

Group Name: Caffeinated Crew

Members: Issac Moreno, Katy Lam, Anisah Chowdhury, Malieka Sutaria

GitHub: crvelworld

CSC-415.03

“Caffeinated Crew”

File System Project: Milestone 1

Issac Moreno (921984788) - GitHub Master

Anisah Chowdhury (921677676)

Malieka Sutaria (922888691)

Katy Lam (921922518)

GitHub: [Link to: Central GitHub](#)

Group Name: Caffeinated Crew
Members: Issac Moreno, Katy Lam, Anisah Chowdhury, Malieka Sutaria
GitHub: crvelworld

Table of Contents

Dump Analysis:	3
VCB Structure:	3
Bitmap:	5
Root Directory:	11
VCB Structure Description:	12
Free Space Structure Description:	12
Directory Description:	13
Group Work Table:	14
Teamwork Efforts:	15
Issues and Resolutions :	16

Group Name: Caffeinated Crew

Members: Issac Moreno, Katy Lam, Anisah Chowdhury, Malieka Sutaria

GitHub: crvelworld

Dump Analysis:

Using the `Hexdump` utility, we were able to test our data structures to ensure that data was written to the `SampleVolume` correctly. Each value corresponds to exactly the way we implemented our structures and assigned each value. The locations of some members span 32 bytes (1 full memory address) because we used `uint64_t` as the type for each relevant member. The size of this type is 8 bytes, which corresponds to $8 \times 4 = 32$. Each screen shot shows this.

VCB Structure:

At 0x00020-0x0002DF, we can see that the name field of our VCB spans to 0x0002E0 in ASCII form. This translates to “Caffeinated Crew FS”, which is exactly what we named our volume. The rest of this space was left as empty characters, because we used the same macro for a directory name, which has a maximum size of 225 characters.

```
Operating System v524 - VMware Workstation
File Edit View VM Tabs Help || □ ×
Home Operavting System v524
Activities
0019C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
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: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....
student@student:/Desktop/Work/csc415-filesystem-crvelworld$ Hexdump/hexdump.linux SampleVolume --start 1 --count 13
Dumping file SampleVolume, starting at block 1 for 13 blocks:

000200: 43 61 66 66 65 69 6E 61 74 65 64 20 43 72 65 77 | Caffeinated Crew
000210: 20 46 53 00 00 00 00 00 00 00 00 00 00 00 00 00 | FS.....
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 | .....
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 4B 4C 90 00 00 00 00 00 | ....KL....
000300: 4A 4C 90 00 00 00 00 00 70 F9 4E F9 AF 00 00 1JL | ....pLEN.
```

At 0x0002E0-0x0002E8, this section is left as padding for the next member due to our VCB implementation. (May get changed later)

At 0x0002EA-0x0002EF, we can see that the VCB member of blockSize is stored. The value of blockSize is 0x0...020, which directly translates to 512. We know this value is correct because we assigned blockSize to our macro BLOCK_SIZE, which is 512 by default.

Group Name: Caffeinated Crew

Members: Issac Moreno, Katy Lam, Anisah Chowdhury, Malieka Sutaria

GitHub: crvelworld

000300:	4A 4C 00 00 00 00 00 00 00	freeBlocks	70 89 B8 F1 44 B1 00 00	freeSpaceManager address	JL.....p000D0...
000310:	00 0A 00 00 00 00 00 00 00	bitmapSize (in bytes)	01 00 00 00 00 00 00 00 00	FSM start
000320:	05 00 00 00 00 00 00 00 00	FSM end	18 30 95 32 F3 83 4F 32	signature020202
000330:	00 00 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 00 00	
000350:	00 00 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 00 00	
000370:	00 00 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 00 00	
000390:	00 00 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 00 00	
0003B0:	00 00 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 00 00	
0003D0:	00 00 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 00 00	
0003F0:	00 00 00 00 00 00 00 00 00		00 00 00 00 00 00 00 00 00	

At 0x0002F0-0x0002F7, this is the value of the rootDirectoryLocation in disk. Here, the location of the directory entry array is held.

At 0x0002F8-0x0002FF, we can see that the value of our totalBlocks member is stored here. The value of 0x0...004C4B converts to 19,531. This value is correct because it is the default value passed to initFileSystem during runtime. Using different run options changes this value accordingly.

At 0x000300-0x000307, we can see that here is the value of freeBlocks. The value in the dump is 0x0...04A4C. This hex number converts to 19530, which is what is expected after allocating space for freeBlocks.

At 0x000308-0x00030F, is the address of the freeSpaceManager when it is pulled into memory. This address is pulled in during runtime and kept there.

At 0x000310-0x00031F, this value is the size of our bitmap (in bytes). The value in our hexdump is 0x0...0A00, which converts to 2560. We know this is correct because there on initializing, there should be blocks * BLOCK_SIZE bytes to hold all the bits. Therefore, our testing was 5 so it is correct.

At 0x000320-0x000327 is the value of our FSM starting index. Here, the VCB keeps track of where the next free space index is. At our time of testing, it was 1. We know this is correct because VCB is at 0 and FSM is at 1.

Group Name: Caffeinated Crew

Members: Issac Moreno, Katy Lam, Anisah Chowdhury, Malieka Sutaria

GitHub: crvelworld

At 0x000328-0x00032F at the time of testing, this value was the ending index of the FSM bitmap. This value corresponds to the 5th block, which is correct because the bitmap took up blocks 1-5 in the following analysis.

At 0x000330-0x00033F, Padding

Bitmap:

Address analysis: 0x000400-0x000E00

Our bitmap during the default run holds 2560 bytes to track the free space in our sample volume. The difference between these memory addresses is exactly 2560. Because these values match, we can ensure that our bitmap extends 5 blocks (as planned) to keep track of free space. Further, the value 0x3F at address 0x000400 corresponds to the binary number 0b00111111. In terms of the bitmap, this corresponds to the first 6 blocks being used. These numbers match expected output because the first 6 blocks are reserved for the VCB and bitmap (5 blocks long).

000400:	3F	00 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 00
000420:	00	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 00
000440:	00	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 00
000460:	00	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 00
000480:	00	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 00
0004A0:	00	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 00
0004C0:	00	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 00
0004E0:	00	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 00

Group Name: Caffeinated Crew

Members: Issac Moreno, Katy Lam, Anisah Chowdhury, Malieka Sutaria

GitHub: crvelworld

Group Name: Caffeinated Crew

Members: Issac Moreno, Katy Lam, Anisah Chowdhury, Malieka Sutaria

GitHub: crvelworld

Group Name: Caffeinated Crew

Members: Issac Moreno, Katy Lam, Anisah Chowdhury, Malieka Sutaria

GitHub: crvelworld

Group Name: Caffeinated Crew

Members: Issac Moreno, Katy Lam, Anisah Chowdhury, Malieka Sutaria

GitHub: crvelworld

Group Name: Caffeinated Crew

Members: Issac Moreno, Katy Lam, Anisah Chowdhury, Malieka Sutaria

GitHub: crvelworld

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

Group Name: Caffeinated Crew
Members: Issac Moreno, Katy Lam, Anisah Chowdhury, Malieka Sutaria
GitHub: crvelworld

Root Directory:

VMware Fusion File Edit View Virtual Machine Window Help

OperatingSystemsSP24ARM

Activities Terminal Apr 9 00:19

```
student@student: ~/Downloads/csc415-filesystem-crvelworld
```

000E00: 00 FF FF FF 00 00 00 00 00 00 00 00 C0 00 00 00
000FF0: 00 00 00 00 C0 00 00 00 00 00 00 00 00 00 00 00
001000: 2E 2E 00 FF FB 61 12 42 30 A3 9E C3 FF FF 00 00	...0.a.B0.....
001010: 98 1B B3 88 35 BF 00 00 30 A3 9E C3 FF FF 00 00	0..050..0.....
001020: 30 A3 9E C3 FF FF 00 00 F0 A2 9E C3 FF FF 00 00	0.....0.....
001030: C8 FF FF 80 FF FF 50 A3 9E C3 FF FF 00 00	0000000000000000
001040: 30 A3 9E C3 FF FF 00 00 30 A3 9E C3 FF FF 00 00	0.....0.....
001050: F0 A2 9E C3 FF FF 00 00 C8 FF FF 80 FF FF FF	0000000000000000
001060: 00 E7 2C FF FB 61 12 42 25 25 25 25 25 25 25 25	0..0.a.B9.....
001070: 25 25 25 25 25 25 25 64 00 00 00 00 00 00 00 00	0000000000000000
001080: 53 74 61 72 74 20 50 61 OF F0 OF F0 OF F0 OF F0	Start Pa.0.0.0.
001090: OF F0 OF F0 OF F0 OF F0 00 00 OF F0 OF F0 OF F0 OF F0	0.0.0.0.0.0.0.0.
0010A0: 00 00 00 00 00 00 00 00 OF F0 OF F0 OF F0 OF F00.0.0.
0010B0: OF F0 OF F0 OF F0 OF F0 00 FF FF FF 00 00 00 00	0.0.0.0.0.0.0.0.
0010C0: 00 FF FF FF 00 00 00 00 00 00 00 00 00 C0 00 00 00	0000000000000000
0010D0: 00 00 00 00 C0 00 00 00 00 00 00 00 00 00 00 00 00	0.....0.
0010E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 000.
0010F0: 81 EB 14 66 00 00 00 06 00 00 00 00 00 00 00 00	00.E.....
001100: 01 00 00 00 00 00 00 00 30 31 2F 30 31 2F 30 3001/01/00
001110: 20 30 30 3A 30 30 3A 30 30 00 00 00 00 00 00 00	00:00:00.....
001120: 00 E7 2C FF FB 61 12 42 50 A3 9E C3 FF FF 00 00	0..0.a.BP.....
001130: 48 1B B3 88 35 BF 00 00 00 02 00 00 00 00 00 00 00	H.050.....
001140: 4B 4C 00 00 00 00 00 00 B0 A3 9E C3 FF FF 00 00	KL.....0.....
001150: FC 73 BD 7E FD F5 00 00 28 A5 9E C3 FF FF 00 00	0..0..0..0.....
001160: CC 51 E9 7E 04 00 00 28 A5 9E C3 FF 00 00 00 00	0..0.....00000000
001170: 00 96 98 00 00 00 00 00 00 02 00 00 00 00 00 00 00	0000000000000000
001180: 11 B4 9E C3 FF FF 00 00 B0 A3 9E C3 FF FF 00 00	0000000000000000
001190: C0 73 BD 7E FD F5 00 00 28 A5 9E C3 FF FF 00 00	0..0..0..0.....
0011A0: 00 E7 2C FF FB 61 12 42 C0 A4 9E C3 FF FF 00 00	0..0.a.B0.....
0011B0: CC 74 79 7E FD F5 00 00 74 77 E9 7E FD F5 00 00	0..0..0..t..0..0..
0011C0: 54 1A B3 88 35 BF 00 00 50 A5 9E C3 04 00 00 00	T.050..P00000000
0011D0: 28 A5 9E C3 FF FF 00 00 28 A5 9E C3 FF FF 00 00	0000000000000000
0011E0: 04 00 00 00 00 00 00 00 78 4C B4 88 35 BF 00 00x100000000000
0011F0: 40 F0 EB 7E FD F5 00 00 54 1A B3 88 35 BF 00 00	00000.T.050.....

Group Name: Caffeinated Crew
Members: Issac Moreno, Katy Lam, Anisah Chowdhury, Malieka Sutaria
GitHub: crvelworld

0x000E00-0x0011FF is our rootDirectory array. This array holds the file handles contiguously. The locations of each file can be stored elsewhere on disk, but will have their location mapped.

VCB Structure Description:

Operating systems use the Volume Control Block (VCB) structure to handle important data related to a given volume. This includes information about the volume's name, unique identifier, size, file system type, mount status, mount point directory, availability of free space, details about allocation tables, access control information, and file system metadata such as the location of the root directory. As it serves an important part in managing and organizing volumes inside a system, this structure is essential for the effective management and access of data on storage devices.

1. Structure Definition:

```
typedef struct VCB {  
    char name[MAX_NAME_LENGTH + 1]; // Name of volume  
    uint64_t blockSize;           // Max block size (standard)  
    uint64_t rootDirectoryLocation; // Location of the root directory  
    uint64_t totalBlocks;         // Total blocks used by the program  
    uint64_t freeBlocks;          // Total number of free blocks  
    FSM freeSpaceManager;        // Pointer to the free space bitmap  
                                // & also contains start and end blocks  
    uint64_t FSMstart;           // First free block  
    uint64_t FSMend;             // End block  
    uint64_t signature;          // Initialized flag  
} VCB;
```

Free Space Structure Description:

We used bitmap as a way to store our free space and use it to manage the storage and resources efficiently.

Steps taken:

- Allocate memory for bitmap by the total number of blocks
 - Check if current bit is in use or not
 - Initialize variable to index of free space

Group Name: Caffeinated Crew

Members: Issac Moreno, Katy Lam, Anisah Chowdhury, Malieka Sutaria

GitHub: crvelworld

- Store information in the VCB (volume control block)
- Free memory after it is not in use

Information stored in this way is stored into each block where each block in the memory or in the disk will contain information of the block next to it in the LBA and the block after that in the file or if there is any free space system. Doing it this way also holds the data knowing where it begins and ends and the total amount of free blocks that are on the disk and in the memory.

The functions from the freespace system will be working together to make it easy for us to use the storage space without worrying how it is being organized at the lower levels. This way, it will hide the process and how data is being stored into each of the individual blocks, letting it do whatever it needs and letting us deal with the blocks of data afterwards and making it less complex for us to work with.

In addition, we went ahead and consolidated two functions (bitCheck & allocateSpace) to streamline the process. Originally we wanted to keep them separate, but bitCheck will only be used in allocateSpace. So, we added the arguments from bitCheck into allocateSpace.

Directory Description:

The directory structure in our file system is represented by the dirEnt structure. Each dirEnt entry contains information about a file or directory, including its name, size, date it has been modified, location, and if it is a directory or a file. We also have a field that stores the date in a readable format. The directory entries are stored into an array, and then saved into the disk.

With each directory entry containing:

- Name: The name of the file or directory, this is stored as a string with a maximum length of MAX_NAME_LENGTH.
- Size: The size of the file or Directory in bytes.
- Date: The last modification date of the file or directory, which is stored as a time_t value and well as when it was last accessed.
- Location: The location of the file or directory within the file system itself.
- isDir: A flag indicating whether the entry represents a directory(1) or file (0).
- Dates: The date is in readable format
 - In example 04/07/2024 12:40:30

The root directory is initialized with at least two entries “.” and “..”. These entries represent the current directory and the parent directory, respectively. They are an important part of the file system structure navigation.

Group Name: Caffeinated Crew

Members: Issac Moreno, Katy Lam, Anisah Chowdhury, Malieka Sutaria

GitHub: crvelworld

Function createDirectory

This function creates a new directory entry in the file system. It checks if there is enough free space to create a directory, then creates a new directory entry with the name, size, date, location, and type. It updates the parent directory entry to show the new changes of the new directory. It also updates the free block count in the Volume Control Block.

Parameters for createDirectory

- Name: the name of the new directory
- vcb: Pointer to the Volume Control Block
- parentDirLocation: parent directory in the file system's location

newDirEntry creates a new directory entry with a specified name, size, date and time, and isDir flag is set to 1, indicating it is a directory.

The write directory entry newDirEntry writes the new entry to the disk in the specified location.

Update Parent Directory , reads the parent directory entry from the disk at the parentDirLocation, writes the updated entry back to the disk.

Update Free Blocks, decrements the freeBlock count in the VCB to reflect the allocation of a new block for the directory and writes the updated VCB to disk.

Group Work Table:

Names:	What we did:
Issac Moreno	<ul style="list-style-type: none">- initVCB function- Free space initialization- Dump analysis- Write up- Planned meet up times- Code clean up/organization- GitHub management
Katy Lam	<ul style="list-style-type: none">- Worked on the freespace and plan what should be inside and structuring of the freespace function

Group Name: Caffeinated Crew

Members: Issac Moreno, Katy Lam, Anisah Chowdhury, Malieka Sutaria

GitHub: crvelworld

	<ul style="list-style-type: none">- Worked on organizing the docs partially- Worked on the free space structure description- Worked on reviewing and double checking if the directory and the dump is correct and make sure we had everything- Finding the reasons behind some of the errors- Written the parent directory entry
Anisah Chowdhury	<ul style="list-style-type: none">- Free space<ul style="list-style-type: none">- Space allocation, checking if there's free space for a certain number of blocks.- Header description- Debugged with teammates on space allocation- Write up<ul style="list-style-type: none">- Teamwork efforts- Added onto freespace description
Malieka Sutaria	<ul style="list-style-type: none">- VCB structure<ul style="list-style-type: none">- Created struct of VCB, with the appropriate descriptions- Created dirEnt structure- Initialization<ul style="list-style-type: none">- Free space bitmap- VCB- Root Directory- Write up<ul style="list-style-type: none">- VCB Structure description<ul style="list-style-type: none">- Structure Definitions- Directory Description

Summary:

Teamwork Efforts:

Our team worked together by meeting twice (once after class and once over zoom) throughout this milestone. We divided up the tasks by creating separate two-person

Group Name: Caffeinated Crew

Members: Issac Moreno, Katy Lam, Anisah Chowdhury, Malieka Sutaria

GitHub: crvelworld

teams. Issac and Malieka's main objective was the VCB Structure, while Anisah and Katy's main objective was the Free Space allocation. Whichever team finished their objective, moved onto the Root Directory. Each sub-team was also responsible for their own description of the respective task. We all took part in contributing to the issues and resolution as we faced them. To keep up to date, we communicated in a group chat via text message and discord.

Issues and Resolution

Issues with Free space:

- Freeing what needed to be freed, still had some memory leaks
As we went along with our code, we were closely checking on the memory management step by step to ensure that memory was being managed correctly and check in places that possibly might need to be freed.

Had some trouble figuring out how to allocate properly, could not decide if it was best to allocate first based on the total amount of blocks or do it by the amount of free space available in the blocks to allocate the memory for.

Resolution:

We figured out the best solution to the allocation was to check the block itself to see if it contained any bits before so that we can find the location of the freespace in the beginning.

Another error we encountered:

The possible reasons for this error are that the free space is not properly taking in the data or the data is not properly being written into the block.

Group Name: Caffeinated Crew

Members: Issac Moreno, Katy Lam, Anisah Chowdhury, Malieka Sutaria

GitHub: crvelworld

```
student@student:~/csc415-filesystem-crvelworld$ Hexdump/hexdump.linuxM1 SampleVolume --start 1 --count 1
Dumping file SampleVolume, starting at block 1 for 1 block:

000200: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....I.....
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 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 | .....I.....
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 |
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 | .....
```

Resolution:

A small bug we had was that we never used the LBAWrite, therefore it was not writing into the block like it was supposed to.