File System Milestone One

Team:
Innov8

Names:
Justin Ho 922266680
Junyoung Kim 920303420
Ryan Flannery 921824732
Bilguun Bayasgalan 922564900

Central Repository:
csc415-filesystem-justinh128

Team:
Innov8

Names:
Justin Ho 922266680
Junyoung Kim 920303420
Ryan Flannery 921824732
Bilguun Bayasgalan 922564900

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

```
student@student:~/csc415-filesystem-justinh128$ Hexdump/hexdump.linux SampleVolume --start 1 --count 1
Dumping file SampleVolume, starting at block 1 for 1 block:

000200: 00 96 98 00 00 00 00 00  00 02 00 00 00 00 00 00 | .◆◆.............
000210: 4B 4C 00 00 00 00 00 00  4A 4C 00 00 00 00 00 00 | KL......JL......
000220: 06 00 00 00 01 00 00 00  05 00 00 00 00 00 00 00 | ................
000230: 40 56 24 67 00 00 00 00  40 56 24 67 00 00 00 00 | @V$g....@V$g....
000240: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000250: CD AB 34 12 00 00 00 00  00 00 00 00 00 00 00 00 | 4.............
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  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 | ................
```

Team:
Innov8

Names:
Justin Ho 922266680
Junyoung Kim 920303420
Ryan Flannery 921824732
Bilguun Bayasgalan 922564900

VCB is located at Block 1:

- volume_size (8 bytes):
    - 00 96 98 00 00 00 00 00
    - Little-endian: 00 00 00 00 98 96 00 00
    - Address: 0x000200 to 0x000207
    - Offset: 512 bytes
- block_size (4 bytes):
    - 00 02 00 00
    - Little-endian: 00 00 02 00
    - Address: 0x000208 to 0x00020B
    - Offset: 520 bytes
- total_blocks (8 bytes):
    - 00 00 00 00 00 00 00 00
    - Little-endian: 00 00 00 00 00 00 00 00
    - Address: 0x00020C to 0x000213
    - Offset: 524 bytes
- free_blocks (8 bytes):
    - 4B 4C 00 00 00 00 00 00
    - Little-endian: 00 00 00 00 00 00 4C 4B
    - Address: 0x000214 to 0x00021B

Team:
Innov8

Names:
Justin Ho 922266680
Junyoung Kim 920303420
Ryan Flannery 921824732
Bilguun Bayasgalan 922564900

    - Offset: 532 bytes
- root_directory (4 bytes):
    - 4A 4C 00 00
    - Little-endian: 00 00 4C 4A
    - Address: 0x00021C to 0x00021F
    - Offset: 540 bytes
- bitmap_start (4 bytes):
    - 06 00 00 00
    - Little-endian: 00 00 00 06
    - Address: 0x000220 to 0x000223
    - Offset: 544 bytes
- bitmap_size (4 bytes):
    - 01 00 00 00
    - Little-endian: 00 00 00 01
    - Address: 0x000224 to 0x000227
    - Offset: 548 bytes
- creation_time (8 bytes):
    - 40 56 24 67 00 00 00 00
    - Little-endian: 00 00 00 00 67 24 56 40
    - Address: 0x000228 to 0x00022F

Team:
Innov8

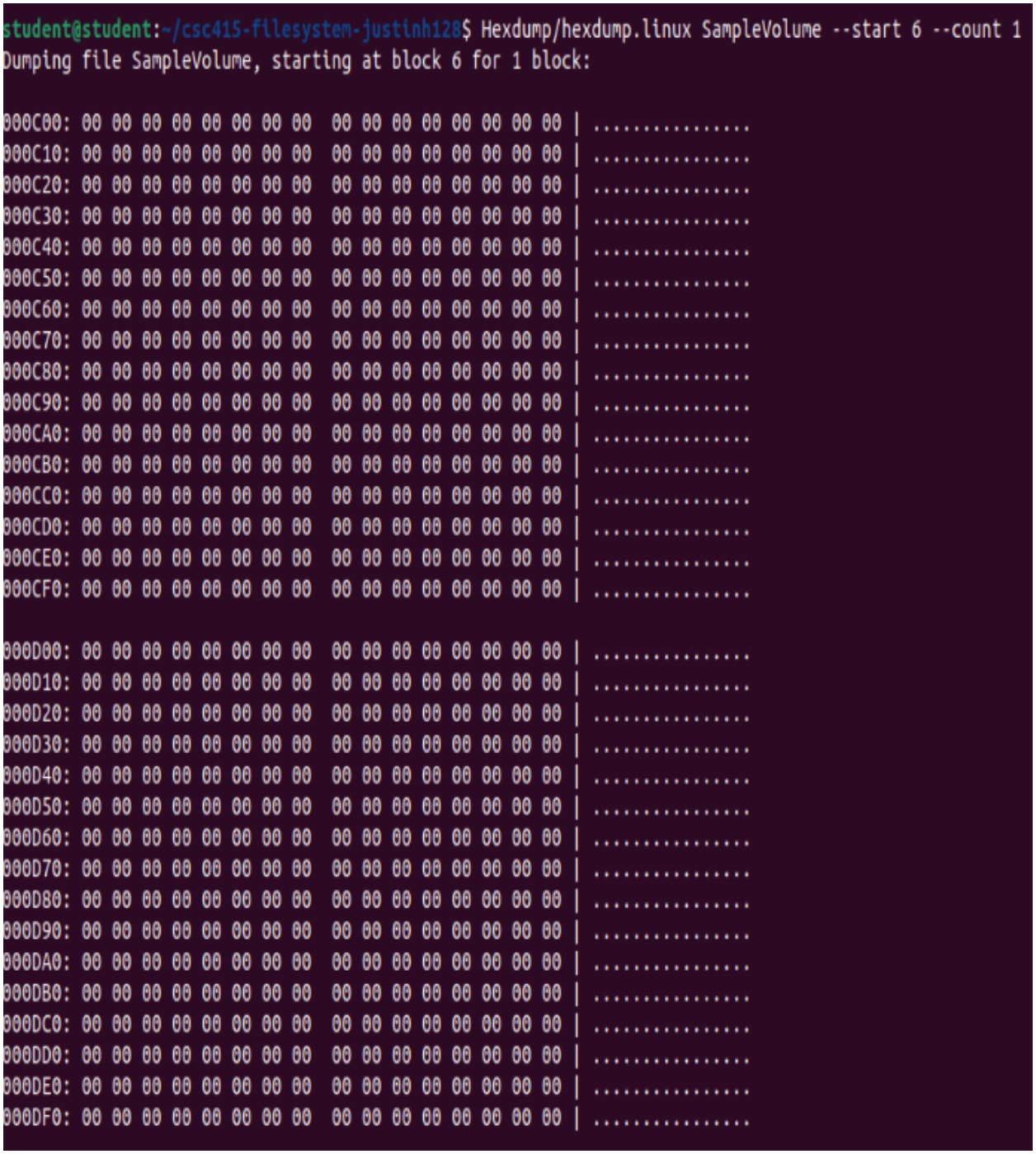
Names:
Justin Ho 922266680
Junyoung Kim 920303420
Ryan Flannery 921824732
Bilguun Bayasgalan 922564900

    - Offset: 552 bytes

- modification_time (8 bytes):
    - 40 56 24 67 00 00 00 00
    - Little-endian: 00 00 00 00 67 24 56 40
    - Address: 0x000230 to 0x000237
    - Offset: 560 bytes
- volume_id (16 bytes):
    - 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    - Little-endian: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    - Address: 0x000238 to 0x000247
    - Offset: 568 bytes
- signature (4 bytes):
    - CD AB 34 12
    - Little-endian: 12 34 AB CD (matches 0x1234ABCD signature)
    - Address: 0x000248 to 0x00024B
    - Offset: 584 bytes

Team:
Innov8

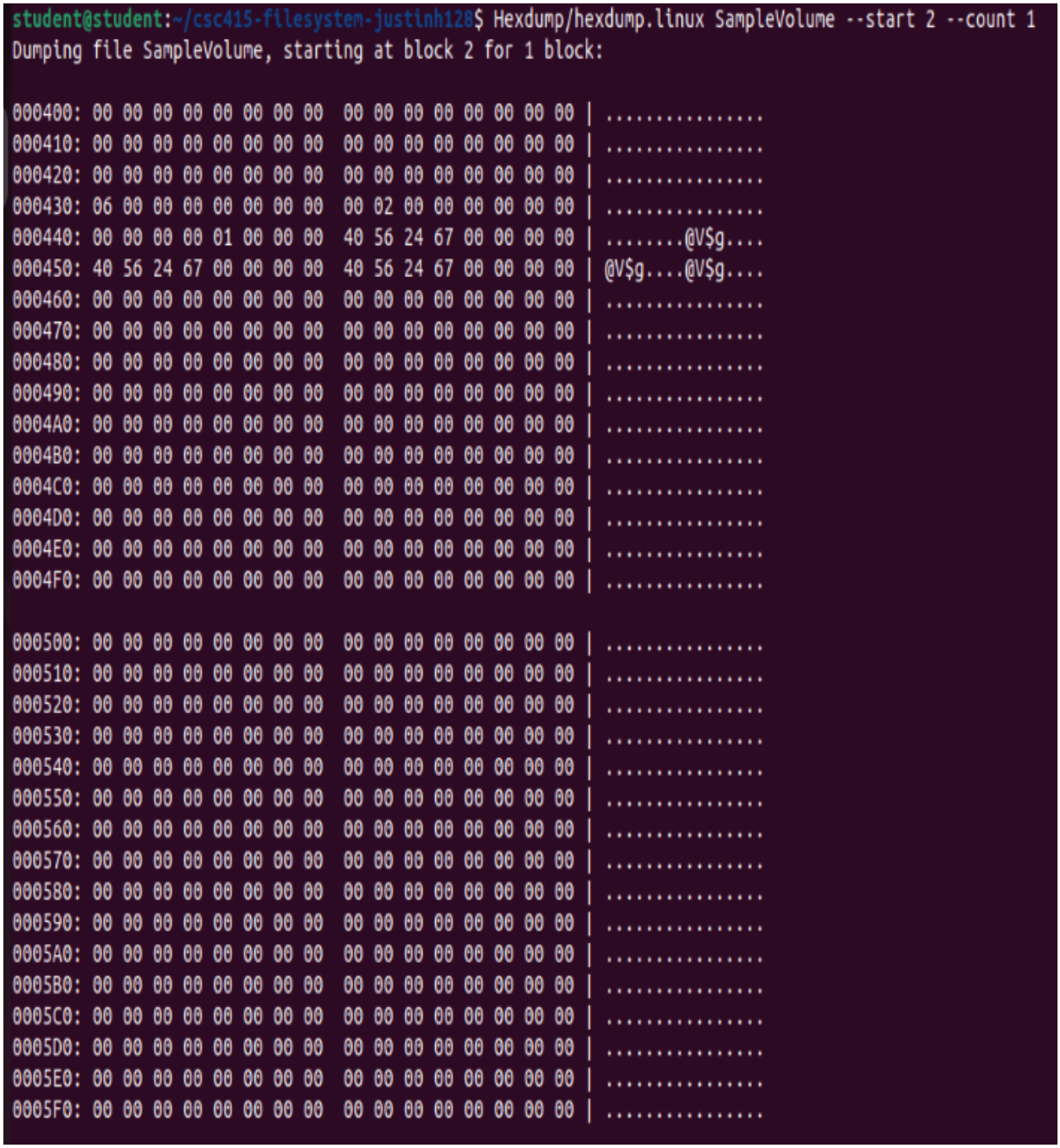
Names:
Justin Ho 922266680
Junyoung Kim 920303420
Ryan Flannery 921824732
Bilguun Bayasgalan 922564900



Team:
Innov8

Names:
Justin Ho 922266680
Junyoung Kim 920303420
Ryan Flannery 921824732
Bilguun Bayasgalan 922564900

Free Space is located at Block 6 (above)



Team:
Innov8

Names:
Justin Ho 922266680
Junyoung Kim 920303420
Ryan Flannery 921824732
Bilguun Bayasgalan 922564900

Root Directory starts at Block 2:

“.” Entry:

- name (32 bytes):
    - 06 00 00 00 00 00 00 00 00 02 00 00…
    - Address: 0x000430 to 0x00044F
    - Offset: 1072 bytes
- location (4 bytes):
    - 40 56 24 67
    - Little-endian: 67 24 56 40
    - Address: 0x000450 to 0x000453
    - Offset: 1104 bytes
- size (8 bytes):
    - 00 00 00 00 00 00 00 00
    - Little-endian: 00 00 00 00 00 00 00 00
    - Address: 0x000454 to 0x00045B
    - Offset: 1108 bytes
- isDir (1 byte):
    - 00
    - Little-endian: 00
    - Address: 0x00045C

Team:
Innov8

Names:
Justin Ho 922266680
Junyoung Kim 920303420
Ryan Flannery 921824732
Bilguun Bayasgalan 922564900

    - Offset:1116 bytes

- creation_time (8 bytes):
    - 40 56 24 67 00 00 00 00
    - Little-endian: 00 00 00 00 67 24 56 40
    - Address: 0x000460 to 0x000467
    - Offset: 1120 bytes
- modification_time (8 bytes):
    - 40 56 24 67 00 00 00 00
    - Little-endian: 00 00 00 00 67 24 56 40
    - Address: 0x000468 to 0x00046F
    - Offset: 1128 bytes
- access_time (8 bytes):
    - 40 56 24 67 00 00 00 00
    - Little-endian: 00 00 00 00 67 24 56 40
    - Address: 0x000470 to 0x000477
    - Offset: 1136 bytes

“..” Entry:

- name (32 bytes):
    - 00 00 00 00 01 00 00 00 …
    - Address: 0x000490 to 0x0004AF

Team:
Innov8

Names:
Justin Ho 922266680
Junyoung Kim 920303420
Ryan Flannery 921824732
Bilguun Bayasgalan 922564900

    - Offset: 1168 bytes
- location (4 bytes):
    - 40 56 24 67
    - Little-endian: 67 24 56 40
    - Address: 0x0004B0 to 0x0004B3
    - Offset: 1200 bytes
- size (8 bytes):
    - 00 00 00 00 00 00 00 00
    - Little-endian: 00 00 00 00 00 00 00 00
    - Address: 0x0004B4 to 0x0004BB
    - Offset: 1204 bytes
- isDir (1 byte):
    - 00
    - Little-endian: 00
    - Address: 0x0004BC
    - Offset:1212 bytes
- creation_time (8 bytes):
    - 40 56 24 67 00 00 00 00
    - Little-endian: 00 00 00 00 67 24 56 40
    - Address: 0x0004C0 to 0x0004C7

Team:
Innov8

Names:
Justin Ho 922266680
Junyoung Kim 920303420
Ryan Flannery 921824732
Bilguun Bayasgalan 922564900

- Offset: 1216 bytes

2. A description of the VCB structure

The VCB basically serves as the control center of the file system that provides essential information about the layout and status of the storage volume. It stores metadata that defines the structure of the volume, which makes it possible for the system to manage files, directories, and available space effectively. It includes fields such as volume_size, which indicates the total storage capacity of the volume, and block_size, which specifies the size of each individual block on the disk. It also keeps track of the total count of blocks and the count of currently free blocks, which tells the system where new data can be stored. The VCB uses a bitmap to track which blocks are free or occupied, starting at the bitmap_start block and spanning bitmap_size bytes. It designates the root_directory block, marking where the file system hierarchy starts. Timestamps are included to record when the volume was created and last modified. The signature field ensures the integrity of the VCB and validates that the volume is correctly formatted.

3. A description of the Free Space structure

The Free Space structure handles the allocation and keeping track of free blocks in the file system using the bitmap approach. Each bit of a bitmap represents either the status of the block specified by its bit number a 0 for free, a 1 for in use. First, we must allocate memory for the free

Team:
Innov8

Names:
Justin Ho 922266680
Junyoung Kim 920303420
Ryan Flannery 921824732
Bilguun Bayasgalan 922564900

space bitmap. The bitmap would require 2442 bytes to manage 19,531 blocks of size 512 bytes. But it is allocated on a block basis of 512 plus additional blocks for the metadata, resulting in a total of 5 blocks allocated for it. After memory was allocated, the entire bitmap was filled with zeros using memset() and thus marked all blocks as free. This way, the file system began with all blocks available for use. Next, the first 6 blocks were reserved for system use, Block 1 for the Volume Control Block (VCB) and block 6 for the free space bitmap itself. These blocks were marked as used by setting the first 6 bits of the bitmap to 1, determined by their predetermined byte and bit positions and the bitwise OR operation. Then, the bitmap, once initialized, was written to disk using the LBAwrite() function. Blocks numbered 1 through 6 were written onto the disk, containing the free space map for persistent storage. Lastly, a check was performed to make sure the correct number of blocks had indeed been written on the disk after writing the bitmap to it. If writing failed, an error message popped up, and the program exited to prevent further errors from developing. Ultimately, the VCB contained file system information, so it was altered to specify the particular beginning of the free space bitmap. The starting block for the free space was then determined according to the update and written over into block 6 to keep the system running without any inconsistency. The allocations for the blocks in memory initialization and actual writing into the disk.

4. A description of the Directory system

Team:
Innov8

Names:
Justin Ho 922266680
Junyoung Kim 920303420
Ryan Flannery 921824732
Bilguun Bayasgalan 922564900

The directory system in our project is designed to manage and organize files and directories on the storage volume. It establishes structures and functions that allow for effective file tracking and navigation. The directory system begins with the root directory, which is located at block 1. It serves as the starting point for the entire file hierarchy. The root directory includes entries for ".", the root itself, and "..", the parent directory. Each entry in the directory system is defined by the dir_entry structure. This information includes the name of the file or directory, the block address where the data is stored, the size of the file, permissions, a flag to distinguish if an entry is a directory, and timestamps. RootDirecInit initializes the root directory, setting up "." and ".." entries. writeRootToVol saves the root directory to disk at block 2. These are the basic foundations of the directory system as of right now. Next, we will have to expand functionality to support actions like creating, reading, and deleting files and directories.

5. A table of who worked on which components

Justin Ho	Junyoung Kim	Ryan Flannery	Bilguun Bayasgalan
vcb.h, fsInit.c, root.c, free_space.c	vcb.h, fsInit.c, root.c, free_space.c	vcb.h, fsInit.c, root.c, free_space.c	vcb.h, fsInit.c, root.c, free_space.c
Debugging and checking each others' work	Debugging and checking each others' work	Debugging and checking each others' work	Debugging and checking each others' work
Report: 1, 2, 5, 6, 7	Report: 4, 5, 6, 7	Report: 3, 5, 6, 7	Report: 4, 5, 6, 7

Team:
Innov8

Names:
Justin Ho 922266680
Junyoung Kim 920303420
Ryan Flannery 921824732
Bilguun Bayasgalan 922564900

6. How did your team work together, how often you met, how did you meet, how did you divide up the tasks.

Our team worked together through after class discussions, a text group chat, and a discord server. We met at least twice a week and more when it was needed. In our first meetings, we got to know each other to figure out strengths and weaknesses. This helped us delegate tasks and know where additional help would be needed. For this milestone, not one person was left alone to do anything by themselves. We had main members responsible for certain tasks, but we all collaborated on each aspect especially with debugging and checking other's work. When we would encounter issues, we would communicate to each other what the issue was, what we tried to fix, and possible solutions.

7. A discussion of what issues you faced and how your team resolved them.

The first issue we faced was not knowing where to start. We knew everyone had to be on the same page for this project, so we went over communicating pushes, pulls, changes, and problems. We also spent time on a call looking through the various aspects of the project including files and Canvas. Thankfully, we were provided with the steps for milestone one document. This helped us figure out where to start.

An issue we faced was with root.c and free_space.c including the structs within the same .c file. We knew that we needed to be able to call these functions across files and pass information. So,

Team:
Innov8

Names:
Justin Ho 922266680
Junyoung Kim 920303420
Ryan Flannery 921824732
Bilguun Bayasgalan 922564900

what we did was separate the structs into separate header files. This included vcb.h, root.h, and free_space.h. Within these files we had the designated structs as well as function prototypes to ensure there were no linkage errors.

An issue we faced was initializing the free_space, the VCB, and the root directory within their own .c files. We saw that there could be issues with this, so we looked through the files and decided to initialize everything in fsInit.c. Doing it this way would make sure that every aspect initialized when the file system initialized.

While initializing the free space bitmap in free_space.c, we faced a recurring issue where the memory allocation sometimes went out of bounds due to improper allocation sizes and handling. The bitmap initially required 2442 bytes to manage all blocks, but the additional blocks for metadata were causing the total allocation to round up, occasionally leading to insufficient memory. We resolved this by carefully calculating the total memory needed, including extra blocks for metadata, and allocating memory accordingly. We implemented boundary checks to ensure no allocation exceeded defined limits, such as the set block size, and used memset() to zero out the allocated memory. Adding debug statements also allowed us to track memory locations and identify areas where overflow could occur, which we corrected by refining the allocation code and using constants for clarity.

Team:
Innov8

Names:
Justin Ho 922266680
Junyoung Kim 920303420
Ryan Flannery 921824732
Bilguun Bayasgalan 922564900

Another issue we encountered was correctly interpreting the hexdump for different structures, such as the VCB. Given that each block contains data in hexadecimal format, understanding the structure and pinpointing each component within the dump was challenging. Also, we had to make sure each field in the VCB aligned with its correct address offset. We attempted to resolve this issue by systematically reviewing the structures and confirming each field's size and offset within the block. We also looked back on Assignment 2 for help deciphering the hexdump.