

Project 3: FAT32 File System

Part 3

Function: read FILENAME OFFSET SIZE

- Read SIZE bytes of the FILENAME starting at OFFSET and print to stdout
- Throw error if
 - File doesn't exist in current working directory
 - File is directory
 - File isn't open for reading
 - Offset is larger than file size

Function: read FILENAME OFFSET SIZE

- Data is stored in a file as RAW bytes inside the FAT32 image file (no worrying about endianness)
- Don't worry about the contents – use fwrite to write to stdout
- The example below shows a text file with the contents “Hi there!\n”

```
001361E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
001361F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00136200  48 69 20 74 68 65 72 65 21 0A 00 00 00 00 00 00 Hi there!.....
00136210  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00136220  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00136230  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00136240  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00136250  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00136260  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00136270  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00136280  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

Function: read FILENAME OFFSET SIZE

- To read in the contents from FILENAME,
 - check that FILENAME is open with MODE r, rw, or wr
 - Find position in the image file to read
 - Use fread to read the image file
- You may need to iterate through multiple clusters to print the requested content

Function: read FILENAME OFFSET SIZE

- Note: OFFSET may not be perfectly divisible by sector_size (i.e. OFFSET may not be synchronized with the cluster.)
- The / and % operators are your friend:
 - $\text{OFFSET} / \text{bytes_per_sec} = \text{cluster offset}$
 - $\text{OFFSET} \% \text{bytes_per_sec} = \text{byte offset within cluster}$

Function: read FILENAME OFFSET SIZE

At *each* cluster, need to calculate which bytes to read based on SIZE and OFFSET!

- Edge cases: $\text{OFFSET} > \text{sizeof}(\text{FILENAME})$
 - Print error
- $\text{SIZE} > \text{sizeof}(\text{FILENAME})$,
 - Print entire file
- $\text{OFFSET} + \text{SIZE} > \text{sizeof}(\text{FILENAME})$
 - Prints $\text{sizeof}(\text{FILENAME}) - \text{OFFSET}$ bytes (from offset to end of file)

Function: read FILENAME OFFSET SIZE

```
vagrant@ubuntu-bionic:~/project3/src$ ./FAT32 fat32.img
```

```
Welcome to the ./FAT32 shell utility
```

```
Image, fat32.img, is ready to view
```

```
For a list of commands, type "help" or "h"
```

```
/] open hello r
```

```
/] size hello
```

```
10
```

```
/] read hello 0 10
```

```
Hi there!
```

```
/] read hello 2 5
```

```
ther
```

```
/] read hello 5 12
```

```
ere!
```

```
/] exit
```

```
exiting fat32 utility
```

```
vagrant@ubuntu-bionic:~/project3/src$
```

Function: write FILENAME OFFSET SIZE STRING

- Write SIZE bytes in the form of STRING to FILENAME starting at OFFSET
- Throw error if
 - Filename doesn't exist
 - Filename is a directory
 - File is not open for writing

Function: write FILENAME OFFSET SIZE STRING

- To write STRING to FILENAME,
 - Check MODE to ensure file was opened with w, wr, rw
 - Find position in image file
 - Write STRING to image file using fwrite
- Once again, the file might span multiple clusters
 - You may need to allocate new clusters

Function: write FILENAME OFFSET SIZE STRING

- Edge cases:
- $\text{OFFSET} > \text{sizeof}(\text{FILENAME})$
 - Print error
- $\text{OFFSET} < \text{sizeof}(\text{FILENAME})$
 - $\text{OFFSET} + \text{SIZE} \leq \text{sizeof}(\text{FILENAME})$
 - Overwrite the data there
 - $\text{OFFSET} + \text{SIZE} > \text{sizeof}(\text{FILENAME})$
 - Overwrite data starting at OFFSET and expand the file to contain the rest of the buffer
 - This may require allocating a new cluster
- Make sure to update DIR_Size appropriately for all writes!

Function: write FILENAME OFFSET SIZE STRING

- What if the length of STRING is not equal to SIZE?
 - If `sizeof(STRING) < SIZE`, write STRING and pad the rest of the space with zeroes (`'\0'`, null byte)
 - If `sizeof(STRING) >= SIZE`, just use the first SIZE bytes of STRING

Expanding Files

- Find an empty cluster
- Have last cluster of file point to new cluster
- Have new cluster point to end of file

Function: write FILENAME OFFSET SIZE STRING

```
vagrant@ubuntu-bionic:~/project3/src$ ./FAT32 fat32.img
```

```
Welcome to the ./FAT32 shell utility
```

```
Image, fat32.img, is ready to view
```

```
For a list of commands, type "help" or "h"
```

```
/] creat apple
```

```
/] creat banana
```

```
/] size apple
```

```
0
```

```
/] open apple wr
```

```
/] write apple 500 16 "This is an apple"
```

```
/] read apple 500 16
```

```
This is an apple
```

```
/] size apple
```

```
516
```

```
/] exit
```

```
exiting fat32 utility
```

```
vagrant@ubuntu-bionic:~/project3/src$
```

Function: write FILENAME OFFSET SIZE STRING

File Allocation Table

000048E0	41	02	00	00	FF	FF	FF	0F	FF	FF	FF	0F	FF	FF	FF	0F	A.....
000048F0	FF	FF	FF	0F	FF	FF	FF	0F	FF	FF	FF	0F	FF	FF	FF	0F
00004900	FF	FF	FF	0F	FF	FF	FF	0F	FF	FF	FF	0F	FF	FF	FF	0F
00004910	FF	FF	FF	0F	FF	FF	FF	0F	48	02	00	00	F8	FF	FF	0FH.....
00004920	F8	FF	FF	0F	00	00	00	00	00	00	00	00	00	00	00	00
00004930	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00004940	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00004950	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Apple

Banana

- Apple contains two clusters (0x0246 and 0x0248) – the first cluster points to the second cluster and the second cluster points to end of file (0x0FFFFFFF8)
- The banana cluster sits in between the apple cluster and points to end of file

Function: write FILENAME OFFSET SIZE STRING

End of apple first cluster

Banana cluster

Start of apple second cluster

```
00148DD0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148DE0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148DF0 00 00 00 00 54 68 69 73 20 69 73 20 61 6E 20 61 ....This is an a
00148E00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148E10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148E20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148E30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148E40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148E50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148E60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148E70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148E80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148E90 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148EA0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148EB0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148EC0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148ED0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148EE0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148EF0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148F00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148F10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148F20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148F30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148F40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148F50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148F60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148F70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148F80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148F90 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148FA0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148FB0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148FC0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148FD0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148FE0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00148FF0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00149000 70 70 6C 65 00 00 00 00 00 00 00 00 00 00 00 00 .....
00149010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00149020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00149030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00149040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00149050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00149060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- fat32.img --0x149000/0x4000000-----
```

When in doubt, check the FATspec!