The structure should have two variables: **signature**, which uniquely identifies the existence of a particular file system and **freeSpaceStartBlock**, which defines the block number where the free space starts. The next steps involve allocating memory for the VCB, reading the VCB from the disk, and determining whether the file system has already been initialized.

Description of the Free Space Structure:

The free space structure is implemented to keep track of the allocation status of blocks and accurately track the allocated and free blocks in our file system. In the structure, we have implementations of **freeSpaceMap** array, **blockNumber**, **isAllocated**, **setBlockAsAllocated**, and **setBlockAsAloacted.**

freeSpaceMap: This is where each block structure represents a byte in the freeSpaceMap.

setBlockAsAllocated and **setBlockAsAloacted:** These functions are designed to manipulate the allocation status by updating the bits and bytes.

Description of the Directory System:

```
typedef struct directoryInfo
{
   char *filename;
   uint64_t date;
   uint64_t id;
   uint64_t size;
   uint64_t location;
} directoryInfo;
```

The directories are established using the directory system. The root directory is created, and its attributes (root name, size, location, etc.) are initialized. The name field will hold the file's name and extension. Date will keep track of the time and date that file was created. The file size will be stored in the size field. The location field contains the LBA of the file's first block.

Group Member Efforts

Group Members	Role
Vijayraj Tolnoorkar	Worked on the implementation of free space and writeup
Philip Ma	Implemented/Initialized root directory, worked on writeup
Janvi Patel	worked on writeup
Leo Wu	Worked on initialization of file system, worked on pdf writeup

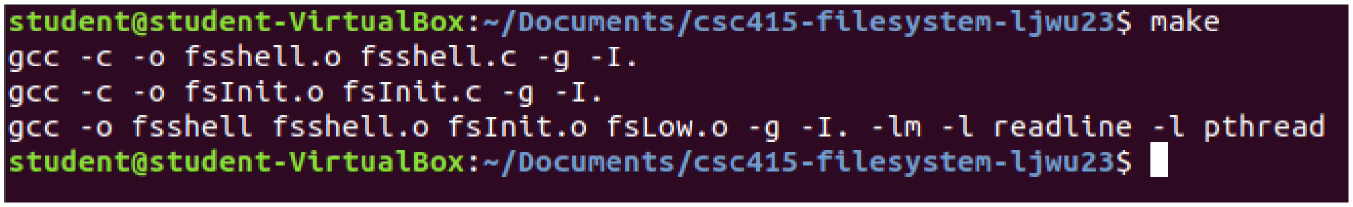
How did your team work together, how often you met, how did you meet, and how did you divide up the tasks?

Our team used Discord for collaboration, allowing us to communicate and discuss our work effectively. After class, we would get together to discuss the project and assign duties to one another. We used Visual Studio Code's live share which made it easier to work on code by allowing us to meet virtually. It made it easy to work on the project into manageable pieces and put everything together. Whenever anyone faced challenges, we readily provided assistance and support to overcome problems. This teamwork helped our team function well together.

What issues you faced and how your team resolved them:

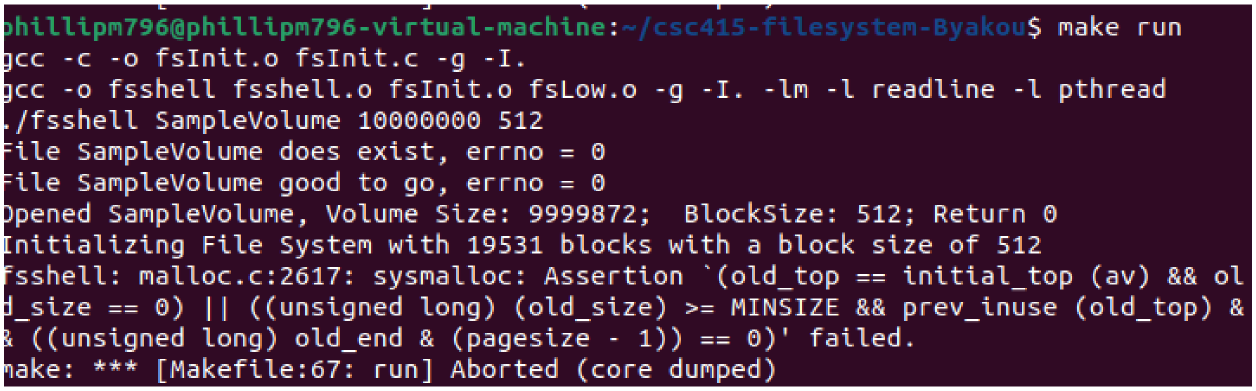
While we were planning on how to implement the freeSpace function, we were not sure what approach we should adapt. Instead of using a bit-map approach, we tried to utilize the linked list method. Finding a method to allocate space for the blocks was also difficult. By calculating the byte index and bit offset, and returns one if the block is allocated and 0 if not. Setting up the directory was quite easy, since initialization only happens once, however, within the function initFileSystem(), Phillip was having a difficult time writing the implementation for the root directory. Phillip decided to check availability of free blocks (with findFreeBlocks()) before allocating.

Screenshot of compilation:



```
student@student-VirtualBox:~/Documents/csc415-filesystem-ljwu23$ 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 fsLow.o -g -I. -lm -l readline -l pthread
student@student-VirtualBox:~/Documents/csc415-filesystem-ljwu23$ 
```

Screenshot(s) of the execution of the program:



```
hillipm796@phillipm796-virtual-machine:~/csc415-filesystem-Byakou$ make run
gcc -c -o fsInit.o fsInit.c -g -I.
gcc -o fsshell fsshell.o fsInit.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
fsshell: malloc.c:2617: sysmalloc: Assertion `(old_top == initial_top (av) && ol
d_size == 0) || ((unsigned long) (old_size) >= MINSIZE && prev_inuse (old_top) &
& ((unsigned long) old_end & (pagesize - 1)) == 0)' failed.
make: *** [Makefile:67: run] Aborted (core dumped)
```