

# Project 3: FAT32 File System

## Part 2

# Modifying the FAT32 Image File

- **Open file with “r+”**
  - r+ -> allows reading and updating
  - Opening with w or w+ will overwrite the entire image file!
- **Warning: incorrectly modifying the image can corrupt it**
  - For instance, think about how an incorrect value for a cluster number can mess up an entire cluster chain
  - But don't panic, you can simply copy a clean image file

# Optional Tip

- While debugging, you are likely to mess up your image file which will make future tests inaccurate
- To aid in development, I suggest adding a target to your makefile which reloads a clean image file (by copying a clean version from somewhere else)

# Function: creat FILENAME

- Search current directory to see if FILENAME is unique
  - If there is a file with the same name, then return an error
- To create a new file, need to find a place to put it
- Iterate through FAT table and look for an empty entry (marked by 0x0) -> this cluster is available for a new file
  - Change entry to end of cluster marker, this means that you have allocated this space to be used and that the file is only one cluster long
  - If no empty clusters, FAT is full, return an error

# Function: creat FILENAME

- Need to create a new DIR entry in the *current* directory to direct users to new file location
- Point the new DIR entry in the current directory to the previously found empty cluster
  - Set DIR\_ClusHI to the top bits of empty cluster, and DIR\_ClusLO set to the bottom bits of the empty cluster
- Initialize the other fields described in the FATspec
  - ex) Set DIR\_Size to 0

# Function: creat FILENAME

```
00100490  FF FF FF FF  FF FF FF FF  FF FF 00 00  FF FF FF FF  .....
001004A0  42 4C 55 45  20 20 20 20  20 20 20 10  00 64 04 8E  BLUE    ..d..
001004B0  78 4E 78 4E  00 00 04 8E  78 4E B2 01  00 00 00 00  xNxN....xN....
001004C0  41 67 00 72  00 65 00 65  00 6E 00 0F  00 42 00 00  Ag.r.e.e.n...B..
001004D0  FF FF FF FF  FF FF FF FF  FF FF 00 00  FF FF FF FF  .....
001004E0  47 52 45 45  4E 20 20 20  20 20 20 10  00 64 04 8E  GREEN    ..d..
001004F0  78 4E 78 4E  00 00 04 8E  78 4E B3 01  00 00 00 00  xNxN....xN....
00100500  41 72 00 65  00 64 00 00  00 FF FF 0F  00 37 FF FF  Ar.e.d.....7..
00100510  FF FF FF FF  FF FF FF FF  FF FF 00 00  FF FF FF FF  .....
00100520  52 45 44 20  20 20 20 20  20 20 20 10  00 00 05 8E  RED      .....
00100530  78 4E 78 4E  00 00 05 8E  78 4E B4 01  00 00 00 00  xNxN....xN....
00100540  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  .....
00100550  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  .....
00100560  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  .....
00100570  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  .....
00100580  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  .....
00100590  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  .....
001005A0  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  .....
001005B0  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  .....
001005C0  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  .....
001005D0  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  .....
001005E0  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  .....
--- fat32.img      --0x1005E0/0x4000000-----
```

Before creat

First empty  
Directory entry

# Function: creat FILENAME

```
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 test
/] ls
longfile          hello   blue/   green/  red/    test
/] exit
exiting fat32 utility
vagrant@ubuntu-bionic:~/project3/src$
```

# Function: creat FILENAME

```
00100490  FF FF FF FF FF FF FF FF FF FF 00 00 FF FF FF FF .....
001004A0  42 4C 55 45 20 20 20 20 20 20 20 10 00 64 04 8E BLUE ..d..
001004B0  78 4E 78 4E 00 00 04 8E 78 4E B2 01 00 00 00 00 xNxN....xN.....
001004C0  41 67 00 72 00 65 00 65 00 6E 00 0F 00 42 00 00 Ag.r.e.e.n...B..
001004D0  FF FF FF FF FF FF FF FF FF FF 00 00 FF FF FF FF .....
001004E0  47 52 45 45 4E 20 20 20 20 20 20 10 00 64 04 8E GREEN ..d..
001004F0  78 4E 78 4E 00 00 04 8E 78 4E B3 01 00 00 00 00 xNxN....xN.....
00100500  41 72 00 65 00 64 00 00 00 FF FF 0F 00 37 FF FF Ar.e.d.....7..
00100510  FF FF FF FF FF FF FF FF FF FF 00 00 FF FF FF FF .....
00100520  52 45 44 20 20 20 20 20 20 20 10 00 00 05 8E RED .....
00100530  78 4E 78 4E 00 00 05 8E 78 4E B4 01 00 00 00 00 xNxN....xN.....
00100540  54 45 53 54 20 20 20 20 20 20 20 00 00 00 86 9E TEST .....
00100550  66 4F 66 4F 00 00 86 9E 66 4F 46 02 00 00 00 00 fofo....foF.....
00100560  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00100570  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00100580  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00100590  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
001005A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
001005B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
001005C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
001005D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
001005E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- fat32.img ---0x100490/0x4000000-----
```

After creat

Directory entry  
now populated

DIR\_FstClusHI

DIR\_FstClusLO



# Function: creat FILENAME

```
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 FF FF FF 0F 00 00 00 00 .....
00004920  00 00 00 00 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 .....
--- fat32.img --0x4918/0x4000000-----
```

- DIR\_FstClusHI = 0x0000, DIR\_FstClusLO = 0x0246 (remember little endian) → cluster number = 0x00000246
- This corresponds to offset 0x4918 which has 0x0FFFFFFF8 as its value (marking the last cluster in file)

# Function: mkdir DIRNAME

- Very similar to creat
- Need to set DIR\_Attr = 0x10

# Function: mkdir DIRNAME

- You need to initialize the “.” (current) and “..” (parent) directories
- Go to the new directory’s cluster
- Write “.” and “..” directories
  - So, “.” will have DIR\_Cluster = new\_dir\_cluster
  - And “..” will have DIR\_Cluster = parent directory’s cluster number

# Function: mkdir DIRNAME

```
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"
/] mkdir yellow
/] ls
longfile          hello   blue/   green/  red/    test   yellow/
/] cd yellow
/yellow/] ls
./      ../
/yellow/] exit
exiting fat32 utility
vagrant@ubuntu-bionic:~/project3/src$
```

# Function: mkdir DIRNAME

Yellow

00148DF0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	.....
00148E00	2E 20 20 20	20 20 20 20	20 20 20 10	00 00 9A 14	. ....
00148E10	67 4F 67 4F	00 00 9A 14	67 4F 47 02	00 00 00 00	g0g0....g0G.....
00148E20	2E 2E 20 20	20 20 20 20	20 20 20 10	00 00 9A 14	.. ....
00148E30	67 4F 67 4F	00 00 9A 14	67 4F 00 00	00 00 00 00	g0g0....g0.....
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	.....

Dot file

Dot dot file

# Function: open FILENAME MODE

- You need to have some sort of record to remember which files are open and which mode they were opened with
- Linked list is a good option
  - Create a struct to represent an open file which contains an integer for the file's first cluster number, and an integer or short integer to represent the write mode
  - (ie mode 1 = read, 2 = write, 3 = read and write)
  - Make a linked list of these struct instances
  - Need global variable pointer to start of list, initialize to NULL before any files are opened

# Function: open FILENAME MODE

- First, check if FILENAME is in the current directory
- Make sure FILENAME is not a directory ( $\text{DIR\_Attr} \& 0x10 == 0x00$ )
- Check FILENAME's permissions- is the file read-only?
  - If  $\text{DIR\_Attr} \& 0x01 == 0x01$ , file is read-only
  - You need to enforce the read-only property
  - If the file is read-only and MODE is W, RW, or WR, return an error

# Function: open FILENAME MODE

- After checking for valid FILENAME and MODE arguments,
  - Get FILENAME's first\_cluster\_number (remember you must combine the HI and LO)
  - Check if the file is open by searching through the linked list
  - If first\_cluster\_number is in the linked list, the file is already open, return an error
  - Else, create a struct entry for the file with the FILENAME's first\_cluster\_number and MODE
  - Add the struct to the linked list



# Function: close FILENAME

- Check if FILENAME is in the current directory, if not, print an error
- Check if FILENAME's first\_cluster\_number is in the linked list
  - If it is, remove the entry
  - If it isn't, print an error