Operating Systems
CSCI 5806

Spring Semester 2021 — CRN 21176

Term Project — Step 3 — Low-Level Ext2 Access

Target completion date: Friday, March 12, 2021

Goals

- Provide functions to provide low-level access to an ext2 file system contained within a VDI file.
- Create a structure or class to contain the data necessary to implement these functions.

Details

In this step, we bridge the gap between "raw" data — disk partitions and byte / sector-level access — and "structured" data — file system-level access.

For convenience, you will still want two functions to open and close the file and a structure to hold necessary information. The functions are:

- **`struct Ext2File *ext2Open(char *fn,int32_t pNum)`**
  Use **`vdiOpen()`** and **`partitionOpen()`** to open the given VDI file and use partition number **`pNum`**. Populate all of the fields of your structure and return a pointer to it.
  Pro tip: Write your code so that **`pNum`**`=-1` uses the first partition with a Linux file type.
- **`void ext2Close(struct Ext2File *f)`**
  Close the file whose pointer is given. Deallocate any dynamically created memory regions.

Low-level ext2 access involves three structures — blocks, superblocks and block group descriptors.

▹*Blocks*

All space in an ext2 partition is divided into fixed-size blocks. The size of the blocks is determined by the superblock. With one exception, all disk access is performed by reading or writing entire blocks. To that end, you will need two block access functions:

- **`int32_t fetchBlock(struct Ext2File *f,uint32_t blockNum, void *buf)`**
  Read the given block number from the file system into the buffer. Return `0` if successful, some other value if the read fails.
- **`int32_t writeBlock(struct Ext2File *f,uint32_t blockNum, void *buf)`**
  Write the buffer to the given block in the file system. Return `0` if successful, some other value if the write fails.

There is one slight quirk in how the disk space is laid out; the main superblock (see next section) is *always* located in block zero. However, in a 1KB file system, that is the second physical 1KB block of space. The **`s_first_data_block`** field in the superblock indicates how many blocks to skip over to access block zero. The field is always `1` for 1KB file systems and `0` for all other block sizes.

▹*Superblocks*

The superblock is the main data structure in a UNIX file system. A good description of the structure can be found at https://www.nongnu.org/ext2-doc/ext2.html.

The main superblock is always located at an offset of `1024` bytes from the start of the disk partition, regardless of block size. Backup copies of the superblock are stored at various locations throughout the partition (see next section), always at the beginning of a block. You should read the main superblock and store it in the structure you create for this step.

There are two important values that the superblock does not directly contain, but need to be calculated from values in the superblock. The first value is the file system's block size. It is derived from the **`s_log_block_size`** field: $b = 1024 \cdot 2^{\texttt{s\_log\_block\_size}}$. The second value is the number of block groups, derived from the **`s_blocks_count`** and **`s_blocks_per_group`** fields: $n = \lceil \texttt{s\_blocks\_count/s\_blocks\_per\_group} \rceil$. Calculate these values and store them in the structure you create for this step.

You should write two functions for superblock access:

- **`int32_t fetchSuperblock(struct Ext2File *f,uint32_t blockNum, struct Ext2Superblock *sb)`**
  Read the superblock found in the given block number from the file system into the buffer. Return `0` for success, non-zero for failure.
- **`int32_t writeSuperblock(struct Ext2File *f,uint32_t blockNum, struct Ext2Superblock *sb)`**
  Write the superblock to the given block. Return `0` for success, non-zero for failure.

For these, use **`partitionSeek()`**, **`partitionRead()`** and **`partitionWrite()`** to access the main superblock in block `0`; use **`fetchBlock()`** and **`writeBlock()`** to access copies of the superblock. Verify that you have read a valid superblock by checking the **`s_magic`** field, it should be `0xef53`.

▹*Block Groups and Their Descriptors*

Blocks are split into block groups; groups act as a crude form of low-level disk access optimization, as the system typically tries to place all of the data blocks for one file in one block group. Each block group contains the following items:

- A copy of the superblock, if the block group number is `0`, `1` or a power of `3`, `5` or `7`. This is always contained in the first block of the group.
- A copy of the *block group descriptor table*, an array of block group descriptors. This array begins in the second block of the group and has as many blocks as necessary to hold the table. The table is stored contiguously (no gaps between entries). Copies are stored in the same groups that have superblock copies.
- A single block containing a bitmap of used and unused blocks in the group.
- A single block containing a bitmap of used and unused inodes in the group.
- A portion of the inode array.
- Data blocks.

A *block group descriptor* is a small structure that contains the block numbers of a group's bitmaps and the first block of the inode table, along with the number of unused blocks and inodes. See the link in the previous section for information about this structure.

You will need to read the main copy of the descriptor table and store it in the structure you create for this step. Since the table size is unknown until calculating the number of block groups, you will have to allocate the table space dynamically.

To access the table, implement these two functions:

- **`int32_t fetchBGDT(struct Ext2File *f,uint32_t blockNum,struct Ext2BlockGroupDescriptor *bgdt)`**
  Read the block group descriptor table that begins at the given block number. Store the table in the array pointed to by **`bgdt`**. Return `0` for success, non-zero for failure.
- **`int32_t writeBGDT(struct Ext2File *f,uint32_t blockNum,struct Ext2BlockGroupDescriptor *bgdt)`**
  Write the block group descriptor table to the file system starting at the given block number. Return `0` for success, non-zero for failure.

▹*Other Functions*

You will probably want a function to display a superblock with the fields labeled and in text form and a function to display the values in the block group descriptor table. The following examples illustrate my version of these.

▸*Example 1*

This is the output from my step 3 program, on the fixed VDI file with 1KB blocks. It shows the superblock in byte form, in readable form and then shows the block group descriptor table.

```
Offset: 0x400
   00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f    0...4...8...c...
  +-----------------------------------------------+  +----------------+
00|00 7f 00 00 00 fc 01 00 66 19 00 00 ef 53 00 00|00|        f    S  |
10|af 7d 00 00 01 00 00 00 00 00 00 00 00 00 00 00|10| }              |
20|00 20 00 00 00 20 00 00 f0 07 00 00 5f e7 a9 58|20|            _  X|
30|87 e7 a9 58 04 00 ff ff 53 ef 01 00 01 00 00 00|30|   X    S       |
40|88 bb ba 56 00 00 00 00 00 00 00 00 01 00 00 00|40|   V            |
50|00 00 00 00 0b 00 00 00 80 00 00 00 38 00 00 00|50|            8   |
60|02 00 00 00 01 00 00 00 5f 86 41 71 27 65 4b c9|60|        _ Aq'eK |
70|87 be a7 4a bb 9f 7d 28 00 00 00 00 00 00 00 00|70|   J  }(        |
80|00 00 00 00 00 00 00 00 2f 6d 65 64 69 61 2f 62|80|        /media/b|
90|6f 62 2f 35 66 38 36 34 31 37 31 2d 32 37 36 35|90|ob/5f864171-2765|
a0|2d 34 62 63 39 2d 38 37 62 65 2d 61 37 34 61 62|a0|-4bc9-87be-a74ab|
b0|62 39 66 37 64 32 38 00 64 32 38 00 00 00 00 00|b0|b9f7d28 d28     |
c0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01|c0|                |
d0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|d0|                |
e0|00 00 00 00 00 00 00 00 00 00 00 00 b5 7f 76 83|e0|              v |
f0|7f cd 4d 67 a6 34 20 ae 2f fd b0 6b 01 00 00 00|f0|  Mg 4  /  k    |
  +-----------------------------------------------+  +----------------+

Offset: 0x500
   00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f    0...4...8...c...
  +-----------------------------------------------+  +----------------+
00|0c 00 00 00 00 00 00 00 88 bb ba 56 00 00 00 00|00|           V    |
10|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|10|                |
20|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|20|                |
30|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|30|                |
40|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|40|                |
50|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|50|                |
60|01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|60|                |
70|00 00 00 00 00 00 00 00 44 93 01 00 00 00 00 00|70|        D       |
80|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|80|                |
90|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|90|                |
a0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|a0|                |
b0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|b0|                |
c0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|c0|                |
d0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|d0|                |
e0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|e0|                |
f0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|f0|                |
  +-----------------------------------------------+  +----------------+

Offset: 0x600
   00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f    0...4...8...c...
  +-----------------------------------------------+  +----------------+
00|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|00|                |
10|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|10|                |
20|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|20|                |
```

```
30|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|30|                  |
40|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|40|                  |
50|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|50|                  |
60|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|60|                  |
70|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|70|                  |
80|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|80|                  |
90|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|90|                  |
a0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|a0|                  |
b0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|b0|                  |
c0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|c0|                  |
d0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|d0|                  |
e0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|e0|                  |
f0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|f0|                  |
  +-----------------------------------------------+  +----------------+

Offset: 0x700
   00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f    0...4...8...c...
  +-----------------------------------------------+  +----------------+
00|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|00|                  |
10|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|10|                  |
20|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|20|                  |
30|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|30|                  |
40|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|40|                  |
50|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|50|                  |
60|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|60|                  |
70|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|70|                  |
80|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|80|                  |
90|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|90|                  |
a0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|a0|                  |
b0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|b0|                  |
c0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|c0|                  |
d0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|d0|                  |
e0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|e0|                  |
f0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|f0|                  |
  +-----------------------------------------------+  +----------------+

Superblock contents:
Number of inodes: 32512
Number of blocks: 130048
Number of reserved blocks: 6502
Number of free blocks: 21487
Number of free inodes: 32175
First data block: 1
Log block size: 0 (1024)
Log fragment size: 0 (1024)
Blocks per group: 8192
Fragments per group: 8192
Inodes per group: 2032
Last mount time: Sun Feb 19 13:43:43 2017
Last write time: Sun Feb 19 13:44:23 2017
Mount count: 4
Max mount count: 65535
Magic number: 0xef53
State: 1
```

```
Error processing: 1
Revision level: 1.0
Last system check: Tue Feb  9 23:24:40 2016
Check interval: 0
OS creator: 0
Default reserve UID: 0
Default reserve GID: 0
First inode number: 11
Inode size: 128
Block group number: 0
Feature compatibility bits: 0x00000038
Feature incompatibility bits: 0x00000002
Feature read/only compatibility bits: 0x00000001
UUID: 5f864171-2765-4bc9-87be-a74abb9f7d28
Volume name: []
Last mount point: [/media/bob/5f864171-2765-4bc9-87be-a74abb9f7d28]
Algorithm bitmap: 0x00000000
Number of blocks to preallocate: 0
Number of blocks to preallocate for directories: 0
Journal UUID: 5f864171-2765-4bc9-87be-a74abb9f7d28
Journal inode number: 0
Journal device number: 0
Journal last orphan inode number: 0
Default hash version: 1
Default mount option bitmap: 0x0000000c
First meta block group: 0

Block group descriptor table:
Block   Block   Inode   Inode   Free    Free    Used
Number  Bitmap  Bitmap  Table   Blocks  Inodes  Dirs
------  ------  ------  -----   ------  ------  ----
0       259     260     261     2499    2008    5
1       8451    8452    8453    1946    2012    6
2       16385   16386   16387   1258    2012    6
3       24835   24836   24837   1044    2012    7
4       32769   32770   32771   105     2011    5
5       41219   41220   41221   1025    2012    5
6       49153   49154   49155   1255    2011    5
7       57603   57604   57605   1041    2008    7
8       65537   65538   65539   1018    2010    6
9       73987   73988   73989   1528    2012    6
10      81921   81922   81923   762     2012    6
11      90113   90114   90115   1786    2010    6
12      98305   98306   98307   1786    2015    6
13      106497  106498  106499  1634    2016    5
14      114689  114690  114691  1787    2017    5
15      122881  122882  122883  1013    1997    10
```

▸*Example 2*

Same output from the dynamic-allocation VDI file with 1KB block size.

```
Offset: 0x400
   00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f    0...4...8...c...
  +-----------------------------------------------+  +----------------+
00|00 7f 00 00 00 fc 01 00 66 19 00 00 3b d6 01 00|00|        f   ;   |
10|f4 7e 00 00 01 00 00 00 00 00 00 00 00 00 00 00|10| ~              |
20|00 20 00 00 00 20 00 00 f0 07 00 00 db ea bc 56|20|               V|
30|19 eb bc 56 03 00 ff ff 53 ef 01 00 01 00 00 00|30|   V    S       |
40|9a bb ba 56 00 00 00 00 00 00 00 00 01 00 00 00|40|   V            |
50|00 00 00 00 0b 00 00 00 80 00 00 00 38 00 00 00|50|            8   |
60|02 00 00 00 01 00 00 00 71 2b 0f f6 04 66 4a a7|60|        q+   fJ |
70|86 c4 5d b7 72 22 07 09 00 00 00 00 00 00 00 00|70|  ] r"          |
80|00 00 00 00 00 00 00 00 2f 6d 65 64 69 61 2f 62|80|        /media/b|
90|6f 62 2f 37 31 32 62 30 66 66 36 2d 30 34 36 36|90|ob/712b0ff6-0466|
a0|2d 34 61 61 37 2d 38 36 63 34 2d 35 64 62 37 37|a0|-4aa7-86c4-5db77|
b0|32 32 32 30 37 30 39 00 00 00 00 00 00 00 00 00|b0|2220709         |
c0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01|c0|                |
d0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|d0|                |
e0|00 00 00 00 00 00 00 00 00 00 00 00 2d 98 fc 1b|e0|            -   |
f0|11 69 47 40 93 c8 52 24 9c 57 46 c9 01 00 00 00|f0| iG@  R$ WF     |
  +-----------------------------------------------+  +----------------+

Offset: 0x500
   00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f    0...4...8...c...
  +-----------------------------------------------+  +----------------+
00|0c 00 00 00 00 00 00 00 9a bb ba 56 00 00 00 00|00|           V    |
10|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|10|                |
20|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|20|                |
30|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|30|                |
40|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|40|                |
50|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|50|                |
60|01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|60|                |
70|00 00 00 00 00 00 00 00 b8 0f 00 00 00 00 00 00|70|                |
80|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|80|                |
90|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|90|                |
a0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|a0|                |
b0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|b0|                |
c0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|c0|                |
d0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|d0|                |
e0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|e0|                |
f0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|f0|                |
  +-----------------------------------------------+  +----------------+

Offset: 0x600
   00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f    0...4...8...c...
  +-----------------------------------------------+  +----------------+
00|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|00|                |
10|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|10|                |
20|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|20|                |
30|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|30|                |
40|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|40|                |
50|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|50|                |
```

```
60|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|60|                |
70|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|70|                |
80|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|80|                |
90|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|90|                |
a0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|a0|                |
b0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|b0|                |
c0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|c0|                |
d0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|d0|                |
e0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|e0|                |
f0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|f0|                |
  +-----------------------------------------------+  +----------------+

Offset: 0x700
   00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f    0...4...8...c...
  +-----------------------------------------------+  +----------------+
00|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|00|                |
10|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|10|                |
20|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|20|                |
30|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|30|                |
40|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|40|                |
50|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|50|                |
60|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|60|                |
70|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|70|                |
80|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|80|                |
90|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|90|                |
a0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|a0|                |
b0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|b0|                |
c0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|c0|                |
d0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|d0|                |
e0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|e0|                |
f0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|f0|                |
  +-----------------------------------------------+  +----------------+

Superblock contents:
Number of inodes: 32512
Number of blocks: 130048
Number of reserved blocks: 6502
Number of free blocks: 120379
Number of free inodes: 32500
First data block: 1
Log block size: 0 (1024)
Log fragment size: 0 (1024)
Blocks per group: 8192
Fragments per group: 8192
Inodes per group: 2032
Last mount time: Thu Feb 11 15:11:07 2016
Last write time: Thu Feb 11 15:12:09 2016
Mount count: 3
Max mount count: 65535
Magic number: 0xef53
State: 1
Error processing: 1
Revision level: 1.0
Last system check: Tue Feb  9 23:24:58 2016
```

```
Check interval: 0
OS creator: 0
Default reserve UID: 0
Default reserve GID: 0
First inode number: 11
Inode size: 128
Block group number: 0
Feature compatibility bits: 0x00000038
Feature incompatibility bits: 0x00000002
Feature read/only compatibility bits: 0x00000001
UUID: 712b0ff6-0466-4aa7-86c4-5db772220709
Volume name: []
Last mount point: [/media/bob/712b0ff6-0466-4aa7-86c4-5db772220709]
Algorithm bitmap: 0x00000000
Number of blocks to preallocate: 0
Number of blocks to preallocate for directories: 0
Journal UUID: 712b0ff6-0466-4aa7-86c4-5db772220709
Journal inode number: 0
Journal device number: 0
Journal last orphan inode number: 0
Default hash version: 1
Default mount option bitmap: 0x0000000c
First meta block group: 0

Block group descriptor table:
Block   Block   Inode   Inode   Free    Free    Used
Number  Bitmap  Bitmap  Table   Blocks  Inodes  Dirs
------  ------  ------  -----   ------  ------  ----
0       259     260     261     6623    2020    2
1       8451    8452    8453    4709    2032    0
2       16385   16386   16387   7936    2032    0
3       24835   24836   24837   7678    2032    0
4       32769   32770   32771   7936    2032    0
5       41219   41220   41221   7678    2032    0
6       49153   49154   49155   7936    2032    0
7       57603   57604   57605   7678    2032    0
8       65537   65538   65539   7936    2032    0
9       73987   73988   73989   7678    2032    0
10      81921   81922   81923   7936    2032    0
11      90113   90114   90115   7936    2032    0
12      98305   98306   98307   7936    2032    0
13      106497  106498  106499  7936    2032    0
14      114689  114690  114691  7936    2032    0
15      122881  122882  122883  6911    2032    0
```

▸*Example 3*

This is the program's output using the test VDI file with 4KB block size.

```
Offset: 0x400
   00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f    0...4...8...c...
  +-----------------------------------------------+  +----------------+
00|00 7f 00 00 00 7f 00 00 59 06 00 00 f2 15 00 00|00|        Y       |
10|af 7d 00 00 00 00 00 00 02 00 00 00 02 00 00 00|10| }              |
20|00 80 00 00 00 80 00 00 00 7f 00 00 d8 ea bc 56|20|               V|
30|19 eb bc 56 03 00 ff ff 53 ef 01 00 01 00 00 00|30|   V    S       |
40|92 bb ba 56 00 00 00 00 00 00 00 00 01 00 00 00|40|   V            |
50|00 00 00 00 0b 00 00 00 80 00 00 00 38 00 00 00|50|             8   |
60|02 00 00 00 03 00 00 00 8c b4 8d bc 5c 10 4e 70|60|             \ Np|
70|a5 68 cd d0 ad 4f 12 0e 00 00 00 00 00 00 00 00|70| h   O          |
80|00 00 00 00 00 00 00 00 2f 6d 65 64 69 61 2f 62|80|        /media/b|
90|6f 62 2f 38 63 62 34 38 64 62 63 2d 35 63 31 30|90|ob/8cb48dbc-5c10|
a0|2d 34 65 37 30 2d 61 35 36 38 2d 63 64 64 30 61|a0|-4e70-a568-cdd0a|
b0|64 34 66 31 32 30 65 00 32 30 65 00 00 00 00 00|b0|d4f120e 20e     |
c0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 07 00|c0|                |
d0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|d0|                |
e0|00 00 00 00 00 00 00 00 00 00 00 00 05 e6 e3 aa|e0|                |
f0|bf 36 43 4a 9f c4 82 9f af f7 80 c0 01 00 00 00|f0| 6CJ            |
  +-----------------------------------------------+  +----------------+

Offset: 0x500
   00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f    0...4...8...c...
  +-----------------------------------------------+  +----------------+
00|0c 00 00 00 00 00 00 00 92 bb ba 56 00 00 00 00|00|           V    |
10|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|10|                |
20|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|20|                |
30|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|30|                |
40|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|40|                |
50|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|50|                |
60|01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|60|                |
70|00 00 00 00 00 00 00 00 38 95 01 00 00 00 00 00|70|        8       |
80|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|80|                |
90|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|90|                |
a0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|a0|                |
b0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|b0|                |
c0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|c0|                |
d0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|d0|                |
e0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|e0|                |
f0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|f0|                |
  +-----------------------------------------------+  +----------------+

Offset: 0x600
   00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f    0...4...8...c...
  +-----------------------------------------------+  +----------------+
00|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|00|                |
10|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|10|                |
20|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|20|                |
30|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|30|                |
40|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|40|                |
50|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|50|                |
```

```
60|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|60|                 |
70|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|70|                 |
80|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|80|                 |
90|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|90|                 |
a0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|a0|                 |
b0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|b0|                 |
c0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|c0|                 |
d0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|d0|                 |
e0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|e0|                 |
f0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|f0|                 |
  +-----------------------------------------------+  +----------------+

Offset: 0x700
   00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f    0...4...8...c...
  +-----------------------------------------------+  +----------------+
00|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|00|                 |
10|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|10|                 |
20|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|20|                 |
30|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|30|                 |
40|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|40|                 |
50|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|50|                 |
60|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|60|                 |
70|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|70|                 |
80|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|80|                 |
90|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|90|                 |
a0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|a0|                 |
b0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|b0|                 |
c0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|c0|                 |
d0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|d0|                 |
e0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|e0|                 |
f0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|f0|                 |
  +-----------------------------------------------+  +----------------+

Superblock contents:
Number of inodes: 32512
Number of blocks: 32512
Number of reserved blocks: 1625
Number of free blocks: 5618
Number of free inodes: 32175
First data block: 0
Log block size: 2 (4096)
Log fragment size: 2 (4096)
Blocks per group: 32768
Fragments per group: 32768
Inodes per group: 32512
Last mount time: Thu Feb 11 15:11:04 2016
Last write time: Thu Feb 11 15:12:09 2016
Mount count: 3
Max mount count: 65535
Magic number: 0xef53
State: 1
Error processing: 1
Revision level: 1.0
Last system check: Tue Feb  9 23:24:50 2016
```

```
Check interval: 0
OS creator: 0
Default reserve UID: 0
Default reserve GID: 0
First inode number: 11
Inode size: 128
Block group number: 0
Feature compatibility bits: 0x00000038
Feature incompatibility bits: 0x00000002
Feature read/only compatibility bits: 0x00000003
UUID: 8cb48dbc-5c10-4e70-a568-cdd0ad4f120e
Volume name: []
Last mount point: [/media/bob/8cb48dbc-5c10-4e70-a568-cdd0ad4f120e]
Algorithm bitmap: 0x00000000
Number of blocks to preallocate: 0
Number of blocks to preallocate for directories: 0
Journal UUID: 8cb48dbc-5c10-4e70-a568-cdd0ad4f120e
Journal inode number: 0
Journal device number: 0
Journal last orphan inode number: 0
Default hash version: 1
Default mount option bitmap: 0x0000000c
First meta block group: 0

Block group descriptor table:
Block   Block   Inode   Inode   Free    Free    Used
Number  Bitmap  Bitmap  Table   Blocks  Inodes  Dirs
------  ------  ------  -----   ------  ------  ----
0       9       10      11      5618    32175   96
```