TEAM CARROT

NAME	GITHUB NAME	STUDENT ID
Prit Panchani	PritPanchani	923603327
Viraj Hingu	vhingu28	923687905
Shiv Patel	Spatel2603	923603418
Krushit Moradiya	krushitmoradiya	923079492

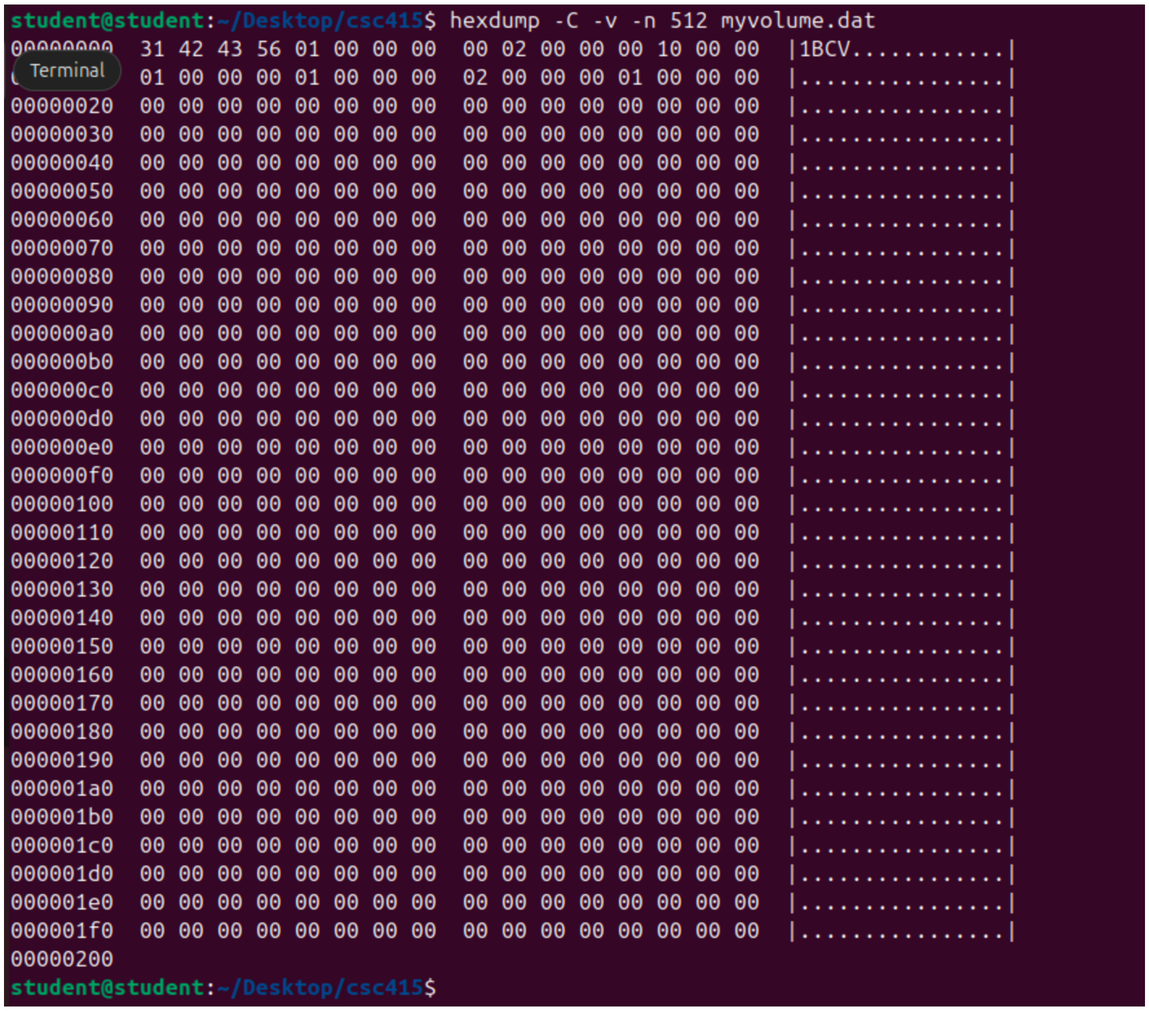
1. **A dump (use the provided HexDump utility) of the volume file that shows the VCB, FreeSpace, and complete root directory.**

Our team created a formatted volume file named myvolume.dat using our program format_volume.c.
This program initializes the volume by writing three main structures:

- Block 0: Volume Control Block (VCB) — stores basic file-system metadata such as block size, total blocks, and starting points for free space and root directory.
- Block 1: Free-Space Map — tracks which blocks in the volume are free or allocated.
- Block 2: Root Directory — initialized with "." (current directory) and ".." (parent directory) entries.

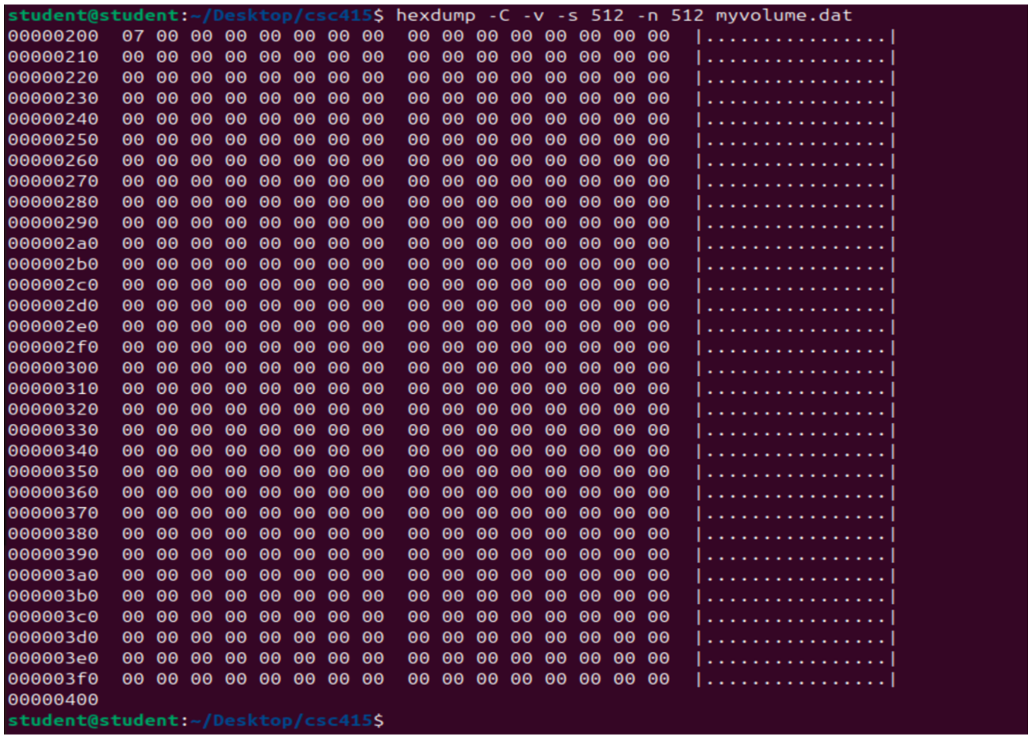
To verify that these components were correctly written, we used the HexDump utility to display the raw data from each block in hexadecimal form.

HexDump of Block 0 (Volume Control Block) showing the "VCB1" signature and stored file-system metadata.



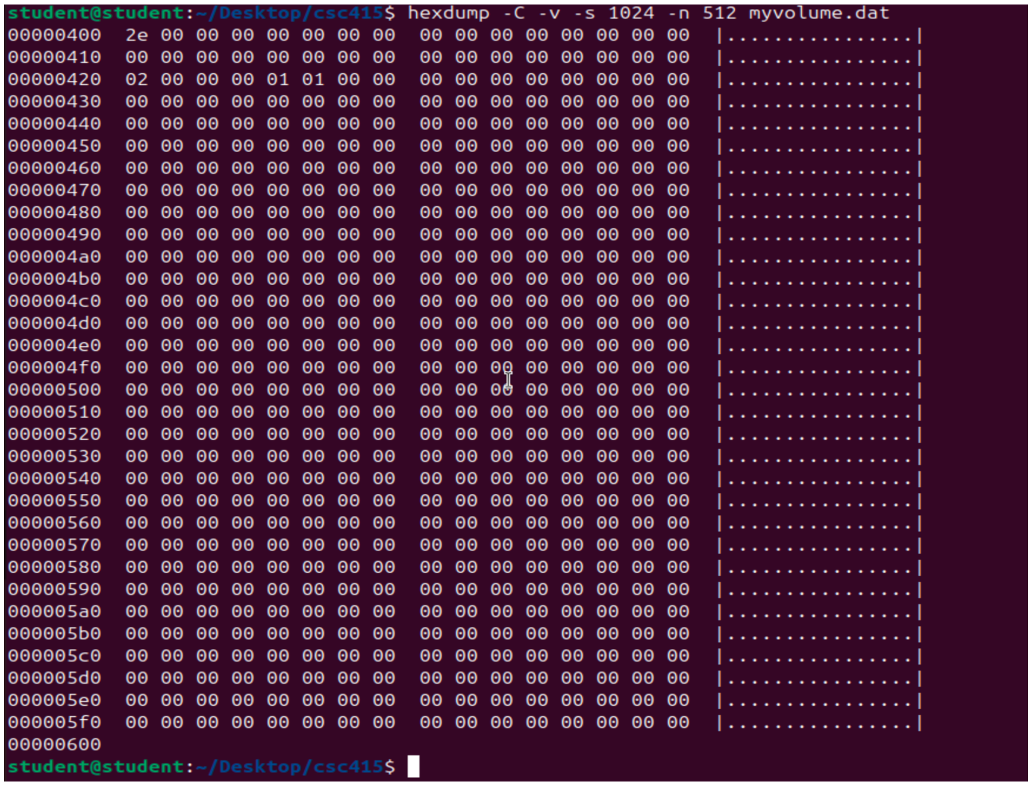
```
student@student:~/Desktop/csc415$ hexdump -C -v -n 512 myvolume.dat
00000000  31 42 43 56 01 00 00 00  00 02 00 00 00 10 00 00  |1BCV............|
 Terminal 01 00 00 00 01 00 00 00  02 00 00 00 01 00 00 00  |................|
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000100  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000110  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000120  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000130  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000140  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000150  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000160  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000170  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000180  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000190  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000001a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000001b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000001c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000001d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000001e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000001f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000200
student@student:~/Desktop/csc415$
```

HexDump of Block 1 (Free-Space Map) showing the bitmap where used and free blocks are tracked.



```
student@student:~/Desktop/csc415$ hexdump -C -v -s 512 -n 512 myvolume.dat
00000200  07 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000210  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000220  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000230  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000240  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000250  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000260  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000270  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000280  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000290  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000002a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000002b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000002c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000002d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000002e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000002f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000300  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000310  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000320  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000330  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000340  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000350  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000360  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000370  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000380  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000390  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000003a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000003b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000003c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000003d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000003e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000003f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000400
student@student:~/Desktop/csc415$
```

HexDump of Block 2 (Root Directory) showing initialization of the "." and ".." directory entries.



```
student@student:~/Desktop/csc415$ hexdump -C -v -s 1024 -n 512 myvolume.dat
00000400  2e 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000410  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000420  02 00 00 00 01 01 00 00  00 00 00 00 00 00 00 00  |................|
00000430  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000440  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000450  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000460  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000470  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000480  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000490  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000004a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000004b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000004c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000004d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000004e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000004f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000500  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000510  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000520  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000530  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000540  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000550  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000560  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000570  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000580  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000590  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000005a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000005b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000005c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000005d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000005e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000005f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000600
student@student:~/Desktop/csc415$
```

2. A description of the VCB structure

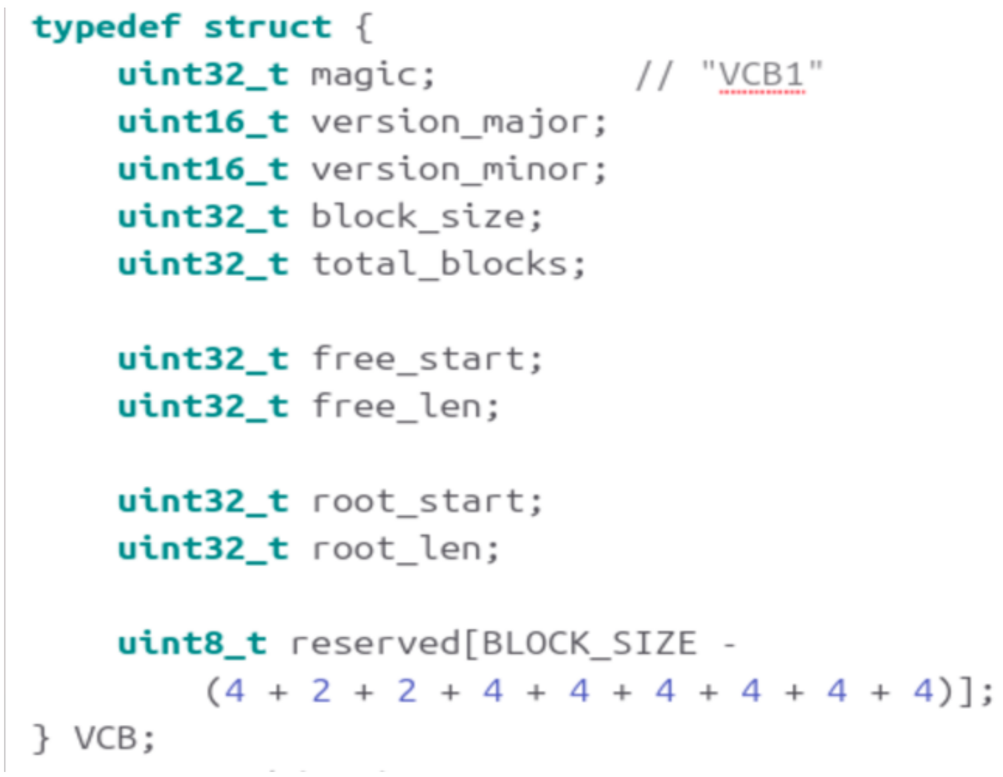
The Volume Control Block (VCB) is stored in block 0 of our volume and defines the overall layout and key metadata of the file system. It acts as the "table of contents" for the volume, allowing the operating system to locate and manage the other structures such as free space and the root directory.

Our VCB contains the following fields:

Field	Type	Description
magic	uint32_t	A unique signature ("VCB1") that identifies this block as a valid Volume Control Block.
version_major / version_minor	uint16_t each	Defines the version of the file system (currently version 1.0).
block_size	uint32_t	The size, in bytes, of each block in the volume (512 bytes).
total_blocks	uint32_t	The total number of blocks in the volume (4096).
free_start	uint32_t	The block number where the free-space bitmap begins (block 1).
free_len	uint32_t	The number of blocks used by the free-space structure (1 block).
root_start	uint32_t	The starting block of the root directory (block 2).
root_len	uint32_t	The number of blocks reserved for the root directory (1 block).
reserved	uint8_t[]	Unused space within the 512-byte block, reserved for future expansion.

Each field in the VCB is stored using little-endian byte order. Together, these values fully describe how the file system is organized on disk.

As a team, we confirmed that the VCB structure was correctly packed to occupy exactly one block (512 bytes) and that all fields align properly. We verified the written data through the HexDump output showing the "VCB1" magic value at the start of block 0.



```
typedef struct {
    uint32_t magic;          // "VCB1"
    uint16_t version_major;
    uint16_t version_minor;
    uint32_t block_size;
    uint32_t total_blocks;

    uint32_t free_start;
    uint32_t free_len;

    uint32_t root_start;
    uint32_t root_len;

    uint8_t reserved[BLOCK_SIZE -
        (4 + 2 + 2 + 4 + 4 + 4 + 4 + 4 + 4)];
} VCB;
```

3. A description of the Free Space structure

The Free Space structure is stored in block 1 of the volume and is implemented as a bitmap. Each bit in the bitmap represents one block in the volume. A bit value of 1 means the block is allocated, and a bit value of 0 means the block is free. In our implementation, we use a 512-byte block, giving us 4096 bits to track all 4096 blocks in the volume (1 bit per block). This allows the entire free-space map to fit neatly in a single block.

During initialization, our program marks the first three blocks as used:

- Block 0: Volume Control Block (VCB)
- Block 1: Free Space Map itself
- Block 2: Root Directory

All remaining bits are set to 0, meaning those blocks are available for future allocation.

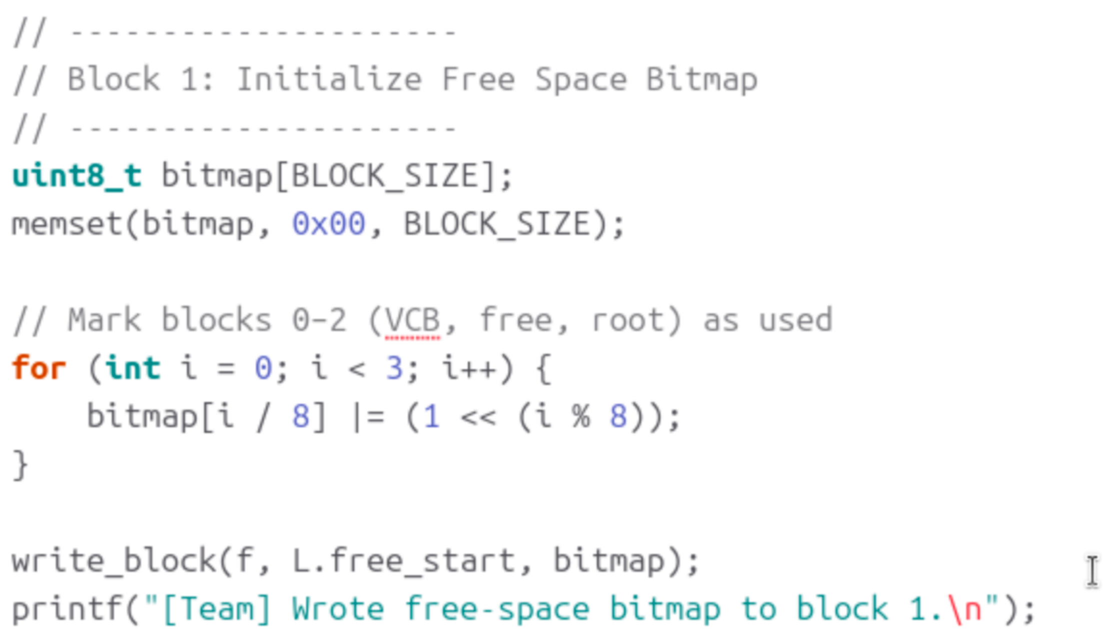
This bitmap is written to block 1 using a simple byte array where each bit position corresponds to a block index. For example:

```
for (int i = 0; i < 3; i++) {
   bitmap[i / 8] |= (1 << (i % 8));
}
```

This sets bits 0, 1, and 2 to 1, marking them as allocated.

The Free Space structure is initialized only once during formatting and the allocate() function will use this bitmap later to find the next available block.

We verified the bitmap by dumping block 1 using HexDump, where the first byte shows bits set for blocks 0–2, and the rest of the block appears as zeros. This confirms the correct setup of our free-space management system.



```c
// ---------------------
// Block 1: Initialize Free Space Bitmap
// ---------------------
uint8_t bitmap[BLOCK_SIZE];
memset(bitmap, 0x00, BLOCK_SIZE);

// Mark blocks 0-2 (VCB, free, root) as used
for (int i = 0; i < 3; i++) {
    bitmap[i / 8] |= (1 << (i % 8));
}

write_block(f, L.free_start, bitmap);
printf("[Team] Wrote free-space bitmap to block 1.\n");
```

4. A description of the Directory system

The Directory System is stored in block 2 of the volume and is responsible for managing the organization of files and folders within the file system.
During initialization, our program creates and writes the Root Directory, which serves as the starting point for all file operations.

In this milestone, the root directory is initialized with the mandatory entries:

- ”.” (dot): Refers to the current directory (itself).
- ”..” (dot dot): Refers to the parent directory (for the root, this also points to itself).

Each directory entry includes the name, block number, type, and valid flag. In this phase, only the "." entry is written, and the structure is set up so additional entries can be easily added in later milestones.

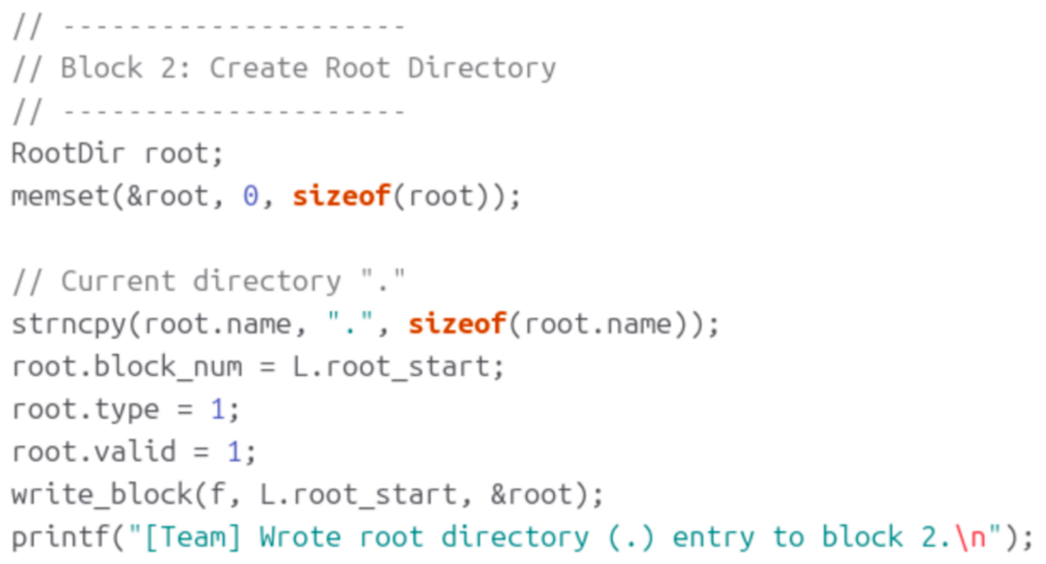
The directory structure uses a simple C struct that fits entirely within one 512-byte block. It is defined as:

```
typedef struct {
    char name[32];
    uint32_t block_num;
    uint8_t type;   // 1 = directory, 2 = file
    uint8_t valid;
    uint8_t padding[BLOCK_SIZE - 32 - 4 - 1 - 1];
} RootDir;
```

When the volume is formatted, our program writes this initialized root directory structure to block 2, marking it as valid and ready for use.

We verified this step using HexDump, where the ASCII representation clearly shows "." in the name field of block 2, confirming successful directory creation.

This directory initialization ensures that the file system has a working root from which future subdirectories and files can be created.



```
// ---------------------
// Block 2: Create Root Directory
// ---------------------
RootDir root;
memset(&root, 0, sizeof(root));

// Current directory "."
strncpy(root.name, ".", sizeof(root.name));
root.block_num = L.root_start;
root.type = 1;
root.valid = 1;
write_block(f, L.root_start, &root);
printf("[Team] Wrote root directory (.) entry to block 2.\n");
```

5. A table of who worked on which components.

Component	Team Member(s)	Description of Work
General Design, Coding, and Testing (50%)	All Members	The majority of the project, including brainstorming, debugging, and overall development, was completed together as a team. Everyone contributed equally to coding, verification, and HexDump testing.
Volume Control Block (VCB)	Prit Panchani & Viraj Hingu	Designed and implemented the VCB structure. Prit led the main logic and verification, and Viraj assisted with structure testing and output validation.
Free Space Management	Shiv Patel & Krushit Moradiya	Handled bitmap initialization and free-block allocation logic. Both worked on verifying that the free-space map was written correctly to block 1.
Root Directory System	Prit Panchani & Shiv Patel	Implemented the root directory structure and initialized entries for "." and "..". Verified block 2 data using HexDump.
Documentation and Report (PDF)	Viraj Hingu & Krushit Moradiya	Compiled screenshots, organized write-ups, and finalized the milestone report with team input and proof review.

6. How Our Team Worked Together

Team Carrot worked together really well during this milestone. We usually met in person about three times a week to talk about what we finished, fix bugs, and test our program together. Shiv, Prit, and Viraj live together in the same apartment, and Krushit lives close by, so it was easy for all of us to meet up and work on the project. Meeting face to face made things faster because we could test the code right away and check the HexDump results together.

When someone couldn't join in person, we used Zoom for short meetings to share updates and plan what to do next. We all kept in touch often so everyone knew what was going on.

At the beginning of the project, we divided the work based on what each person was best at. Prit and Viraj worked on building and testing the Volume Control Block (VCB). Shiv and Krushit worked on the Free Space Management part, setting up the bitmap and checking that it worked properly. Everyone helped with the Root Directory part and made sure the "." and ".." entries were written correctly.

We always talked before making big changes, helped test each other's code, and reviewed the results together to make sure everything looked right. Because we live close and worked together often, we stayed organized and were able to finish the milestone on time.

7. Issues Faced and How We Resolved Them

We ran into a few problems while working on this milestone. One problem was that our Volume Control Block (VCB) didn't fit into one block at first. We found out it was because of how the structure was lined up in memory. We fixed it by using #pragma pack(1) and checking the size until it was exactly 512 bytes. After that, it wrote to block 0 correctly, and the HexDump showed "VCB1" at the start.

Another issue was with the Free Space Bitmap. At first, the wrong blocks were marked as used, so the HexDump didn't look right. We tested it by printing values and checking each bit until we found the problem in the code. After fixing it, the first three blocks were shown as used, and the rest were free like we wanted.

We also had a small problem with the Root Directory. The "." and ".." entries were overlapping in the same block. We fixed this by changing how we wrote the structure so both entries fit without overwriting each other.

There were also some small issues we faced while testing. Sometimes the HexDump didn't refresh right after we changed the code, so we had to delete the old myvolume.dat file and make a new one. A few times, the program didn't close the file properly, which made it hard to rerun until we fixed the file permissions. We also had small mistakes like typos in variable names or missing semicolons that stopped the program from compiling.

Sometimes, two people worked on the same file at the same time, which caused small mix-ups in the code. We solved that by talking before making changes and telling the group what we were doing.

In the end, we fixed all our issues by testing often, communicating clearly, and helping each other whenever something broke. Once everything worked, the HexDump showed that the VCB, Free Space, and Root Directory were all written correctly.