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)

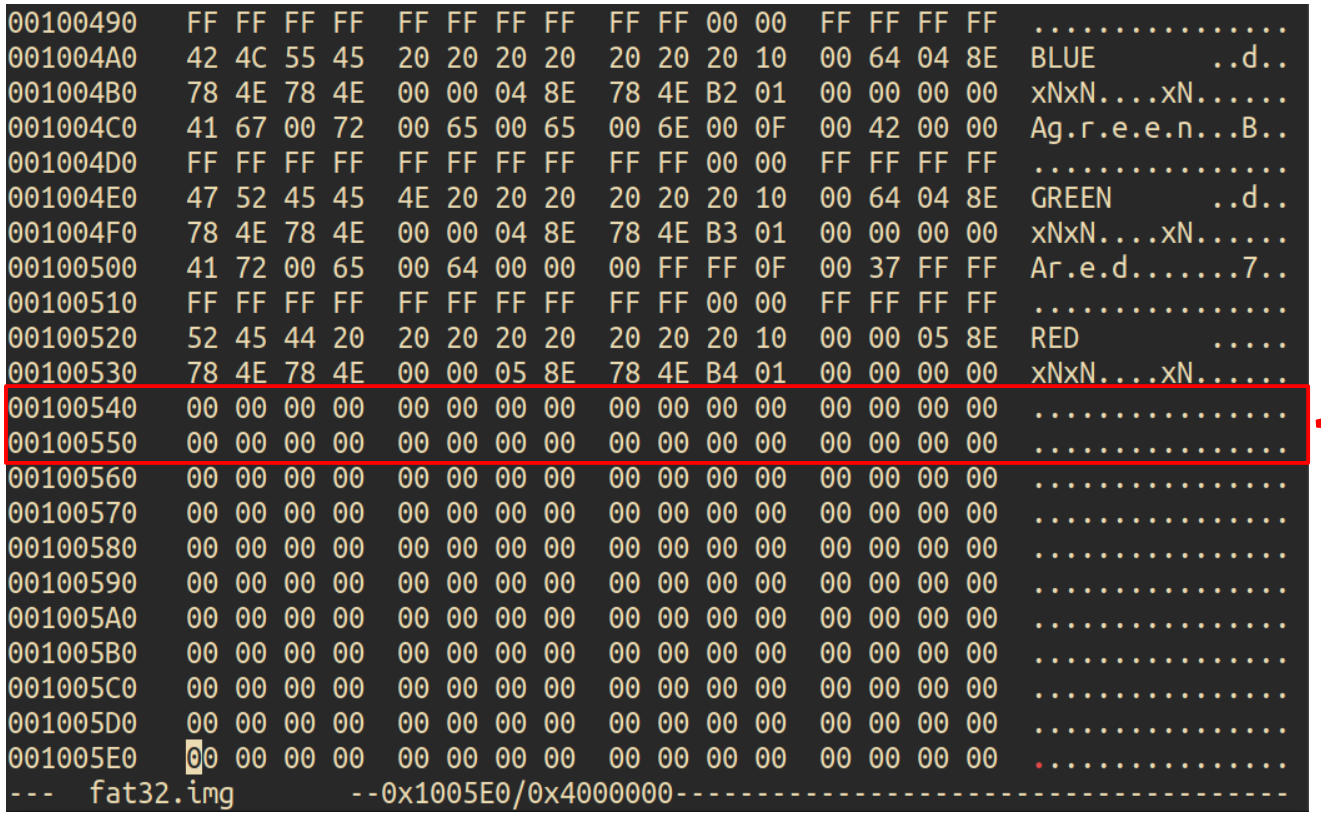
Function: creat FILENAME

- 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



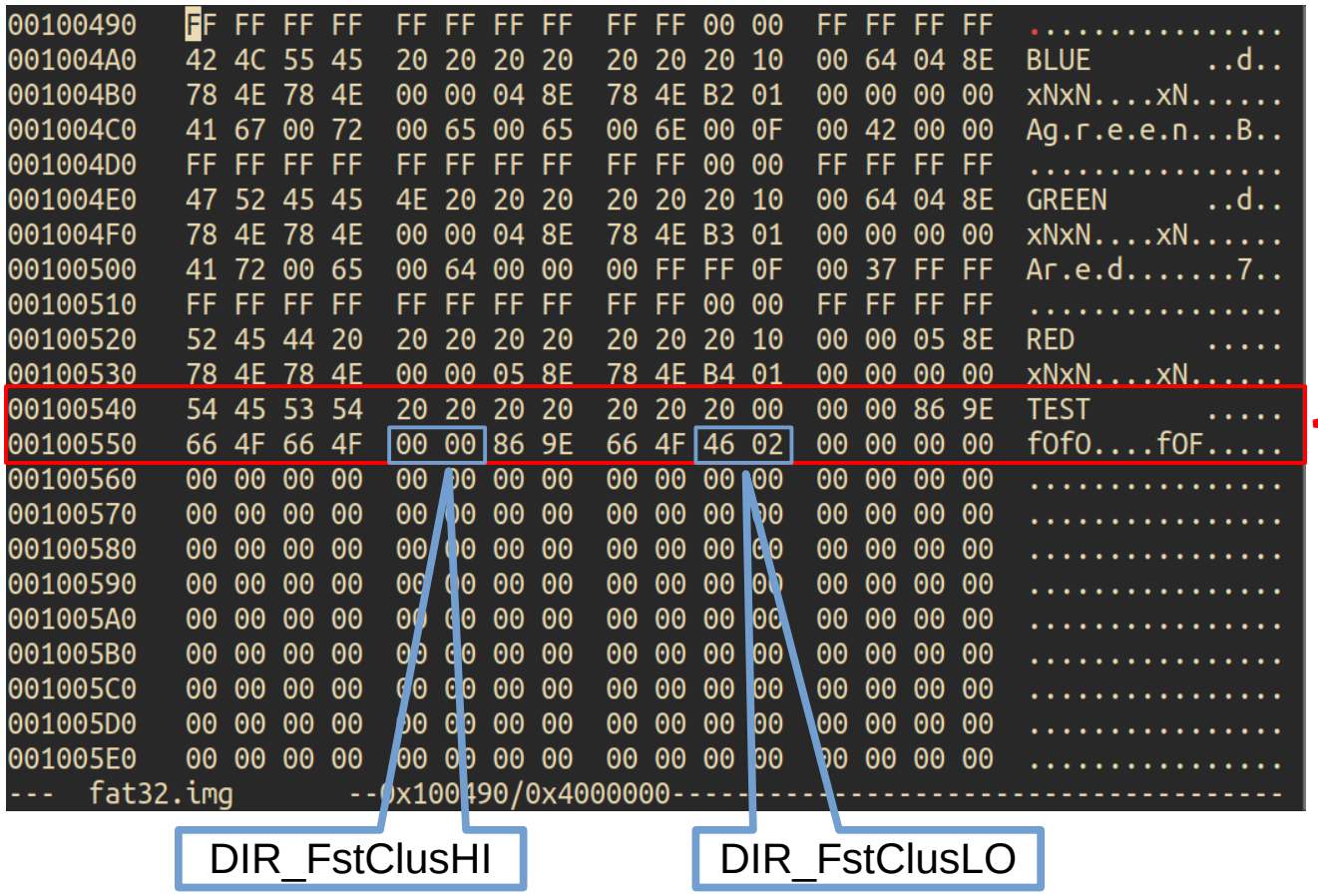
Before creat

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$
```

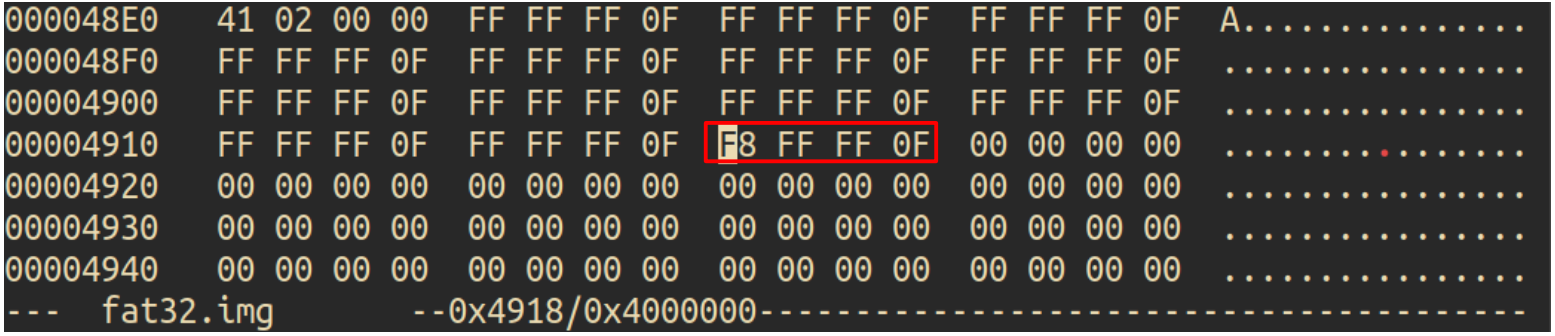
Function: creat FILENAME



After creat

Directory entry now populated

Function: creat FILENAME



- DIR_FstClusHI = 0x0000, DIR_FstClusLO = 0x0246 (remember little endian) → cluster number = 0x00000246
- This corresponds to offset 0x4918 which has 0x0FFFFFF8 as its value (marking the last cluster in file)

Function: mkdir DIRNAME

- Very similar to creat
- Need to set DIR_Attr = 0x10

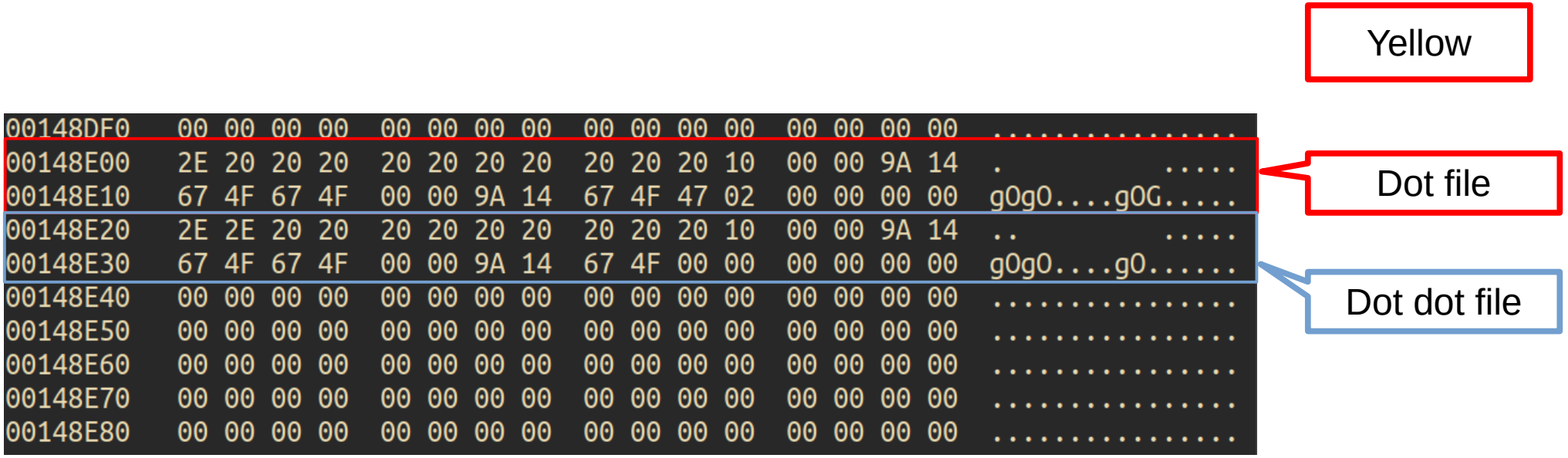
Function: mkdir DIRNAME

- 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



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 (DIR_Attr & 0x10 == 0x00)
- Check FILENAME's permissions- is the file read-only?
  - If 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