

CSC 415-01: Summer 2023

Github Name: asafi67

Github link: <https://github.com/CSC415-2023-Summer/csc415-filessystem-asafi67.git>

Team: File System Soldiers

Team members: Joe Sand (920525382), Anish Khadka (921952002), Ameen Safi (920689065), Eduardo Hernandez(), Carlos Campos Lozano()

# File System

## Description of File System:

Our group has implemented a basic file system with key file I/O operations. It is designed to manage files on disk efficiently. It supports essential operations for a file system such as file creation, file opening, reading and writing.

To organize files on disk, we utilize blocks of 512 bytes, known as CHUNKS. Each file is represented by one or more extents, which are sequences of contiguous blocks used to store the file's data. Our file system employs the bitmap to manage the allocation of free and used blocks on the disk. Each open file has an associated file control block (FCB), which stores crucial information such as current buffer, file position, parent directory entry.

We have implemented functions to traverse directories and locate file entries within the directory structure. This enables efficient file lookup and management. Our file system ensures that multiple files can be opened simultaneously, and it dynamically expands files to accommodate data growth.

CSC 415-01: Summer 2023

Github Name: asafi67

Github link: <https://github.com/CSC415-2023-Summer/csc415-filesystem-asafi67.git>

Team: File System Soldiers

Team members: Joe Sand (920525382), Anish Khadka (921952002), Ameen Safi (920689065), Eduardo Hernandez(), Carlos Campos Lozano()

## Volume Control Block:

```
C volumeCB.h > ...
1  #ifndef _VCB_H_
2  #define _VCB_H_
3
4
5  typedef struct VolumeControlBlock {
6      //This is the signature of our file system
7      //Checked to see if fs is mounted
8      int magic_number;
9      //Total number of blocks partitioned for the fs
10     int num_blocks;
11     //Size of a single block on disk for this fs
12     int block_size;
13     //Number of remaining free blocks in the partitioned region of the disk
14     int freeBlockCount;
15     //Beginning of bitmap on disk
16     int bitmap_index;
17     //Beginning of root on disk
18     int root_index;
19 } VolumeControlBlock;
20
21 extern VolumeControlBlock* vcb;
22
23 #endif
24
25
```

The VolumeControlBlock (VCB) is a data structure used in a file system to store essential information about the file system and manage its various components. Our particular vcb, is influenced by the choice to manage things like the free space and directories, using a bitmap. The various components of our vcb are all of type int, this way we can also customize negative values to mean different things within the context of the program.

**magic\_number** is a signature or identifier used to verify whether the file system is mounted properly. It is a special value that helps identify the file system type and distinguish it from other data on the storage device. When mounting the file system, this value is checked to ensure that the correct file system is being accessed.

**num\_blocks** field stores the total number of blocks that are partitioned for the file system. A block is the smallest unit of storage used by the file system. The num\_blocks field provides essential information about the size and capacity of the file system.

CSC 415-01: Summer 2023

Github Name: asafi67

Github link: <https://github.com/CSC415-2023-Summer/csc415-filessystem-asafi67.git>

Team: File System Soldiers

Team members: Joe Sand (920525382), Anish Khadka (921952002), Ameen Safi (920689065), Eduardo Hernandez(), Carlos Campos Lozano()

**block\_size** indicates the size of a single block on the disk for this file system. It basically defines the smallest unit at which data is read from or written to the disk. The size of the block affects the overall efficiency and performance of the file system.

**freeBlockCount** field keeps track of the number of remaining free blocks in the partitioned region of the disk. As files are created, modified, or deleted, the file system updates this value to maintain an accurate count of available free blocks.

**bitmap\_index** indicates the position or location on the disk where the bitmap of the file system starts. The bitmap is a data structure used to track the allocation status of each block on the disk. It keeps track of which blocks are in use (allocated to files) and which blocks are free for future allocation.

**root\_index** represents the position or location on the disk where the root directory of the file system starts. The root directory is the top-level directory of the file system and serves as the starting point for navigating the entire directory tree.

We declared the VolumeControl block pointer variable \*vcb, which has the keyword “extern” attached to it. This is because we want to declare this as a global variable, and the variable can be declared and accessed easily by other scopes and files.

## HEXDUMP VCB :

```
student@student-VirtualBox:~/Desktop/CSC415/csc415-filessystem-asafi67$ Hexdump/hexdump.linux --start 0 --count 5 SampleVolume
Dumping file SampleVolume, starting at block 0 for 5 blocks:
000000: 43 53 43 20 34 31 35 20 20 20 4F 70 65 72 61 74 | CSC-415 - Operat
000010: 69 6E 67 20 53 79 73 74 65 00 73 20 46 69 6C 65 | ing Systems File
000020: 20 53 79 73 74 65 60 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 | .....
000200: 19 55 06 00 4B 4C 00 00 00 02 00 00 45 4C 00 00 | .U..KL.....EL..
000210: 01 00 00 00 FF FF FF FF 00 00 00 00 00 00 00 00 | ...
000220: 00 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 | .....
```

CSC 415-01: Summer 2023

Github Name: asafi67

Github link: <https://github.com/CSC415-2023-Summer/csc415-filesystem-asafi67.git>

Team: File System Soldiers

Team members: Joe Sand (920525382), Anish Khadka (921952002), Ameen Safi (920689065), Eduardo Hernandez(), Carlos Campos Lozano()

## Free Space Management System:

We chose to manage our free space with a bitmap unsigned character array. A bitmap is an array of bits where each bit represents a unit of free space. A bit has the value of 0 (resource is free) or 1 (resource is allocated). Each unsigned char is 1 byte or 8 bits. Our bitmap allowed us to track free and allocated blocks in our file system. Several functions were created in bitmap.c to provide functionalities such as initialize, load, allocate, and deallocate blocks. Additional helper functions were also added to perform binary manipulations and checks. Many of these functions utilize bitwise logic.

```
home > student > csc415-filesystem-asafi67 > C bitmap.h
1  #ifndef BITMAP_H
2  #define BITMAP_H
3
4  #include <stdlib.h>
5  #include <unistd.h>
6  #include <sys/types.h>
7  #include <stdio.h>
8  #include <string.h>
9  #include "volumeCB.h"
10 #include "fsLow.h"
11
12 //configuration constants, manually modify
13 #define BITMAP_POSITION 1
14 #define BITS_PER_BYTE 8
15
16 extern unsigned char* bitmap; //define a bitmap of char strings
17 extern int blockSize; // declare variable for blocksize
18
19 // Function prototypes
20 int bitmapInit(int numBlocks, int blockSize);
21 char *byteToBinaryString(unsigned char inputByte);
22 unsigned char generateBitMask(int position);
23 int checkBitStatus(int bitPosition, char charToTest);
24 int findFreeBitIndex(int blocksNeeded, unsigned char* bitmapArray, int bitmapSize);
25 int allocateBlocks(int blocksRequired, unsigned char* bitmapArray, int bitmapSize);
26 void releaseBlocks(int blocksToRelease, int initialPosition);
27 int loadBitmap(void);
28
29 #endif // BITMAP_H
```

### Main functions:

#### **int bitmapInit(int numBlocks, int blockSize)**

This function initializes a bitmap with the specified number of blocks and block size. It calculates the number of bytes and blocks needed for the bitmap, then allocates memory for it using calloc, setting all elements to free (0). After checking for memory allocation errors, it allocates blocks

CSC 415-01: Summer 2023

Github Name: asafi67

Github link: <https://github.com/CSC415-2023-Summer/csc415-filesystem-asafi67.git>

Team: File System Soldiers

Team members: Joe Sand (920525382), Anish Khadka (921952002), Ameen Safi (920689065), Eduardo Hernandez(), Carlos Campos Lozano()

for the VCB and the bitmap itself. The allocated bitmap is written to the disk at the specified position and the function prints the size of the bitmap in blocks and returns the position where the bitmap was written.

### **int loadBitmap()**

This function is responsible for loading the bitmap into memory if it does not already exist. It first checks if the bitmap is in memory and if it is not then it allocates space for it and reads it from disk. Return value of 1 on success and -1 on failure.

### **int allocateBlocks(int blocksRequired, unsigned char\* bitmapArray, int bitmapSize)**

This function allocates a specific number of contiguous blocks in the bitmap. Before allocating the function, we make sure to find the starting index of free contiguous blocks in the bitmap. Once the required amount of free contiguous blocks are found the blocks are marked as allocated (1) and the number of blocks are updated. The updated VCB and bitmap are written to disk and the function returns the starting index of the allocated blocks.

### **void releaseBlocks(int blocksToRelease, int initialPosition)**

This function is responsible for deallocating a specific number of contiguous blocks in the bitmap. The function iterates through the blocks to release and clears each bit by setting it to free (0).

### Helper functions included:

### **char \*byteToBinaryString ( unsigned char inputByte)**

This function is responsible for converting the unsigned byte into its binary string representation. It initializes a static array to hold the binary representation and then loops through the bits of the byte, starting from the most significant bit. The result is the binary string representation of the input byte. This function provides a way to manipulate the individual bits and interact with the bitmap.

### **unsigned char generateBitMask(int position)**

This function is designed to create a bitmask for a given position within a byte. By starting with a mask of '00000001', it shifts the '1' to the specified position within the byte and returns the newly generated bitmask. This is basically used to manipulate specific bits within the byte. Returns the new bitmask.

### **int checkBitStatus(int bitPosition, char charToTest)**

This function is implemented to help check specific bits in the data. It creates a mask for the specified bit position and applies the mask to the character. This helps us determine where a particular block in the bitmap is free or allocated. This returns a 1 if the bit is set, 0 if not.

CSC 415-01: Summer 2023

Github Name: asafi67

Github link: <https://github.com/CSC415-2023-Summer/csc415-filesystem-asafi67.git>

Team: File System Soldiers

Team members: Joe Sand (920525382), Anish Khadka (921952002), Ameen Safi (920689065), Eduardo Hernandez(), Carlos Campos Lozano()

**int findFreeBitIndex(int blocksNeeded, unsigned char\* bitmapArray, int bitmapSize)**

This function is used to find a contiguous block of free bits within the bitmap. It does this by iterating through all the bits in the bitmap to find the first free bit and then checks the required contiguous bits to ensure they are all free up to the provided length. This returns the index of the first free bit or an error if no suitable block is found.

CSC 415-01: Summer 2023

Github Name: asafi67

Github link: <https://github.com/CSC415-2023-Summer/csc415-filesystem-asafi67.git>

Team: File System Soldiers

Team members: Joe Sand (920525382), Anish Khadka (921952002), Ameen Safi (920689065), Eduardo Hernandez(), Carlos Campos Lozano()

## Free Space Hexdump

```
student@
File Edit View Search Terminal Help
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 | .....
000330: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000340: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000350: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000360: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000370: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000380: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000390: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0003A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0003B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0003C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0003D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0003E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0003F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

000400: FC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000410: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000420: 00 00 00 00 00 00 00 00 00 B1 06 02 00 00 00 00 | .....
000430: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000440: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000450: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000460: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000470: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000480: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000490: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0004A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0004B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0004C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0004D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0004E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0004F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....

000500: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000510: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000520: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000530: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000540: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000550: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000560: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000570: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000580: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000590: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0005A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0005B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
```

CSC 415-01: Summer 2023

Github Name: asafi67

Github link: <https://github.com/CSC415-2023-Summer/csc415-filesystem-asafi67.git>

Team: File System Soldiers

Team members: Joe Sand (920525382), Anish Khadka (921952002), Ameen Safi (920689065), Eduardo Hernandez(), Carlos Campos Lozano()

## Extent struct

```
#ifndef EXTENT_H
#define EXTENT_H
#include "rootDir.h"

typedef struct {

    //the location of the first block of the extent on disk
    int block_number;
    //number of contiguous blocks in an extent
    int contiguous_blocks;
}extent;

#endif
```



CSC 415-01: Summer 2023

Github Name: asafi67

Github link: <https://github.com/CSC415-2023-Summer/csc415-filesystem-asafi67.git>

Team: File System Soldiers

Team members: Joe Sand (920525382), Anish Khadka (921952002), Ameen Safi (920689065), Eduardo Hernandez(), Carlos Campos Lozano()

## Directory Entry:

```
home > student > csc415-filesystem-asafi67 > C rootDir.h
1  #ifndef ROOTDIR_H_
2  #define ROOTDIR_H_
3  #define EXTENT_COUNT 8
4  #define BUFFER_SIZE 50
5  #include <time.h>
6
7  #include "extent.h"
8
9
10 //definition of DE (directory entry) structure
11 typedef struct DE{
12
13     char name[20];           //array to hold the entry's name
14     unsigned long int size;   //holds the entry's size
15     int loc;                  //Int representing the entry's location
16     time_t last_accessed;     //timestamp indicating the last time the entry accessed
17     time_t created_at;        //timestamp indicating when the entry was created
18     time_t modified_at;       //timestamp indicating the last time was modified
19     char file_type;           //character representing the type of file
20     extent extents[EXTENT_COUNT]; //an array of extents, max of 8
21     int isDirectory;          //flag to identify whether the entry is a directory or file
22 } DE;
23
24 //function declaration for initializing a directory with a given block size
25 int initDir(DE* parent, int blockSize);
26
27 #endif
```

The directory entry (DE) structure is used to manage the files and directories within our file system. This structure holds the key attributes of a directory entry as well as some metadata.

1. Name - Character array designed to hold the entry's name.
2. Size - An unsigned long integer holding the entry's size.
3. location - An integer representing the specific location of the entry within the file system.
4. Last Accessed - A time value indicating the most recent time the entry was accessed.
5. Created at - A time value that records when the entry was initially created
6. Modified at - A time value capturing the last moment the entry was modified.
7. File type - A character variable representing the type of file
8. Extents - An array of extents, each extent is a continuous block of storage on disk. This is utilized for space management and read/write performance.
9. isDirectory - An int value 'flag' that differentiates between a directory and a file.

CSC 415-01: Summer 2023

Github Name: asafi67

Github link: <https://github.com/CSC415-2023-Summer/csc415-filesystem-asafi67.git>

Team: File System Soldiers

Team members: Joe Sand (920525382), Anish Khadka (921952002), Ameen Safi (920689065), Eduardo Hernandez(), Carlos Campos Lozano()

Along with the structure are functions that help manage the directories.

**int fs\_mkdir(const char \*pathname, mode\_t mode){**

This function is responsible for creating a new directory in a file system based on the given pathname and mode. First it identifies the parent directory and the new directory's name by parsing the pathname. The function also checks if the new directory's name conflicts with the parent directory and if there is enough space in the parent directory. Once all checks pass it reserves blocks on disk and initializes them with the appropriate properties and commits the updated info to disk. This function returns a 1 if successful and -1 if there was an error.

**int fs\_isDir(char \*pathname){**

Function checks whether a given path corresponds to a directory within a file system. It takes a pathname as input and uses the parsePath function to analyze and validate the path. Once the path is validated the function returns a 1 if the path corresponds to a directory or a 0 if it corresponds to a file. This function allows other parts of our code to determine whether a given path points to a directory or a file.

**fdDir \*fs\_opendir(const char \*pathname){**

This function is designed to open a directory at a given path within a file system. It begins similarly to fs\_isDir and parses the provided path using parsePath to validate the path. After successfully parsing the memory is allocated for a file stat structure and the fs\_stat function is called to get information about the specified path. The function then allocates memory for a directory entry 'fdDir' and initializes properties of a directory entry. The function calculates the number of blocks required to read the directory and reads them into memory using LBRead. If all these steps are successful the function will return a pointer to the directory entry. This pointer then allows further operations to be performed on the opened directory.

**struct fs\_direntinfo \*fs\_readdir(fdDir \*dirp){**

This function returns the filled fs\_direntinfo structure containing information about the next valid directory entry. This function reads the next directory entry from the provided directory pointer. After checking if the directory pointer is valid the function allocates memory for a fs\_direntinfo structure to hold the information about the directory entry. It sets the recorded length and determines the file type based on where the entry represents a directory or file. The current directory is copied to the structure and the function increments the directory entry position. It iterates over the directory entries until it reaches the end of the buffer.

**int fs\_closedir(fdDir \*dirp){**

CSC 415-01: Summer 2023

Github Name: asafi67

Github link: <https://github.com/CSC415-2023-Summer/csc415-filesystem-asafi67.git>

Team: File System Soldiers

Team members: Joe Sand (920525382), Anish Khadka (921952002), Ameen Safi (920689065), Eduardo Hernandez(), Carlos Campos Lozano()

This function is used to close an opened directory provided by the pointer. The purpose of this function is for memory management and ensuring the resources are released when they are no longer needed. This function will return 1 signaling the memory was freed.

**int fs\_rmdir(const char \*pathname){**

This function handles the removal of a directory or file by ensuring that the directory is empty, releasing the associated blocks and updating the parent directory accordingly.

**DE \*loadDir(DE toBeLoaded){**

This function loads a directory from disk given its directory entry (toBeLoaded). It starts by calculating the number of blocks required to read the entire directory. Memory is then allocated for the directory array. The directory entries are read from the disk in the allocated memory using LBaread. If any of these steps fail an error message is printed and NULL is returned. If successful a pointer to the loaded directory array is returned.

## File Control Block:

```
typedef struct b_fcb
{
    /** TODO add all the information you need in the file control block */
    char *buf; // holds the open file buffer
    int index; // holds the current position in the buffer
    int buflen; // holds how many valid bytes are in the buffer

    int current_block; // holds the current block #
    int num_blocks; // holds # of blocks a file habits
    int bytes_read; // holds # of bytes which has been read
    int current_extent; // holds int specifying the extent we are on
    int file_offset; // byte we are on in file. Can be altered in b_seek
    int permissions; // depends on b_open. notes type of access
    int parent_location; // location of the parent array for file
    DE *file_entry; // a file's entry in the directory
} b_fcb;
```

CSC 415-01: Summer 2023

Github Name: asafi67

Github link: <https://github.com/CSC415-2023-Summer/csc415-filesystem-asafi67.git>

Team: File System Soldiers

Team members: Joe Sand (920525382), Anish Khadka (921952002), Ameen Safi (920689065), Eduardo Hernandez(), Carlos Campos Lozano()

Issues we had:

## Detail of how your driver program works:

Our driver program allows us to interact with files and folders on our computer. Inside the fsshell.c file our driver program allows us to interact with the computer via a terminal window. Inside the terminal window is our shell that knows commands that we have written. The shell understands two types of commands for files and folders. The file b\_io.c helps with files and mfs.c helps with folders. When running this program it sets everything up and remembers your terminal command history. The shell waits for you to type commands, performs the command, then waits for the next command. Only typing 'exit' will cause the shell to stop running. Ultimately, our driver program is a tool that lets you control files and folders by typing commands using a terminal window.

Screenshots showing each of the commands listed:

**Compilation:**

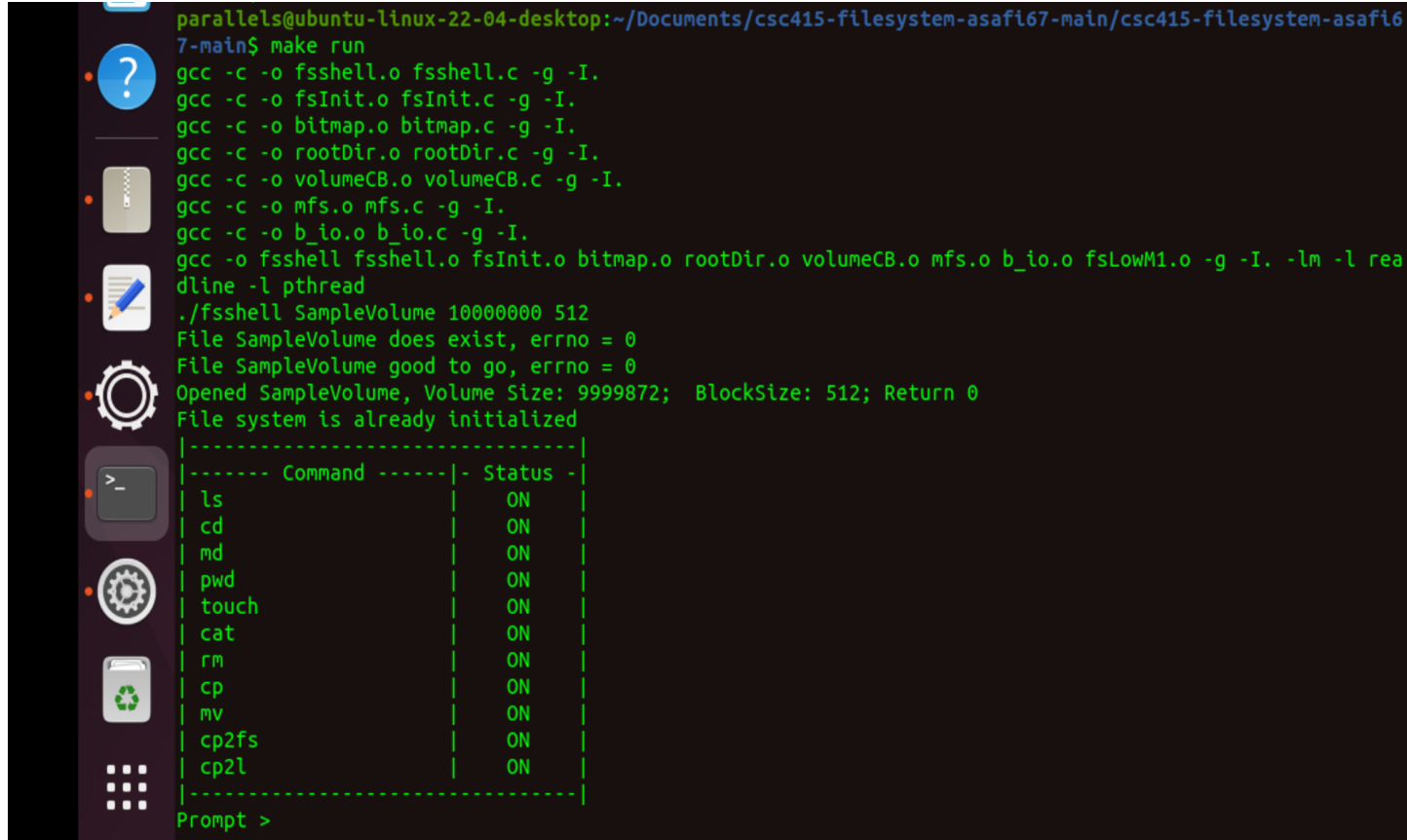
CSC 415-01: Summer 2023

Github Name: asafi67

Github link: <https://github.com/CSC415-2023-Summer/csc415-filesystem-asafi67.git>

Team: File System Soldiers

Team members: Joe Sand (920525382), Anish Khadka (921952002), Ameen Safi (920689065), Eduardo Hernandez(), Carlos Campos Lozano()

A terminal window on a Linux desktop. The prompt is 'parallels@ubuntu-linux-22-04-desktop:~/Documents/csc415-filesystem-asafi67-main/csc415-filesystem-asafi67-main\$'. The user enters 'make run'. The terminal shows the compilation of several C files into object files and then linking them into 'fsshell.o'. It then runs './fsshell SampleVolume 10000000 512'. The output shows that the file 'SampleVolume' exists, is good to go, and is opened with a size of 9999872 and block size of 512. It then displays a table of commands and their status.

```
parallels@ubuntu-linux-22-04-desktop:~/Documents/csc415-filesystem-asafi67-main/csc415-filesystem-asafi67-main$ make run
gcc -c -o fsshell.o fsshell.c -g -I.
gcc -c -o fsInit.o fsInit.c -g -I.
gcc -c -o bitmap.o bitmap.c -g -I.
gcc -c -o rootDir.o rootDir.c -g -I.
gcc -c -o volumeCB.o volumeCB.c -g -I.
gcc -c -o mfs.o mfs.c -g -I.
gcc -c -o b_io.o b_io.c -g -I.
gcc -o fsshell fsshell.o fsInit.o bitmap.o rootDir.o volumeCB.o mfs.o b_io.o fsLowM1.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
File system is already initialized
----- Command -----| Status |
| ls                      | ON     |
| cd                      | ON     |
| md                      | ON     |
| pwd                    | ON     |
| touch                  | ON     |
| cat                    | ON     |
| rm                     | ON     |
| cp                     | ON     |
| mv                     | ON     |
| cp2fs                  | ON     |
| cp2l                   | ON     |
-----
Prompt >
```

“md” and “ls” command:

CSC 415-01: Summer 2023

Github Name: asafi67

Github link: <https://github.com/CSC415-2023-Summer/csc415-filesystem-asafi67.git>

Team: File System Soldiers

Team members: Joe Sand (920525382), Anish Khadka (921952002), Ameen Safi (920689065), Eduardo Hernandez(), Carlos Campos Lozano()

-----	
Command	Status
ls	ON
cd	ON
md	ON
pwd	ON
touch	ON
cat	ON
rm	ON
cp	ON
mv	ON
cp2fs	ON
cp2l	ON
-----	

Prompt > ls

U

test

Prompt >

Prompt >

Prompt > md test2

Prompt > md test3

Prompt > md test4

Prompt > ls

U

test

test2

test3

test4

Prompt >

CSC 415-01: Summer 2023

Github Name: asafi67

Github link: <https://github.com/CSC415-2023-Summer/csc415-filesystem-asafi67.git>

Team: File System Soldiers

Team members: Joe Sand (920525382), Anish Khadka (921952002), Ameen Safi (920689065), Eduardo Hernandez(), Carlos Campos Lozano()

**“pwd” & “cd” command:**

```
|----- Command -----| - Status - |
| ls                      |      ON  |
| cd                      |      ON  |
| md                      |      ON  |
| pwd                    |      ON  |
| touch                  |      ON  |
| cat                    |      ON  |
| rm                     |      ON  |
| cp                     |      ON  |
| mv                     |      ON  |
| cp2fs                  |      ON  |
| cp2l                   |      ON  |
|-----|-----|
Prompt >
Prompt >
Prompt >
Prompt > ls

U
test
test2
test3
test4
Prompt >
Prompt >
Prompt > pwd
/
Prompt >
Prompt >
Prompt > cd test2
Prompt >
Prompt >
Prompt > pwd
/test2
Prompt >
Prompt >
Prompt >
Prompt >
```

CSC 415-01: Summer 2023

Github Name: asafi67

Github link: <https://github.com/CSC415-2023-Summer/csc415-filesystem-asafi67.git>

Team: File System Soldiers

Team members: Joe Sand (920525382), Anish Khadka (921952002), Ameen Safi (920689065), Eduardo Hernandez(), Carlos Campos Lozano()

“touch”:

```
|----- Command -----| Status |
| ls                      |      ON |
| cd                      |      ON |
| md                      |      ON |
| pwd                    |      ON |
| touch                  |      ON |
| cat                    |      ON |
| rm                     |      ON |
| cp                     |      ON |
| mv                     |      ON |
| cp2fs                  |      ON |
| cp2l                   |      ON |
|-----|-----|
Prompt >
Prompt >
Prompt > touch abc.c
```

When we “ls”, we see the file abc.c in the directory.



CSC 415-01: Summer 2023

Github Name: asafi67

Github link: <https://github.com/CSC415-2023-Summer/csc415-filesystem-asafi67.git>

Team: File System Soldiers

Team members: Joe Sand (920525382), Anish Khadka (921952002), Ameen Safi (920689065), Eduardo Hernandez(), Carlos Campos Lozano()

```
|-----|
|----- Command -----| - Status - |
| ls                |      ON      |
| cd                |      ON      |
| md                |      ON      |
| pwd              |      ON      |
| touch            |      ON      |
| cat              |      ON      |
| rm               |      ON      |
| cp               |      ON      |
| mv               |      ON      |
| cp2fs            |      ON      |
| cp2l             |      ON      |
|-----|
Prompt >
Prompt >
Prompt > ls

U
test
test2
test3
test4
abc.c
Prompt >
Prompt >
Prompt >
```

“rm” command:

CSC 415-01: Summer 2023

Github Name: asafi67

Github link: <https://github.com/CSC415-2023-Summer/csc415-filesystem-asafi67.git>

Team: File System Soldiers

Team members: Joe Sand (920525382), Anish Khadka (921952002), Ameen Safi (920689065), Eduardo Hernandez(), Carlos Campos Lozano()

-----	
Command	Status
ls	ON
cd	ON
md	ON
pwd	ON
touch	ON
cat	ON
rm	ON
cp	ON
mv	ON
cp2fs	ON
cp2l	ON
-----	

Prompt >

Prompt >

Prompt > md delete\_dir

Prompt >

Prompt > ls

Terminal

delete\_dir

Prompt >

Prompt >

Prompt > rm delete\_dir

Prompt >

Prompt > ls

U

Prompt >

Prompt > █

CSC 415-01: Summer 2023

Github Name: asafi67

Github link: <https://github.com/CSC415-2023-Summer/csc415-filesystem-asafi67.git>

Team: File System Soldiers

Team members: Joe Sand (920525382), Anish Khadka (921952002), Ameen Safi (920689065), Eduardo Hernandez(), Carlos Campos Lozano()

**“cp” not fully functional, segmentation fault**

```
Prompt > ls
U
Prompt >
Prompt > md copy_dir
Prompt > ls
U
copy_dir
Prompt >
Prompt > cp copy_dir copy_dir2
Unable to open directory with that command
make: *** [Makefile:67: run] Segmentation fault (core dumped)
parallels@ubuntu-linux-22-04-desktop:~/Documents/csc415-filesystem-asafi67-main/csc415-filesystem-asafi67-main$ make run
```

However, we still see the directory being copied when we rerun the program.

```
|----- Command -----| - Status - |
| ls                      |      ON      |
| cd                      |      ON      |
| md                      |      ON      |
| pwd                    |      ON      |
| touch                  |      ON      |
| cat                    |      ON      |
| rm                     |      ON      |
| cp                     |      ON      |
| mv                     |      ON      |
| cp2fs                  |      ON      |
| cp2l                   |      ON      |
|-----|
Prompt > ls
U
copy_dir
copy_dir2
Prompt >
Prompt >
Prompt >
Prompt >
Prompt >
```

CSC 415-01: Summer 2023

Github Name: asafi67

Github link: <https://github.com/CSC415-2023-Summer/csc415-filesystem-asafi67.git>

Team: File System Soldiers

Team members: Joe Sand (920525382), Anish Khadka (921952002), Ameen Safi (920689065), Eduardo Hernandez(), Carlos Campos Lozano()

HEXDUMP after directories such as home and desktop as well as files like txt.txt and txt1.txt were created:

CSC 415-01: Summer 2023

Github Name: asafi67

Github link: <https://github.com/CSC415-2023-Summer/csc415-filesystem-asafi67.git>

Team: File System Soldiers

Team members: Joe Sand (920525382), Anish Khadka (921952002), Ameen Safi (920689065), Eduardo Hernandez(), Carlos Campos Lozano()

```
rm fsshell.o fsInit.o bitmap.o rootDir.o volumeCB.o mfs.o b_io.o fsshell
student@student-VirtualBox:~/Desktop/CSC415/csc415-filesystem-asafi67$ Hexdump/hexdump.linux --start 0 --count 5 SampleVolume
Dumping file SampleVolume, starting at block 0 for 5 blocks:

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 | .....

000200: 19 55 06 00 4B 4C 00 00 00 02 00 00 45 4C 00 00 | .U..KL.....EL..
000210: 01 00 00 00 FF FF FF FF 00 00 00 00 00 00 00 00 | ....♦♦♦♦.....
000220: 00 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 | .....
```

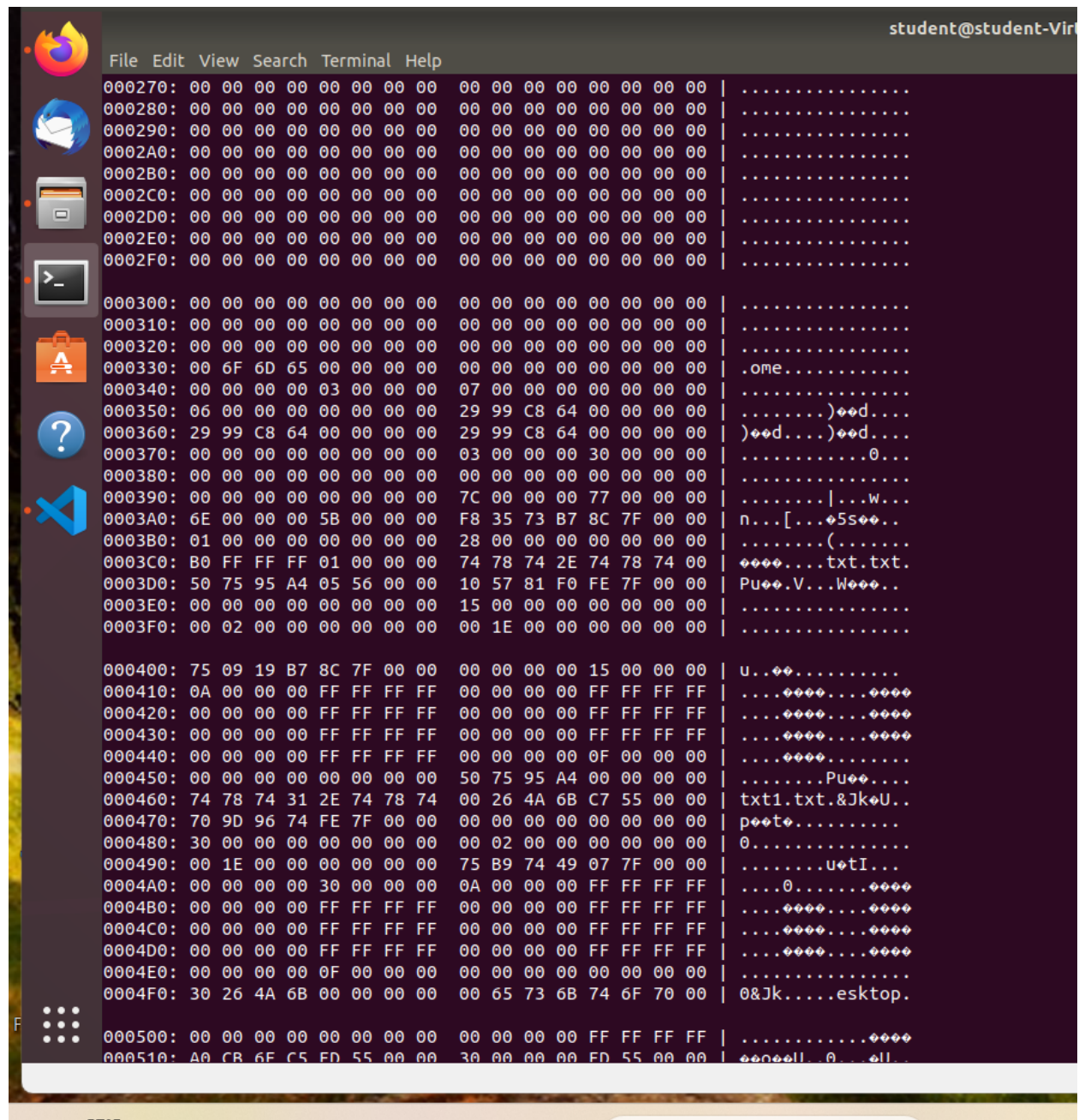
CSC 415-01: Summer 2023

Github Name: asafi67

Github link: <https://github.com/CSC415-2023-Summer/csc415-filesystem-asafi67.git>

Team: File System Soldiers

Team members: Joe Sand (920525382), Anish Khadka (921952002), Ameen Safi (920689065), Eduardo Hernandez(), Carlos Campos Lozano()



```
student@student-Virt
File Edit View Search Terminal Help
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 | .....
000330: 00 6F 6D 65 00 00 00 00 00 00 00 00 00 00 00 00 | .ome.....
000340: 00 00 00 00 03 00 00 00 07 00 00 00 00 00 00 00 | .....
000350: 06 00 00 00 00 00 00 00 29 99 C8 64 00 00 00 00 | .....)ed...
000360: 29 99 C8 64 00 00 00 00 29 99 C8 64 00 00 00 00 | )ed....)ed...
000370: 00 00 00 00 00 00 00 00 03 00 00 00 30 00 00 00 | .....0...
000380: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000390: 00 00 00 00 00 00 00 00 7C 00 00 00 77 00 00 00 | .....|.w...
0003A0: 6E 00 00 00 5B 00 00 00 F8 35 73 B7 8C 7F 00 00 | n...[...Ss...
0003B0: 01 00 00 00 00 00 00 00 28 00 00 00 00 00 00 00 | .....(.....
0003C0: B0 FF FF FF 01 00 00 00 74 78 74 2E 74 78 74 00 | ....txt.txt.
0003D0: 50 75 95 A4 05 56 00 00 10 57 81 F0 FE 7F 00 00 | Pu...V...W...
0003E0: 00 00 00 00 00 00 00 00 15 00 00 00 00 00 00 00 | .....
0003F0: 00 02 00 00 00 00 00 00 00 1E 00 00 00 00 00 00 | .....
000400: 75 09 19 B7 8C 7F 00 00 00 00 00 00 15 00 00 00 | U...
000410: 0A 00 00 00 FF FF FF FF 00 00 00 00 FF FF FF FF | ....
000420: 00 00 00 00 FF FF FF FF 00 00 00 00 FF FF FF FF | ....
000430: 00 00 00 00 FF FF FF FF 00 00 00 00 FF FF FF FF | ....
000440: 00 00 00 00 FF FF FF FF 00 00 00 00 0F 00 00 00 | ....
000450: 00 00 00 00 00 00 00 00 50 75 95 A4 00 00 00 00 | .....Pu...
000460: 74 78 74 31 2E 74 78 74 00 26 4A 6B C7 55 00 00 | txt1.txt.&JkU..
000470: 70 9D 96 74 FE 7F 00 00 00 00 00 00 00 00 00 00 | p...
000480: 30 00 00 00 00 00 00 00 00 02 00 00 00 00 00 00 | 0.....
000490: 00 1E 00 00 00 00 00 00 75 B9 74 49 07 7F 00 00 | .....u...
0004A0: 00 00 00 00 30 00 00 00 0A 00 00 00 FF FF FF FF | ....0.....
0004B0: 00 00 00 00 FF FF FF FF 00 00 00 00 FF FF FF FF | ....
0004C0: 00 00 00 00 FF FF FF FF 00 00 00 00 FF FF FF FF | ....
0004D0: 00 00 00 00 FF FF FF FF 00 00 00 00 FF FF FF FF | ....
0004E0: 00 00 00 00 0F 00 00 00 00 00 00 00 00 00 00 00 | .....
0004F0: 30 26 4A 6B 00 00 00 00 00 65 73 6B 74 6F 70 00 | 0&Jk....esktop.
000500: 00 00 00 00 00 00 00 00 00 00 00 00 FF FF FF FF | .....
000510: A0 CB 6F C5 FD 55 00 00 30 00 00 00 FD 55 00 00 | ...ell...ell...
```



CSC 415-01: Summer 2023

Github Name: asafi67

Github link: <https://github.com/CSC415-2023-Summer/csc415-filessystem-asafi67.git>

Team: File System Soldiers

Team members: Joe Sand (920525382), Anish Khadka (921952002), Ameen Safi (920689065), Eduardo Hernandez(), Carlos Campos Lozano()

```
File Edit View Search Terminal Help
0004D0: 00 00 00 00 FF FF FF FF 00 00 00 00 FF FF FF FF | .....
0004E0: 00 00 00 00 0F 00 00 00 00 00 00 00 00 00 00 00 | .....
0004F0: 30 26 4A 6B 00 00 00 00 00 65 73 6B 74 6F 70 00 | 0&Jk.....esktop.
000500: 00 00 00 00 00 00 00 00 00 00 00 00 FF FF FF FF | .....
000510: A0 CB 6F C5 ED 55 00 00 30 00 00 00 ED 55 00 00 | .....U..0...U..
000520: EA A4 C8 64 00 00 00 00 EA A4 C8 64 00 00 00 00 | .....d.....d....
000530: EA A4 C8 64 00 00 00 00 80 D9 61 F7 C6 7F 00 00 | .....d.....a....
000540: 00 6B 9C 5A 78 5D D3 54 A0 CB 6F C5 ED 55 00 00 | .kZx]T.....U..
000550: C0 CB 6F C5 ED 55 00 00 40 B6 6E C5 ED 55 00 00 | .....U..@n...U..
000560: 39 17 86 F7 C6 7F 00 00 E0 32 85 F7 C6 7F 00 00 | 9.....2....
000570: F8 B5 85 F7 C6 7F 00 00 06 00 00 00 00 00 00 00 | .....
000580: 30 00 00 00 00 00 00 00 10 C2 7D C4 01 00 00 00 | 0.....}+....
000590: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0005A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0005B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0005C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0005D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0005E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0005F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000600: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000610: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000620: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000630: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000640: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000650: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000660: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000670: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000680: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000690: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0006A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0006B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0006C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0006D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0006E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
0006F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000700: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000710: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000720: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000730: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000740: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000750: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000760: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
000770: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
57°F
```

The first screenshot of the beginning of the hexdump demonstrates the VCB initialization, as well as the free space initialization. The second two demonstrate the directories being populated after the times we used the command “md” to make new directories. As well as the files within the directories can also be seen in the hexdump. We can see remnants of the names of

CSC 415-01: Summer 2023

Github Name: asafi67

Github link: <https://github.com/CSC415-2023-Summer/csc415-filesystem-asafi67.git>

Team: File System Soldiers

Team members: Joe Sand (920525382), Anish Khadka (921952002), Ameen Safi (920689065), Eduardo Hernandez(), Carlos Campos Lozano()

directories such as home and desktop, as well as files such as txt.txt and txt1.txt, within the ascii equivalents on the right side of the hex dump.