MileStone 1 (github: jeawanjang)

Group name: GeriatricCats

Name: Jeawan Jang, Paige Hodgkinson, Randale Reyes, Carlos Campos Lozano

ID: 923070860, 922282852, 921008696, 920768261

Github: jeawanjang, phodgkinson1, RandaleReyes, ccamposlozano

Description:

- Volume Control Block (VCB structure)

Screenshot (regarding volume control block structure):

```
// struct VCB
typedef struct VCB
{
        long signature;         // Signature determines if formatted
        int totalNumBlocks;           //total blocks in entire volume including VCB
        int sizeOfBlock;        // bytes per block (512)
        int startFreeSpaceManagement;
        int startDirectory;

} VCB;
```

First of all, the Volume Control Block (VCB) is a fundamental structure in a file system that allows managing and maintaining crucial information about the entire file system. The VCB is located at the initial location of the volume so that serves as a reference indicator for various file system operations.

Variables Usage:

- Signature: is a long integer value which determines whether the file system has been initialized or not. In other words, it acts as an unique identifier.

- TotalNumBlocks: indicates the total number of blocks in the entire volume, including VCB itself. It defines the maximum capacity of the family system in terms of blocks, which can be derived through equation, volume / blockSize

- SizeOfBlock: specifies the size of each block in bytes. Particularly in this code, it is set to 512.
- StartFreeSpaceManagement: holds the block number where the free space management data structure commences.

- StartDirectory: manifests the block number where the root directory of the file system begins.

- Free space

Screenshot (regarding free space structure → Our choice: extent):

```
// struct extent
// Includes
// initFreeSpace() loadFreeSpace() allocateBlocks() releaseBlocks()
typedef struct extent
{
        int start;
        int count;
} extent, *pextent;
```

The extent structure manages the availability of contiguous free space in the system by using two attributes, start and count. This structure streamlines the allocation of free blocks, reducing fragmentation via allowing contiguous space for data storage.

Functions Usage:

```
// InitFreeSpace is called when you initialize the volume
// returns the block number where the freespace map starts
int initFreeSpace (int blockCount, int bytesPerBlock)
        {

        // Calculate the size of the bitmap in bytes
        int bitmapSize = blockCount / 8;

        if (bitmapSize % BLOCK_SIZE > 0)
                {
                bitmapSize = BLOCK_SIZE * (bitmapSize / BLOCK_SIZE + 1);
                }
        bytesInBitmap = bitmapSize;
        int freeSpaceBlockCounts = bitmapSize / BLOCK_SIZE;

        // Allocate a memory of bitmap based on bitmapSize
        unsigned char * bitmap = malloc(bitmapSize);
        bitmapGlobal = bitmap;

        //initialize every bit on the bitmap to 0
        memset(bitmap, 0, bitmapSize);

        //marked the first 6 bits to used, block 0 for VCB + blocks 1-5 for the bitmap
        for (int i = 0; i < freeSpaceBlockCounts + 1; i++)
                {
                int byteIndex = i / 8; //Find the index of the byte (not user it should be *)
                int bitIndex =  i % 8; //Find index of bit inside the byte
                uint64_t mask= 1;
                bitmap[byteIndex] |= (mask << bitIndex); //set the bit to one
                }

        //write to the disk
        LBAwrite(bitmap, freeSpaceBlockCounts, 1);

        //return the block where the free space starts
        return 1;
}
```

- The initFreeSpace function's primary role is to set up and initialize the Free Space Map (FSM), a data structure responsible for tracking the availability of blocks in the file system. By calculating the size of the bitmap in bytes, with each bit in the bitmap representing the status of a block. To ensure the block size, bitmapSize is rounded up to the nearest multiple of the block size. It allocates memory for the bitmap and initializes all bits to free blocks. Besides, as noting certain blocks 0 for VCB and blocks 1 to 5 for FSM, it confirms that the initial reserved blocks are correctly identified.

- Directory

Screenshot (DE structure):

```
// struct directory
typedef struct DE
{
    char fileName[32];
    int offset;                 //= i in array for both directory entries array and extents array
    int fileSize;
    time_t createdTime;
    time_t modifiedTime;
    time_t lastAccessedTime;
    char isDirectory;       //0 for file, 1 for directory
} DE;
```

The DE structure, standing for Directory Entry, represents a single entry in a file system directory. It is defined using the C programming language syntax. Here's a detailed description of each of its fields:

Variables Usage:

- fileName: A character array of size 32, which holds the name of the file or directory.

- offset: An integer that stores the position (index) of the directory entry in some array of directory entries. This offset also corresponds to the index in another array that stores the extents (blocks of storage) of the file or directory.

- fileSize: An integer that represents the size of the file in bytes. For directories, this size may have a different meaning, or could be set to zero or a fixed value.

- createdTime: time_t value that records the creation time of the file or directory.

- modifiedTime: time_t value that records the last modification time of the file or directory.

- lastAccessedTime: time_t value that records the last time the file or directory was accessed.

- isDirectory: A character used as a boolean flag to indicate whether the directory entry is a file or a directory. The value '0' represents a file, and '1' represents a directory.

This structure encapsulates the metadata of a file or directory in a file system, allowing for the management, retrieval, and manipulation of files and directories. The offset field suggests that this structure might be part of a larger file system implementation that involves arrays of entries and extents, facilitating file system navigation and file management operations.

Approach / What we did:

We served as the foundational routine to initialize a file system, encompassing the initialization of the Volume Control Block (VCB), free space management, and the root directory. It started by announcing its operation, indicating the number of blocks and the block size of the file system. Two buffers are then allocated: one for the VCB and another for a directory entry (DE). Our group commenced by attempting to read the VCB from the first block of the file system using the LBAread function. A signature check follows, determining if the volume has already been initialized. If the signature matches, it indicates that the file system is already set up, and the function proceeds to load the existing free space management structure. If the signature does not match, indicating an uninitialized file system, the function transitions into the initialization phase. It starts by initializing the VCB, setting up necessary metadata and parameters for the file system's operation. Subsequently, we initialized the free space management and the root directory. It printed out status messages and return values from these operations, providing insight into their potential issues. In the final part of the function, an additional directory initialization was attempted using the previously allocated DE buffer. Overall, we meticulously initialized and validated each component of the file system even though we did not separate each file.

Issues / Resolutions:

There were no issues in the programming of allocateBlocks() as programming was done according to deep planning and it was coded with systematic testing.
Issues in initDir() below

<u>Issue 1:</u>

We realized there was no if/else structure applied to the hard coded initialization of the first two entries, so we wrote a different routine for those *inside* the if/else condition for null/root creation or else for subdirectory. There was also no print checking of the parent passed in so that we could test for the else condition creating a subdirectory- this is why nothing printed when trying to call again for a subdirectory when passing in the new root. And last, we never checked

for a condition where there was no space for the directory requested- instead of wrapping everything we had in an if statement, which would look messy, we just had an exit statement with proper closure of function resources. This implied moving our closure of the tempArray of extent structs into conditional checks, as we wanted to close it as soon as possible but only when necessary.

```
 GNU nano 2.9.3                                                    fsInit.c

    dir[0].lastAccessedTime = t;
    dir[0].isDirectory = 1;

    struct DE *p;  // Declare p here to make it accessible in the if-else blocks

    if (parent == NULL)
    {
        // If there's no parent (root directory), set p to the root directory entry
        p = &dir[0];  // Removed the struct DE * declaration
    }
    else
    {
        // If there's a parent directory, set p to the provided parent
        p = parent;  // Removed the struct DE * declaration
    }

    // Initialize ".." entry for root
    strcpy(dir[1].fileName, "..");
    dir[1].fileSize = p->fileSize;
    dir[1].location = p->location;
    dir[1].createdTime = p->createdTime;
    dir[1].modifiedTime = p->modifiedTime;
    dir[1].lastAccessedTime = p->lastAccessedTime;
    dir[1].isDirectory = p->isDirectory;

    // Now write it to disk
    LBAwrite(dir, blocksNeeded, 0);

        free(tempArray);

        //save location of block number After filecontrolblock (directories entries) to set as extent location in each dir Entry
        //in each directory entry set offset to i

        //mark freespacemap extent returned by allocateBlocks to 1's
        //return start of directory location
        return 0;
/*
        // 0 -> used 1 -> unused
        for (int i = 2; i < actualDirEntries; i++)
        {
```

Resolution 1:

```
        dir[i].fileName[i]= '\0'; // Null-terminated name
        dir[i].fileSize = bytesNeeded;
        dir[i].offset= i;
        time_t t = time(NULL);
        dir[i].createdTime = t;
        dir[i].modifiedTime = t;
        dir[i].lastAccessedTime = t;
        dir[i].isDirectory = 0;

    // printf("filename: %c\n", dir[i].fileName[i]);
    }

    if (p == NULL)
    {
        // Initialize  . and .. manually

        // If there's no parent (root directory), set p to the root directory entry
        p = &dir[0];  // Removed the struct DE * declaration
        // Initialize "." entry
        strcpy(p->fileName, ".");
        p->fileSize = 0;
        p->offset = 0;
        time_t t = time(NULL);
        p->createdTime = t;
        p->modifiedTime = t;
        p->lastAccessedTime = t;
        p->isDirectory = 1;
```

Issue 2:

The second significant challenge encountered by our group pertains to the debugging of the file system implementation. Debugging in such a complex environment requires a meticulous approach to trace back issues, understand the flow of execution, and verify the data at various points in the program. The intricacy of the system demands a detailed and granular insight into the program, which was tough without handling the proper debugging.

Resolution 2:

To effectively address this debugging challenge, our group has implemented a strategy of extensive printf statement insertion throughout the codebase. This approach enables inspection of variable values and data structures. We strategically place these printf statements to ensure that they encapsulate key areas of the code, such as function entry and exit points, critical conditional branches, and after significant operations. To further enhance this strategy, we maintain clarity and relevance in our debug messages, ensuring that they provide sufficient context and information to aid in troubleshooting. Through this structured and methodical approach to debugging, our group aims to swiftly identify, understand, and resolve issues within the file system somewhat.

```
    // Just for variable check
    printf("bitmapSize: %d\n", bitmapSize); // 2560
    printf("blockCount : %d\n", blockCount); // 19531
    printf("bytesPerBlock: %d\n", bytesPerBlock); // 512
    printf("FreeSpace block counts: %d\n", freeSpaceBlockCounts); // 5
```

```
            // end tempArray loop
/*
                    printf("bitmap printed, bits order low to high: ");
                    //printbitmap allocateblock marks FSM
                    for (int i = 0; i <= exitByte; i++)
                            {
                            printf("[");
                            for(int j=0; j < 8 ; j++)
                                    {
                                    printf("%d", !!((bitmapGlobal[i] >> j) & 0x01));
                                    }
                            printf("]");
                            }
                    printf("|end bitmap \n");
*/
```

```
// (2) Initialize free space management
int startFreeSpaceManagement = initFreeSpace(numberOfBlocks, blockSize);
printf("start FSM: %d\n",startFreeSpaceManagement);

// (3) Initialize root directory
int startDirectory = initDir(DEFAULT_ENTRIES, NULL);
printf("startDirectory: %d\n", startDirectory);

int returnCheck= LBAread(de, 1, 6);
printf("returnCheck: %d, de[0].fileName: %s\n", returnCheck, de[0].fileName);

int subDirReturn = initDir(DEFAULT_ENTRIES, *&de);
printf("subDirReturn: %d\n", subDirReturn);
```

Issue 3:

The current state of our file system implementation presents a significant organizational challenge, as the entirety of the code resides within a single file, fsInit.c. We aim to distribute the code across multiple files, specifically vcb.c, freespace.c, and directory.c, as well as those header files, to encapsulate distinct functionalities related to the Volume Control Block, free space management, and directory handling, respectively. However, this transition has been fraught with challenges, particularly in maintaining parameter consistency across the various functions and modules. Additionally, there may be hidden dependencies and implicit assumptions in the code that were manageable in a single file structure but become problematic when the code is separated across multiple files. These issues have led to functionality problems and unexpected behavior in the separated modules.

Still issue 3: (no resolution until now)

We failed to distribute fsInit.c to meaningful separate files due to the time limit these days. However, we are going to inspect what the issue is later on so that we can profoundly deal with those three separate files. Consequently, so far, our whole code is in fsInit.c and no header file.

Screenshots (compilation):

Case 1 (when volume is not initialized):

```
student@student-VirtualBox:~/Documents/csc415-filesystem-jeawanjang$ 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-jeawanjang$ 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
signature: 123456789
SIGNATURE: 1234567890


 Entered condition to initialize volume
***** After Initializing VCB *****
signature: 1234567890
total num blocks: 19531
block size: 512
bitmapSize: 2560
blockCount : 19531
bytesPerBlock: 512
FreeSpace block counts: 5
start FSM: 1
p filename: (null)

INITDIR CALLED
byteNeeded: 4096, blocksNeeded: 8, actualDirEntries: 56
allocateBlocks() called
required: 8 blocks, minperextent: 8 blocks
extent made ->start: 6, ->count: 8
block: 6, tempByte: 0, tempBitIndex: 6, exitByte: 1, exitBit: 6
dir[0].fileName: .
dir[1].fileName: ..
startDirectory: 6
returnCheck: 1, de[0].fileName: .
p filename: .

INITDIR CALLED
byteNeeded: 4096, blocksNeeded: 8, actualDirEntries: 56
allocateBlocks() called
required: 8 blocks, minperextent: 8 blocks
extent made ->start: 14, ->count: 8
block: 14, tempByte: 1, tempBitIndex: 6, exitByte: 2, exitBit: 6
locaiton rootGlobal: block: 6
else root copy returnCheck: 1, rootDir[0].fileName: .
check subdir[0].fileName: .
dir[1].fileName: .. at block 14 subdirectory
subDirReturn: 14
|---------------------------------|
|------- 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 >
```

case 2 (when volume is initialized already):

```
student@student-VirtualBox:~/Documents/csc415-filesystem-jeawanjang$ 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
signature: 1234567890
SIGNATURE: 1234567890


Volume is already initialized
bitmapSize: 2560
blockCount : 19531
BLOCK_SIZE= bytesPerBlock: 512
FreeSpace block counts: 5
bitmap printed, bits order low to high: start FSM: 1
|---------------------------------|
|------- 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 >
```

Analysis Hexdump:

- A dump (use the provided HexDump utility) of the volume file that shows the VCB, FreeSpace, and complete root directory.

  ☐ **VCB**

```
student@student-VirtualBox:~/Documents/GeriatricCats$ ./Hexdump/hexdump.linux SampleVolume --start 1 --count 1
Dumping file SampleVolume, starting at block 1 for 1 block:

000200: D2 02 96 49 00 00 00 00  4B 4C 00 00 00 02 00 00 | ◆.◆I....KL......
000210: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000220: 00 00 00 00 00 00 00 00  AE 3A 3C 65 00 00 00 00 | ........◆:<e....
000230: AE 3A 3C 65 00 00 00 00  AE 3A 3C 65 00 00 00 00 | ◆:<e....◆:<e....
000240: 01 00 00 00 00 00 00 00  2E 2E 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  01 00 00 00 00 00 00 00 | ................
000270: AE 3A 3C 65 00 00 00 00  AE 3A 3C 65 00 00 00 00 | ◆:<e....◆:<e....
000280: AE 3A 3C 65 00 00 00 00  01 00 00 00 00 00 00 00 | ◆:<e............
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 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 | ................

student@student-VirtualBox:~/Documents/GeriatricCats$
```

D2 02 96 49 00 00 00 00 (= 0x499602D2): These 8 bytes correspond to the long signature in the VCB structure. Based on the little endian format, D2 02 96 49 is 1234567890, which matches the signature that we've defined.

4B 4C 00 00 (= 0x00004C4B): It represents the totalNumBlocks, which is 19531. Since it is integer, it allocates 4 bytes.

00 02 00 00 (= 0x00000200): Just as totalNumBlocks, sizeOfBlock is 4 bytes and manifests the value of 512.

*****

StartFreeSpaceManagement, startDirectory: We got 0, which is not set, but I assume that since we initialize in initFileSystem at the end and it works well, we should get 1 for FSM and 6 for Dir.

- ☐ **FreeSpace**

```
student@student-VirtualBox:~/Documents/GeriatricCats$ ./Hexdump/hexdump.linux SampleVolume --start 2 --count 5
Dumping file SampleVolume, starting at block 2 for 5 blocks:

000400: 3F 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 00 00 00 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 | ................
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 | ................
000780: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000790: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0007A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0007B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0007C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0007D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0007E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0007F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
```

```
000800: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000810: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000820: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000830: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000840: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000850: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000860: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000870: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000880: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000890: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0008A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0008B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0008C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0008D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0008E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0008F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
```

```
000900: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000910: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000920: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000930: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000940: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000950: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000960: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000970: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000980: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000990: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0009A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0009B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0009C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0009D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0009E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0009F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................

000A00: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000A10: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000A20: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000A30: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000A40: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000A50: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000A60: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000A70: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000A80: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000A90: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000AA0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000AB0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000AC0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000AD0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000AE0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000AF0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
```

```
000B00: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000B10: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000B20: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000B30: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000B40: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000B50: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000B60: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000B70: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000B80: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000B90: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000BA0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000BB0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000BC0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000BD0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000BE0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000BF0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
```

```
000C00: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000C10: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000C20: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000C30: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000C40: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000C50: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000C60: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000C70: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000C80: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000C90: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000CA0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000CB0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000CC0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000CD0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000CE0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000CF0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................

000D00: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000D10: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000D20: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000D30: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000D40: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000D50: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000D60: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000D70: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000D80: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000D90: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000DA0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000DB0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000DC0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000DD0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000DE0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000DF0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................

student@student-VirtualBox:~/Documents/GeriatricCats$
```

Hexadecimal value *3F* at the beginning is equivalent to 00111111 and indicates that within a series of 8 blocks, 6 are used and the last two are free. Line numbers [410 to DFO] in decimal is [3568 to 1040] = 2544 bytes of 0 + 16 bytes with 3F marked, represents the expected 2560 bytes of the bitmap (5 blocks x 512 bytes), where the each 0 value represents 8 contiguous blocks on the disk that are free and available for allocation.

- ☐ **Root Directory**

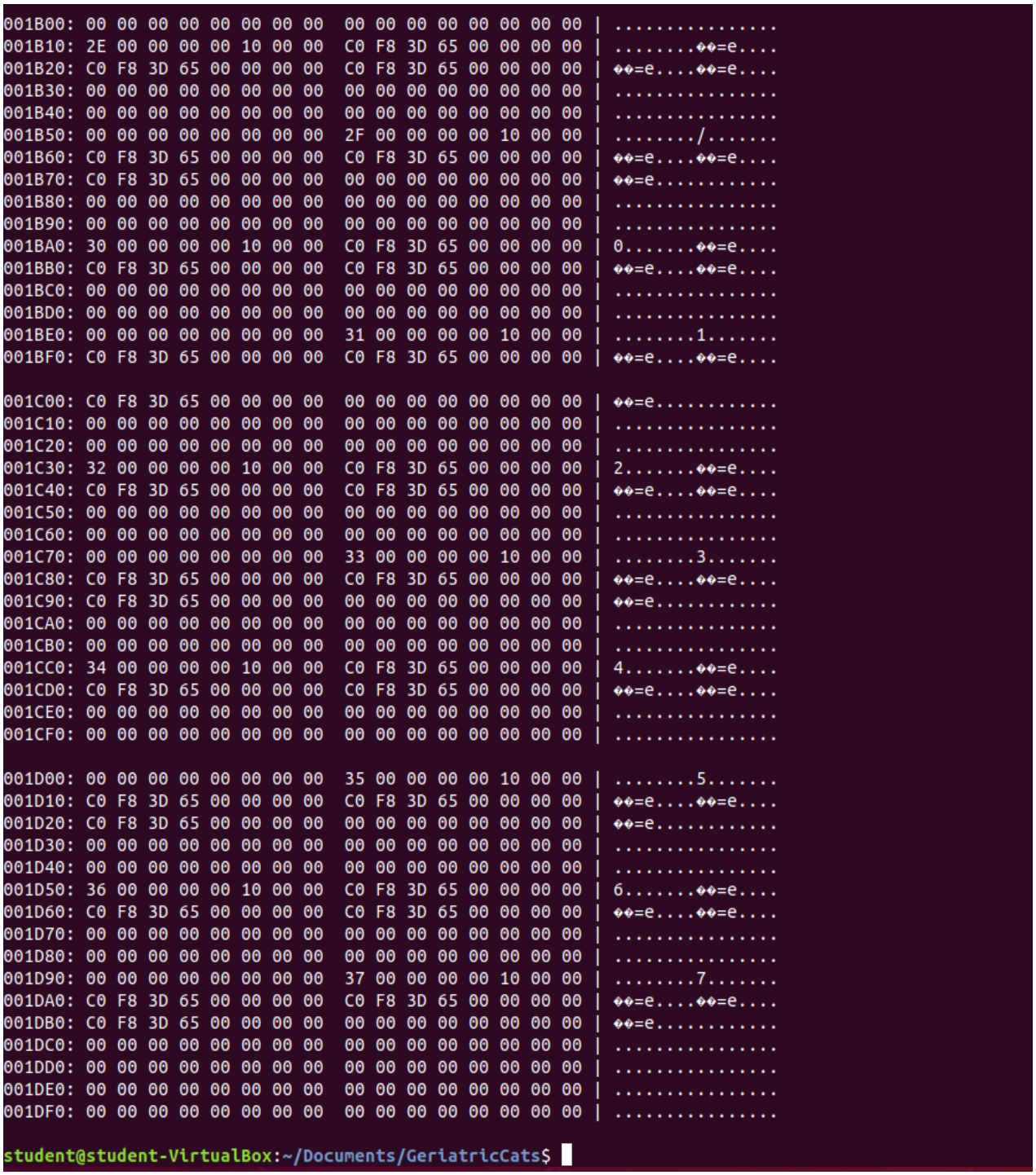
```
student@student-VirtualBox:~/Documents/GeriatricCats$ ./Hexdump/hexdump.linux SampleVolume --start 7 --count 8
Dumping file SampleVolume, starting at block 7 for 8 blocks:

000E00: 2E 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000E10: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000E20: 00 00 00 00 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ........◆◆=e....
000E30: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ◆◆=e....◆◆=e....
000E40: 01 00 00 00 00 00 00 00  2E 2E 00 00 00 00 00 00 | ................
000E50: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000E60: 00 00 00 00 00 00 00 00  01 00 00 00 00 00 00 00 | ................
000E70: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ◆◆=e....◆◆=e....
000E80: C0 F8 3D 65 00 00 00 00  01 00 00 00 00 00 00 00 | ◆◆=e............
000E90: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000EA0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000EB0: 02 00 00 00 00 10 00 00  C0 F8 3D 65 00 00 00 00 | ........◆◆=e....
000EC0: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ◆◆=e....◆◆=e....
000ED0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000EE0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000EF0: 00 00 00 00 00 00 00 00  03 00 00 00 00 10 00 00 | ................
```

```
000F00: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ▯▯=e....▯▯=e....
000F10: C0 F8 3D 65 00 00 00 00  00 00 00 00 00 00 00 00 | ▯▯=e............
000F20: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000F30: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000F40: 04 00 00 00 00 10 00 00  C0 F8 3D 65 00 00 00 00 | ........▯▯=e....
000F50: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ▯▯=e....▯▯=e....
000F60: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000F70: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000F80: 00 00 00 00 00 00 00 00  05 00 00 00 00 10 00 00 | ................
000F90: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ▯▯=e....▯▯=e....
000FA0: C0 F8 3D 65 00 00 00 00  00 00 00 00 00 00 00 00 | ▯▯=e............
000FB0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000FC0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000FD0: 06 00 00 00 00 10 00 00  C0 F8 3D 65 00 00 00 00 | ........▯▯=e....
000FE0: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ▯▯=e....▯▯=e....
000FF0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................

001000: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001010: 00 00 00 00 00 00 00 00  07 00 00 00 00 10 00 00 | ................
001020: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ▯▯=e....▯▯=e....
001030: C0 F8 3D 65 00 00 00 00  00 00 00 00 00 00 00 00 | ▯▯=e............
001040: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001050: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001060: 08 00 00 00 00 10 00 00  C0 F8 3D 65 00 00 00 00 | ........▯▯=e....
001070: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ▯▯=e....▯▯=e....
001080: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001090: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0010A0: 00 00 00 00 00 00 00 00  09 00 00 00 00 10 00 00 | ................
0010B0: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ▯▯=e....▯▯=e....
0010C0: C0 F8 3D 65 00 00 00 00  00 00 00 00 00 00 00 00 | ▯▯=e............
0010D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0010E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0010F0: 0A 00 00 00 00 10 00 00  C0 F8 3D 65 00 00 00 00 | ........▯▯=e....

001100: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ▯▯=e....▯▯=e....
001110: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001120: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001130: 00 00 00 00 00 00 00 00  0B 00 00 00 00 10 00 00 | ................
001140: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ▯▯=e....▯▯=e....
001150: C0 F8 3D 65 00 00 00 00  00 00 00 00 00 00 00 00 | ▯▯=e............
001160: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001170: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001180: 0C 00 00 00 00 10 00 00  C0 F8 3D 65 00 00 00 00 | ........▯▯=e....
001190: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ▯▯=e....▯▯=e....
0011A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0011B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0011C0: 00 00 00 00 00 00 00 00  0D 00 00 00 00 10 00 00 | ................
0011D0: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ▯▯=e....▯▯=e....
0011E0: C0 F8 3D 65 00 00 00 00  00 00 00 00 00 00 00 00 | ▯▯=e............
0011F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
```

```
001200: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001210: 0E 00 00 00 00 10 00 00  C0 F8 3D 65 00 00 00 00 | ........ø=e....
001220: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ø=e....ø=e....
001230: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001240: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001250: 00 00 00 00 00 00 00 00  0F 00 00 00 00 10 00 00 | ................
001260: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ø=e....ø=e....
001270: C0 F8 3D 65 00 00 00 00  00 00 00 00 00 00 00 00 | ø=e............
001280: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001290: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0012A0: 10 00 00 00 00 10 00 00  C0 F8 3D 65 00 00 00 00 | ........ø=e....
0012B0: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ø=e....ø=e....
0012C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0012D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0012E0: 00 00 00 00 00 00 00 00  11 00 00 00 00 10 00 00 | ................
0012F0: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ø=e....ø=e....

001300: C0 F8 3D 65 00 00 00 00  00 00 00 00 00 00 00 00 | ø=e............
001310: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001320: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001330: 12 00 00 00 00 10 00 00  C0 F8 3D 65 00 00 00 00 | ........ø=e....
001340: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ø=e....ø=e....
001350: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001360: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001370: 00 00 00 00 00 00 00 00  13 00 00 00 00 10 00 00 | ................
001380: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ø=e....ø=e....
001390: C0 F8 3D 65 00 00 00 00  00 00 00 00 00 00 00 00 | ø=e............
0013A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0013B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0013C0: 14 00 00 00 00 10 00 00  C0 F8 3D 65 00 00 00 00 | ........ø=e....
0013D0: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ø=e....ø=e....
0013E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0013F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................

001400: 00 00 00 00 00 00 00 00  15 00 00 00 00 10 00 00 | ................
001410: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ø=e....ø=e....
001420: C0 F8 3D 65 00 00 00 00  00 00 00 00 00 00 00 00 | ø=e............
001430: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001440: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001450: 16 00 00 00 00 10 00 00  C0 F8 3D 65 00 00 00 00 | ........ø=e....
001460: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ø=e....ø=e....
001470: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001480: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001490: 00 00 00 00 00 00 00 00  17 00 00 00 00 10 00 00 | ................
0014A0: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ø=e....ø=e....
0014B0: C0 F8 3D 65 00 00 00 00  00 00 00 00 00 00 00 00 | ø=e............
0014C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0014D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0014E0: 18 00 00 00 00 10 00 00  C0 F8 3D 65 00 00 00 00 | ........ø=e....
0014F0: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ø=e....ø=e....
```

```
001500: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001510: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001520: 00 00 00 00 00 00 00 00  19 00 00 00 00 10 00 00 | ................
001530: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ◆◆=e....◆◆=e....
001540: C0 F8 3D 65 00 00 00 00  00 00 00 00 00 00 00 00 | ◆◆=e............
001550: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001560: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001570: 1A 00 00 00 00 10 00 00  C0 F8 3D 65 00 00 00 00 | ........◆◆=e....
001580: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ◆◆=e....◆◆=e....
001590: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0015A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0015B0: 00 00 00 00 00 00 00 00  1B 00 00 00 00 10 00 00 | ................
0015C0: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ◆◆=e....◆◆=e....
0015D0: C0 F8 3D 65 00 00 00 00  00 00 00 00 00 00 00 00 | ◆◆=e............
0015E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0015F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................

001600: 1C 00 00 00 00 10 00 00  C0 F8 3D 65 00 00 00 00 | ........◆◆=e....
001610: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ◆◆=e....◆◆=e....
001620: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001630: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001640: 00 00 00 00 00 00 00 00  1D 00 00 00 00 10 00 00 | ................
001650: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ◆◆=e....◆◆=e....
001660: C0 F8 3D 65 00 00 00 00  00 00 00 00 00 00 00 00 | ◆◆=e............
001670: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001680: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001690: 1E 00 00 00 00 10 00 00  C0 F8 3D 65 00 00 00 00 | ........◆◆=e....
0016A0: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ◆◆=e....◆◆=e....
0016B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0016C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0016D0: 00 00 00 00 00 00 00 00  1F 00 00 00 00 10 00 00 | ................
0016E0: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ◆◆=e....◆◆=e....
0016F0: C0 F8 3D 65 00 00 00 00  00 00 00 00 00 00 00 00 | ◆◆=e............

001700: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001710: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001720: 20 00 00 00 00 10 00 00  C0 F8 3D 65 00 00 00 00 |  .......◆◆=e....
001730: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ◆◆=e....◆◆=e....
001740: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001750: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001760: 00 00 00 00 00 00 00 00  21 00 00 00 00 10 00 00 | ........!.......
001770: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ◆◆=e....◆◆=e....
001780: C0 F8 3D 65 00 00 00 00  00 00 00 00 00 00 00 00 | ◆◆=e............
001790: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0017A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0017B0: 22 00 00 00 00 10 00 00  C0 F8 3D 65 00 00 00 00 | ".......◆◆=e....
0017C0: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ◆◆=e....◆◆=e....
0017D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0017E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0017F0: 00 00 00 00 00 00 00 00  23 00 00 00 00 10 00 00 | ........#.......
```

```
001800: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ◆◆=e....◆◆=e....
001810: C0 F8 3D 65 00 00 00 00  00 00 00 00 00 00 00 00 | ◆◆=e............
001820: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001830: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001840: 24 00 00 00 00 10 00 00  C0 F8 3D 65 00 00 00 00 | $.......◆◆=e....
001850: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ◆◆=e....◆◆=e....
001860: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001870: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001880: 00 00 00 00 00 00 00 00  25 00 00 00 00 10 00 00 | ........%.......
001890: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ◆◆=e....◆◆=e....
0018A0: C0 F8 3D 65 00 00 00 00  00 00 00 00 00 00 00 00 | ◆◆=e............
0018B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0018C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0018D0: 26 00 00 00 00 10 00 00  C0 F8 3D 65 00 00 00 00 | &.......◆◆=e....
0018E0: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ◆◆=e....◆◆=e....
0018F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................

001900: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001910: 00 00 00 00 00 00 00 00  27 00 00 00 00 10 00 00 | ........'.......
001920: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ◆◆=e....◆◆=e....
001930: C0 F8 3D 65 00 00 00 00  00 00 00 00 00 00 00 00 | ◆◆=e............
001940: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001950: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001960: 28 00 00 00 00 10 00 00  C0 F8 3D 65 00 00 00 00 | (.......◆◆=e....
001970: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ◆◆=e....◆◆=e....
001980: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001990: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0019A0: 00 00 00 00 00 00 00 00  29 00 00 00 00 10 00 00 | ........).......
0019B0: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ◆◆=e....◆◆=e....
0019C0: C0 F8 3D 65 00 00 00 00  00 00 00 00 00 00 00 00 | ◆◆=e............
0019D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0019E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0019F0: 2A 00 00 00 00 10 00 00  C0 F8 3D 65 00 00 00 00 | *.......◆◆=e....

001A00: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ◆◆=e....◆◆=e....
001A10: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001A20: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001A30: 00 00 00 00 00 00 00 00  2B 00 00 00 00 10 00 00 | ........+.......
001A40: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ◆◆=e....◆◆=e....
001A50: C0 F8 3D 65 00 00 00 00  00 00 00 00 00 00 00 00 | ◆◆=e............
001A60: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001A70: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001A80: 2C 00 00 00 00 10 00 00  C0 F8 3D 65 00 00 00 00 | ,.......◆◆=e....
001A90: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ◆◆=e....◆◆=e....
001AA0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001AB0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001AC0: 00 00 00 00 00 00 00 00  2D 00 00 00 00 10 00 00 | ........-.......
001AD0: C0 F8 3D 65 00 00 00 00  C0 F8 3D 65 00 00 00 00 | ◆◆=e....◆◆=e....
001AE0: C0 F8 3D 65 00 00 00 00  00 00 00 00 00 00 00 00 | ◆◆=e............
001AF0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
```



Based on where the free space map ends at 000DFO, the next line E00 represents the start of the file control block of directory entries. As we expect 8 blocks for the default entries amount at 512 bytes/block, the total bytes for the FCB is 4096 bytes. 4096 bytes/16 bytes per line is 256 lines. From line E00 hex to decimal 3584 bytes + 4096 bytes expected (or 256 lines) = 7664 bytes, or line 479 decimal/ 0001DF0 in hex.

directoryEntry structure {

```
    char fileName[32];
    int offset;          //= i in array for both directory entries array and extents array
    int fileSize;
    time_t createdTime;
    time_t modifiedTime;
    time_t lastAccessedTime;
    char isDirectory;    //0 for file, 1 for directory
}
```

Each directory entry has 65 bytes of members, but we allocate 72 bytes in our algorithm which determines the maximum fit of entries per block. We see that the name "." is 2E in hex, and the string name is 32 bytes or 2 lines at 000E00-000E10. The 4 bytes of int for the offset member is 0 for the root entry, so the first four bytes of 000E20 are 0. The 4 bytes of int for the fileSize member is 0 for the root entry, so the next 4 bytes of 000E20 are 0. Following the fileSize int, the last 8 bytes of the line of 000E20 is 65 3D F8 C0 00 00 00 00 for the time_t createdTime. This same value is repeated in the first 8 and last 8 bytes of the next line at 000E30, because we used the same value to store in modifiedTime and lastAccessedTime. At line 000E40, we have a 01 for the 1 byte char isDirectory, which is marked 1 in our program. The following 6 bytes are waste bytes from our algorithm. The exact same pattern for the structure is repeated at byte 8 in line 000E40, but with "2E 2E" of the 32 byte string because our second entry is "..". The offset at the appropriate position after a 32 byte string is 01, because the second directory entry will have an offset of 1 in our extent table array block. We can easily observe the offset of each entry increasing by 1 every 72 bytes or after 9 sections of 8 bytes, showing the directoryEntries as they index through the blocks remaining.

Work Table:

Team Work- Based on Github history, Google Doc History, and Meeting Hrs in SFSU portal

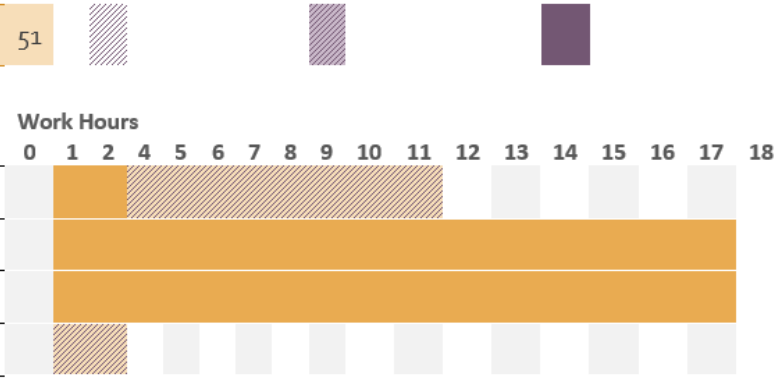
We met once/week for the first three weeks on Mondays, then 2x/week for the following week on Monday/Wednesday and then 3x the last week on Mon/Wed/Fri. Meetings were scheduled in-person in the library reserve study rooms, with a laptop hooked up to the tv for in person, and using the Discord screen share and audio/video lounge for whoever could not attend in person. We did not assign sections of work itself ahead of the entire plan, but maintained a plan and timeline of work progress to meet the deadline for Sunday, which is the exact time it took from our start to finish. Each meeting, someone would take the assignment of a section or portion of a section and complete it by the next meeting, and if they could not complete it, someone else would take the remaining work the morning before the meeting or the work would be done during the meeting to stay on track.

Overall

plan/meet/work/writeup/content engagement

Work Hours: 51

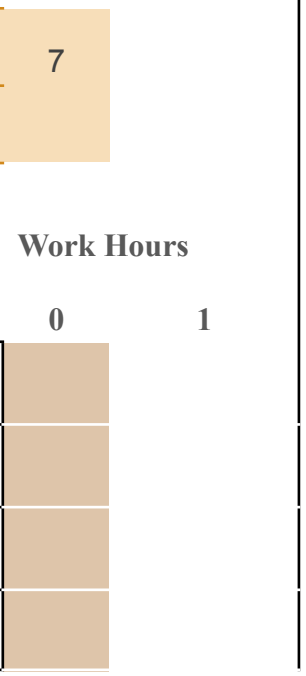
Member	Planning Work Ratio	Work Actual Ratio	Time Duration (hrs$)	PERCENT COMPLETE
Carlos	0.25	0.18	12	25%
Jaewan	0.25	0.25	18	100%
Paige	0.25	0.25	18	100%
Randale	0.25	0.05	3	25%



InitVCB

Work Hours: 1 7

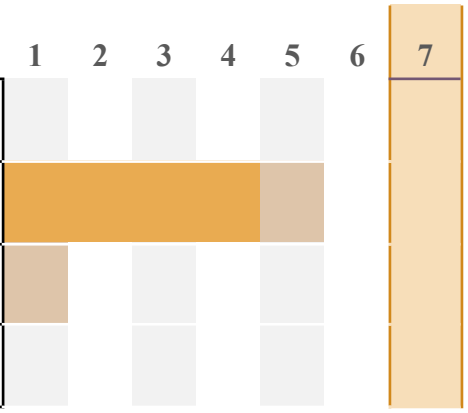
Member	Planning Work Ratio	Work Actual Ratio	Time Duration (hrs)	PERCENT COMPLETE
Carlos	0.25	0.25	0.25	25%
Jaewan	0.25	0.25	0.25	25%
Paige	0.25	0.25	0.25	25%
Randale	0.25	0.25	0.25	25%



Load/Init freespace

Work Hours 7

Member	Work Assignment Ratio	Work Actual Ratio	Time Duration (hrs)	PERCENT COMPLETE
Carlos	0.25	0	0	0%
Jaewan	0.25	0.77	6	77%
Paige	0	0.23	2	23%
Randale	0	0	0	0%



allocateBlocks()

Work Hours

Member	Planning Work Ratio	Work Actual Ratio	Time Duration (hrs)	PERCENT COMPLETE
Carlos	0.25	0.15	1	15%
Jaewan	0	0.05	0.5	5%
Paige	0.75	0.8	6	80%
Randale	0	0	0	0%

InitDir

Work Hours: 8

Member	Planning Work Ratio	Work Actual Ratio	Time Duration (hrs)	PERCENT COMPLETE
Carlos	0	0.15	1	**15%**
Jaewan	0	0.15	1	**15%**
Paige	0	0.65	6	**65%**
Randale	100	0.05	1	**5%**