The Baha Blast

CSC415-02 Operating Systems

Milestone 1 Writeup

Group Members:
Diego Flores - 920372463
Mohammad Dahbour - 921246050
Kemi Adebisi - 921140633

Github: DiegoF001

Hexdumps Analysis

VCB:

```
000200: 00 00 00 00 61 68 61 42  06 00 00 00 00 02 00 00 | ....ahaB........
000210: 13 4C 00 00 4B 4C 00 00  05 00 00 00 00 00 00 00 | .L..KL..........
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 | ................
```

From 000200 to 000207: 8 bytes is the long for the magic number. Value: 0x4261686100000000 matches the value we set for the magic number.

From 000208 to 00020B: 4 bytes is the integer that tells what block the root starts. Value:0x00000006 shows that root starts at block 6 after converting value to decimal.

From 00020C to 00020F: 4 bytes is the integer for the size of each block. Value: 0x00000200 matches the block size which is 512 after converting value to decimal.

From 000210 to 000213: 4 bytes is the integer for the number of free blocks. Value: 0x00004C13 when converted to decimal is value 19475. The number is not 19531 because once we initialize the file system, blocks will be occupied.

From 000214 to 000217: 4 bytes is the integer for the number of blocks in the vcb. Value: 0x00004C4B is equivalent to decimal value 19531 which matches the correct number of blocks in the vcb for this project.

From 000218 to 00021B: 4 bytes is the integer for the number of blocks occupied by the bitmap. Value: 0x00000005 which is decimal value 5 and it matches the correct number of blocks that the bitmap occupies.

FREE SPACE:

```
000400: 00 00 00 00 00 00 00 00  01 00 00 00 01 00 00 00 | ................
000410: 01 00 00 00 01 00 00 00  01 00 00 00 01 00 00 00 | ................
000420: 01 00 00 00 01 00 00 00  01 00 00 00 01 00 00 00 | ................
000430: 01 00 00 00 01 00 00 00  01 00 00 00 01 00 00 00 | ................
000440: 01 00 00 00 01 00 00 00  01 00 00 00 01 00 00 00 | ................
000450: 01 00 00 00 01 00 00 00  01 00 00 00 01 00 00 00 | ................
000460: 01 00 00 00 01 00 00 00  01 00 00 00 01 00 00 00 | ................
000470: 01 00 00 00 01 00 00 00  01 00 00 00 01 00 00 00 | ................
000480: 01 00 00 00 01 00 00 00  01 00 00 00 01 00 00 00 | ................
000490: 01 00 00 00 01 00 00 00  01 00 00 00 01 00 00 00 | ................
0004A0: 01 00 00 00 01 00 00 00  01 00 00 00 01 00 00 00 | ................
0004B0: 01 00 00 00 01 00 00 00  01 00 00 00 01 00 00 00 | ................
```

```
0004C0: 01 00 00 00 01 00 00 00  01 00 00 00 01 00 00 00 | ................
0004D0: 01 00 00 00 01 00 00 00  01 00 00 00 01 00 00 00 | ................
0004E0: 01 00 00 00 01 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 | ................
```

From 00408 to 0004E7: 224 bytes represents the bitmap array that keeps track of the free spaces. The values of 1 represent that the spaces are taken. There are 50 entries for the root and 5 entries for the bitmap plus 1 for the vcb totaling 56 entries.

ROOT:

```
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 | ................

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  06 00 00 00 07 00 00 00 | ................
000F20: 08 00 00 00 09 00 00 00  0A 00 00 00 0B 00 00 00 | ................
000F30: 0C 00 00 00 0D 00 00 00  0E 00 00 00 0F 00 00 00 | ................
000F40: 10 00 00 00 11 00 00 00  12 00 00 00 13 00 00 00 | ................
000F50: 14 00 00 00 15 00 00 00  16 00 00 00 17 00 00 00 | ................
000F60: 18 00 00 00 19 00 00 00  1A 00 00 00 1B 00 00 00 | ................
000F70: 1C 00 00 00 1D 00 00 00  1E 00 00 00 1F 00 00 00 | ................
000F80: 20 00 00 00 21 00 00 00  22 00 00 00 23 00 00 00 |  ...!..."...#...
000F90: 24 00 00 00 25 00 00 00  26 00 00 00 27 00 00 00 | $...%...&...'...
000FA0: 28 00 00 00 29 00 00 00  2A 00 00 00 2B 00 00 00 | (...)...*...+...
000FB0: 2C 00 00 00 2D 00 00 00  2E 00 00 00 2F 00 00 00 | ,...-......./...
000FC0: 30 00 00 00 31 00 00 00  32 00 00 00 33 00 00 00 | 0...1...2...3...
000FD0: 34 00 00 00 35 00 00 00  36 00 00 00 37 00 00 00 | 4...5...6...7...
000FE0: 00 00 00 00 00 64 00 00  38 FD 30 64 00 00 00 00 | .....d..8�0d....
000FF0: 38 FD 30 64 00 00 00 00  38 FD 30 64 00 00 00 00 | 8�0d....8�0d....
001000: 2E 2E 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 | ................
```

From 000E00 to 000E07: 4 bytes is the name for the first directory entry. Value: 0x0000002E is '.' when converted from hex to text. This matches the name we set for the first directory entry.

From 000F18 to 000FDF: 456 bytes represents the array containing the locations of the file.

From 000FE4 to 000FE7: 4 bytes is the integer for the file size. Value: 0x00006400 which comes out to the decimal value 25600 and matches the correct file size.

From 000FE8 to 000FEB, 000FF0 to 000FF3, 000FF8 to 000FFB: 4 bytes each representing the time_t values for creation date, last access date, and last modification date respectively. Value:

0x6430FD38 converted from hex timestamp to human date comes out as Friday, April 7, 2023 10:35:52 PM which matches the time it was created, accessed, and modified.

From 001000 to 001003: 4 bytes represents the parent name. Value: 0x00002E2E matches the name of the second root directory entry, '..', when converted from hexadecimal to text.

VCB SRUCTURE:

```
typedef struct VCB
{
   unsigned long magic_n;
   unsigned int root;
   unsigned int block_size;
   unsigned int free_blocks;
   unsigned int number_of_blocks;
   unsigned int bitmap_total;
} VCB;
```

This is the Volume Control Block Struct(VCB) used to store information about the file system. We have six fields:

- 'magic_n'(magic number), to indicate the validity of the system.
- 'root' to hold the position of the root directory.
- 'block_size' to specify the size of each block which is 512 bytes.
- 'free_blocks' which indicates the number of free blocks.
- 'number_of_blocks' represents the total number of blocks in the file system.
- 'bitmap_total' holds the total number of blocks occupied by the bitmap(5 blocks).

FREE SPACE STRUCTURE:

```
typedef struct BitMap
{
   size_t size;
   unsigned int bitmap[];
} BitMap;
```

This is the struct we are using to manage our free space.

- 'size' indicates the total size of the bitmap in bits
- 'bitmap[]' is an array used to keep track of the free spaces in our file system.

This struct allows us to dynamically allocate memory for the bitmap array.

DIRECTORY SYSTEM:

```
typedef struct DirectoryEntry
{
   char file_name[280];                   // File Name
   unsigned int data_locations[MAX_ENTRIES]; // Array containing locations of file
   unsigned int type;                   // Either File or Directory Entry
   unsigned int file_size;               // File Size in Bytes
   time_t creation_date;                  // When was it Created
   time_t last_access;                  // when it was last accessed
   time_t last_mod;                     // when it was last modified
} DirectoryEntry;
```

The directory entry structure holds the information of the files in the file system.

- 'file_name' is a char array for the name of the file/directory and its maximum length is set to 280 characters.
- 'data_locations' is an array that contains the locations of the files.
- 'type' is an integer determining whether the entry is a directory or a file(0 or 1).
- 'file_size' gives the file size in bytes.
- 'creation_date' is to get the date and time when the file/directory was created.
- 'last_access' is to get the date and time when the file/directory was last accesed.
- 'last_mod' is to get the date and time when the file/directory was last modified.

Table for teamwork:

Task	Members
1)Check if volume needs to be initialized	Diego, Moe, Kemi
2) Initialize volume	Diego, Moe, Kemi
3) Initialize Free Space	Diego, Moe
4) Initialize Root directory	Diego, Moe
5) WriteUp	Kemi

For this milestone, we didn't have any solo tasks. We worked together for every task to complete the milestone. Diego led the team and we met virtually 4 times during the week and worked together for at least 3 hours each time until we completed the milestone.

Issues and resolutions:

We ran into an issue concerning how we would malloc our array of directory entry, because we could not figure out how to assign a set number of entries to it. We solved this issue by creating a normal array of directory entries that looks like this:

```
DirectoryEntry dir_entries[MAX_ENTRIES];
```

We also ran into an issue concerning how we formatted our drive. We couldn't figure out why we were getting errors when we tried to exit the init function if the system had already been initialised. We solved this issue by creating an if statement that frees the vcb and returns the magic number. Code:

```
if (vcb->magic_n == MAGIC_NUMBER)
        {
                printf("--Already initialized--\n");
                magic_n = vcb->magic_n;
                free(vcb);
                return magic_n;
        }
```

We ran into an issue where we were writing our free state directory entries to disk, but there was some random data on the blocks we were trying to write to on disk, so our hexdump was very confusing for blocks 50-57, we solved this by using memset to set everything to 0 before writing to disk. Code:

```
memset(&dir_entries[i], 0, vcb->block_size);
```

We also ran into an issue where our directory entry was about 470 bytes long, which would've been confusing later on because that would take up only 85% of the block and the next entry would start at byte 471, so we padded our directory entry to make it so that each entry takes up 1 block of 512 bytes which makes it easier to grasp and deal with later on in the project. "We just added more bytes to the filename variable" (char file_name[280];).