Team: Michael

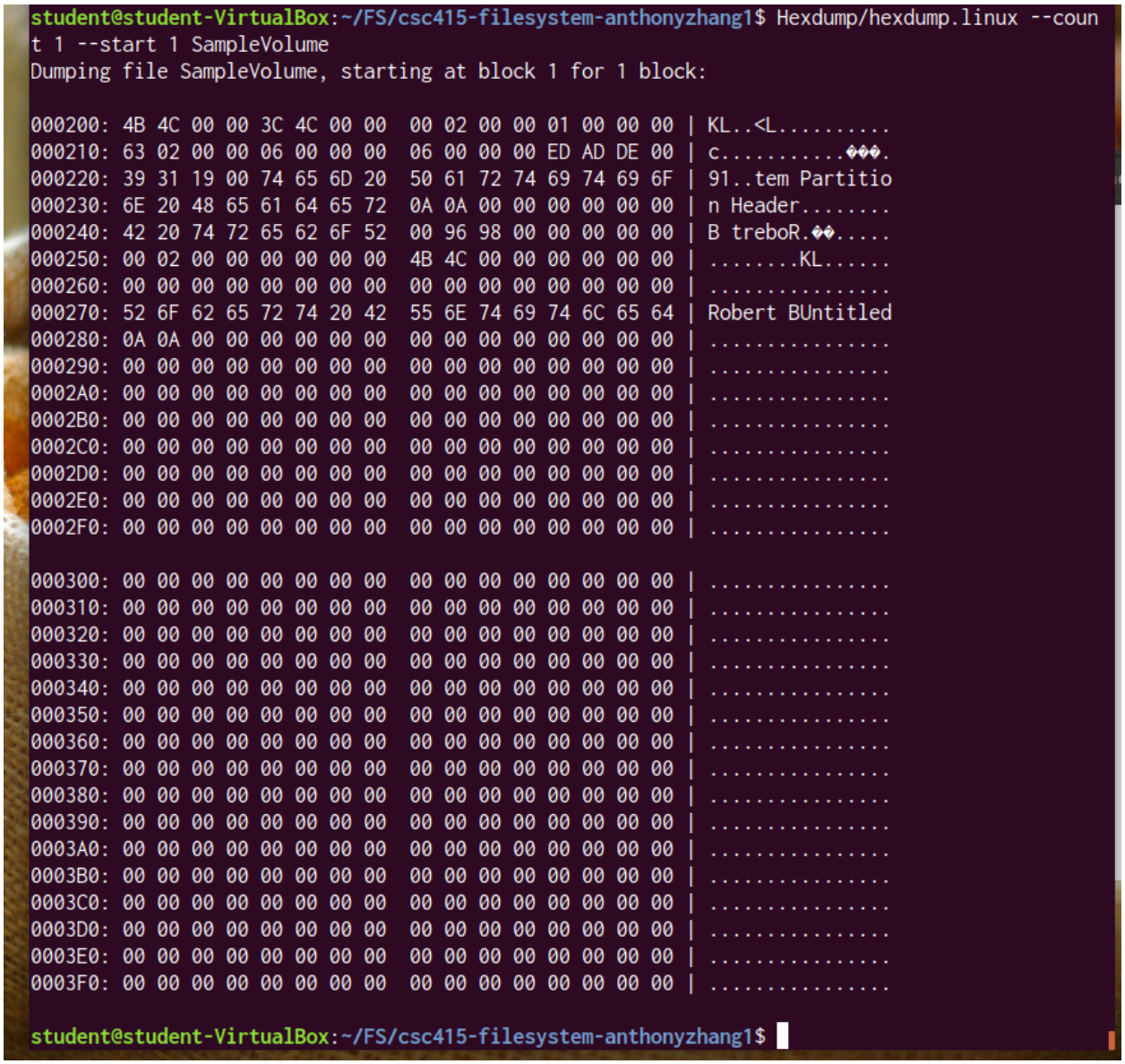
Team Members:
Edel Jhon Cenario (ID: 921121224)
Michael Wang (ID: 921460979)
Michael Widergren (ID: 921363622)
Anthony Zhang (ID: 921544101)

GitHub Name: anthonyzhang1

The dump of the volume file that shows the VCB, FreeSpace, and root directory:

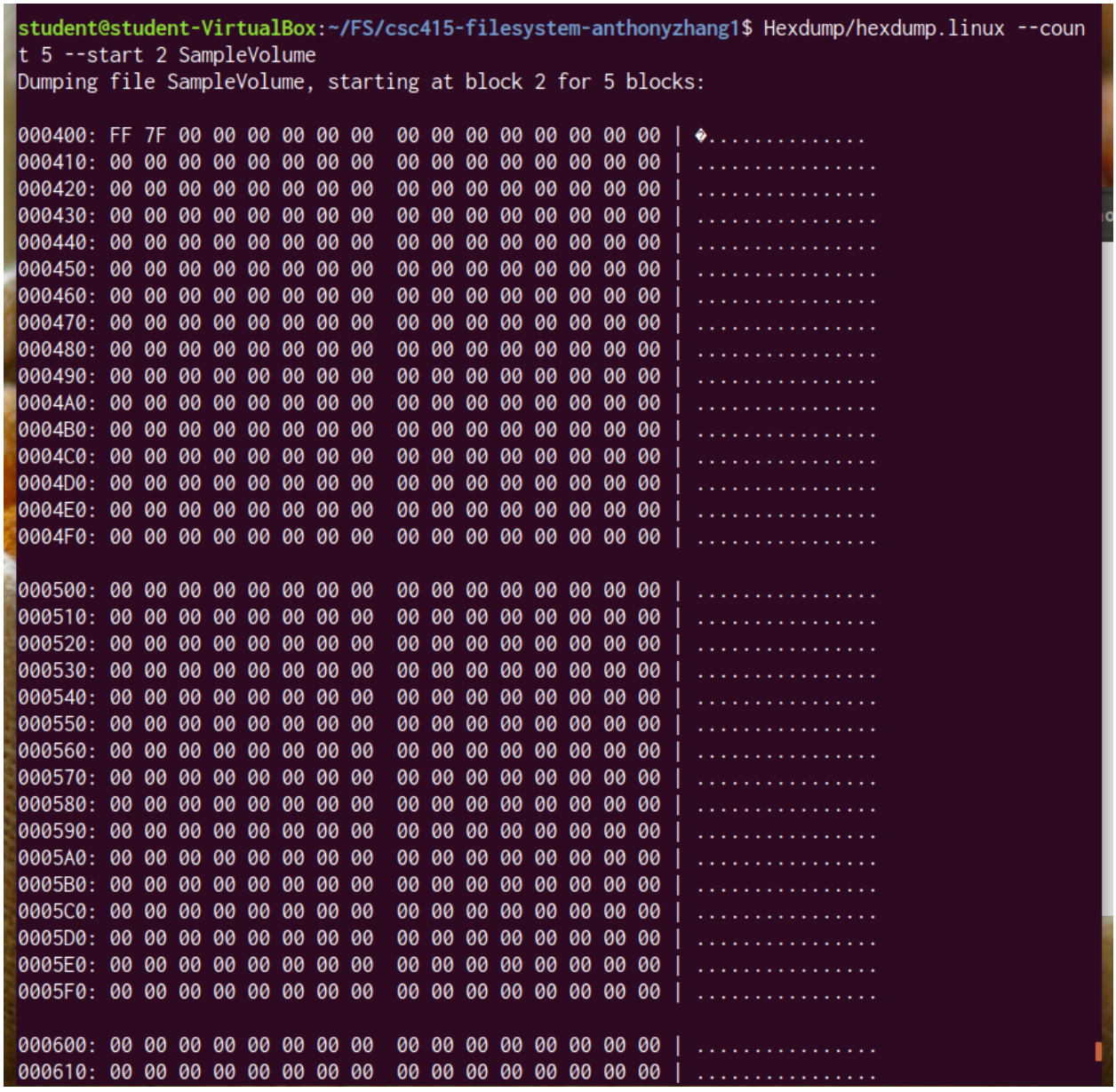
Block 0 (VCB):

We know this is the VCB block because our signature “DEADED” appears on the second line, on bytes 21C to 21E. It appears backward because of little-endian order.



Blocks 1-5 (Bitmap / Freespace):

We can verify this is our bitmap because 7F FF in hexadecimal converts to 0111 1111 1111 1111 in binary. Since we used 15 blocks in total for our VCB, bitmap, and root directory, we would expect to have 15 1's starting from bit 0 on the bitmap, as 1 represents a non-free block. Indeed, there are 15 contiguous 1's in the hexadecimal to binary conversion.



Blocks 6-15 (Root Directory):

The very first byte, E00, is the ASCII for the '.' character, which was used as the name for the first directory entry in our root directory. On bytes E44 to E45, the hexadecimal value 11 E0 maps to 4576 in decimal. 4576 is the value of the size data member of rootArray[0], which represents how many bytes the root directory takes up.

On bytes E58 to E59, we see 2E 2E. This is the string "..", which is the name of the second directory entry in our root directory. Also, FF FF FF FE appears multiple times, which is -2 in signed 2's complement. -2 is the value we use to check if a directory entry is free. This confirms that this hexdump is part of our root directory.

```
student@student-VirtualBox:~/FS/csc415-filesystem-anthonyzhang1$ Hexdump/hexdump.linux --count 9 --s
tart 7 SampleVolume
Dumping file SampleVolume, starting at block 7 for 9 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  00 00 00 00 00 00 00 00 | ................
000E30: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000E40: 06 00 00 00 E0 11 00 00  01 00 00 00 00 00 00 00 | ....�...........
000E50: 00 00 00 00 00 00 00 00  2E 2E 00 00 00 00 00 00 | ................
000E60: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000E70: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000E80: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000E90: 00 00 00 00 00 00 00 00  06 00 00 00 E0 11 00 00 | ............�...
000EA0: 01 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000EB0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000EC0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
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 FE FF FF FF  00 00 00 00 00 00 00 00 | ....����........

000F00: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000F10: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
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: 00 00 00 00 00 00 00 00  00 00 00 00 FE FF FF FF | ............����
000F50: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
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  00 00 00 00 00 00 00 00 | ................
000F90: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000FA0: 00 00 00 00 FE FF FF FF  00 00 00 00 00 00 00 00 | ....����........
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: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000FE0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000FF0: 00 00 00 00 00 00 00 00  00 00 00 00 FE FF FF FF | ............����

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  00 00 00 00 00 00 00 00 | ................
001020: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
```

Description of the VCB structure:

Our volume control block is a pointer in fsInit.c, for now.

The data members of our Volume Control Block:

- num_blocks: An **int** that stores the total number of blocks in our volume.
- num_free_blocks: An **int** that tracks how many free blocks we have.
- block_size: An **int** that stores the size of our blocks in bytes.
- bitmap_start: A **32-bit unsigned int** which stores the block number at which our bitmap starts.
- bitmap_size: A **32-bit unsigned int** whose value is how many integers we use in our bitmap to represent the bits.
- free_space_start: A **32-bit unsigned int** which stores the block number at which our free space starts.
- root_dir: An **int** that stores the block number of the root directory.
- signature: An **int** that acts as a flag as to whether our volume has been initialized yet. The value is a magic number.

Description of the Free Space structure:

Our Free Space structure is a bitmap that uses an integer array. First, we create our bitmap through a function called initializeBitMap, which takes two parameters.

The first parameter is the number of blocks in the volume. The parameter is used to calculate the number of bytes our bitmap will need.

The second parameter is the block size in bytes. This is used to measure how many blocks our bitmap takes up. By using the bitmap bytes derived from the first parameter, we can use: bitmap bytes + block size - 1 / block size to get the number of blocks our bitmap needs, rounded up.

What the bitmap allows us to do is, it allows us to represent each block in the volume with a single bit from the integer array. Then we can use the bit to check whether the block is free or not. The value 1 represents a used block, while 0 represents an empty block.

Description of the Directory system:

Our Directory is an array-based system. We are creating a root directory where all the folders and files will be stored. We created a structure that will define the entries in our directory.

Our structure is called dir_entry. These are its data members:

- name: A **C string** that will store the name/identifier for the entry.
- starting_block: An **int** that will store the starting block of the directory entry.
- size: An **int** that will store the size of the entry in bytes.
- is_directory: An **int** which will identify whether the entry is a file or a directory. If the value is 1, then it is a directory entry. If the value is 0, then it is a file.

We have also included metadata variables that will be implemented in the future. Their type is subject to change:

- creation_date: An **int** that will store the date the entry was created.
- last_modified: An **int** that will store the date the entry was last modified.
- last_opened: An **int** that will store the entry was last opened.

The table of who worked on which components:

Edel John Cenario	**PDF Write up**
Michael Wang	**Root Directory Initialization, Code Cleanup**
Michael Widergren	**Hexdump analysis, part PDF writeup**
Anthony Zhang	**File System Initialization, Volume Control Block Initialization, Bitmap Initialization, Root Directory Initialization, Hexdump Writeup, PDF Writeup Editing**

How our team worked together, how often we met, how we met, and how we divided up the tasks:

We used Discord as our main form of communication. We texted and called each other through it. We sent messages to each other at least weekly to coordinate on what we planned to do. We gave each other a heads up when a change was coming so everyone could do their pulls. And everyone brainstormed on how things could be done when something went wrong. We divided the tasks by working on the project whenever we could. If something needed to be done, then one of us would eventually work on it.

What issues we faced and how we resolved them:

Everyone had different viewpoints and ways of tackling the project at first which was our initial challenge. It was intimidating to get so many files and not know which one to start on and what to do with it. Despite not knowing where to start, we all came together and settled on where we should start: we should start by following the Steps for Milestone 1 PDF.

Initializing the bitmap was our first major struggle because we did not know how to do bit manipulations. We referenced this post to learn how to do bit manipulations: https://stackoverflow.com/questions/2525310/how-to-define-and-work-with-an-array-of-bits-in-c. Then, we were able to make some bit manipulation functions in fsInit.c, though we will later move those functions into a more appropriate file.

Also, we had a hard time keeping our bitmap persistent as we had difficulty coming up with a way to keep track of it. Ultimately we used a global array, though this is just a temporary solution. It did what it was supposed to though, in that it was persistent and correctly stored the bits representing whether a block was used or free, as shown by the hexdump.

Another issue we had involved figuring out how to initialize the root directory. It took an all-nighter to finish that part. What really slowed us down was that we had to implement methods for finding free space and we had to learn how to manipulate bits. Furthermore, we were unsure on how the root directory was going to be formatted, and initially assumed a Root Directory was going to be in every block. We learnt that we were mistaken in our assumptions upon reading the Steps for Milestone 1 PDF. We followed that PDF step-by-step to then initialize the root directory.

Finally, we had some disagreements over how to interpret our hexdump and what the count and start blocks were supposed to be for our VCB, Bitmap, and root directory. Your video from class explained it very well and we were able to find our VCB, bitmap, and root directory with no further issues. Altogether, we made a great team and look forward to the next milestone.