Project 3: FAT32 File System Implementation

If Dark Souls Was A Programming Assignment

Last Week You ..

- Learnt in depth about how a FAT32 File System works
- Understood about directory entries and cluster numbers
- Realized many read-based file operations like ls, cd, size, etc
- Tried to implement those functions in C

So What Next?

- This time we will be working on the next 4 functions:
  mkdir, creat, rmdir, rm

- These functions are unlike the previous ones as the modify the file image

Some Things To Note

- Use rb+ as the mode of opening the image file
- Modifying the file image is risky. You may overwrite/ corrupt the file system
- Easy way to check that is running ls/cd commands just after that to ensure everything is in order
- Use the hex editor to check whether correct values are placed in the correct places

Some Utilities First

- For mkdir, creat (and partially for write), we need to have some function that will tell us where we can get an empty cluster.

- This is because, all of these functions require an empty cluster to be assigned to a directory/file

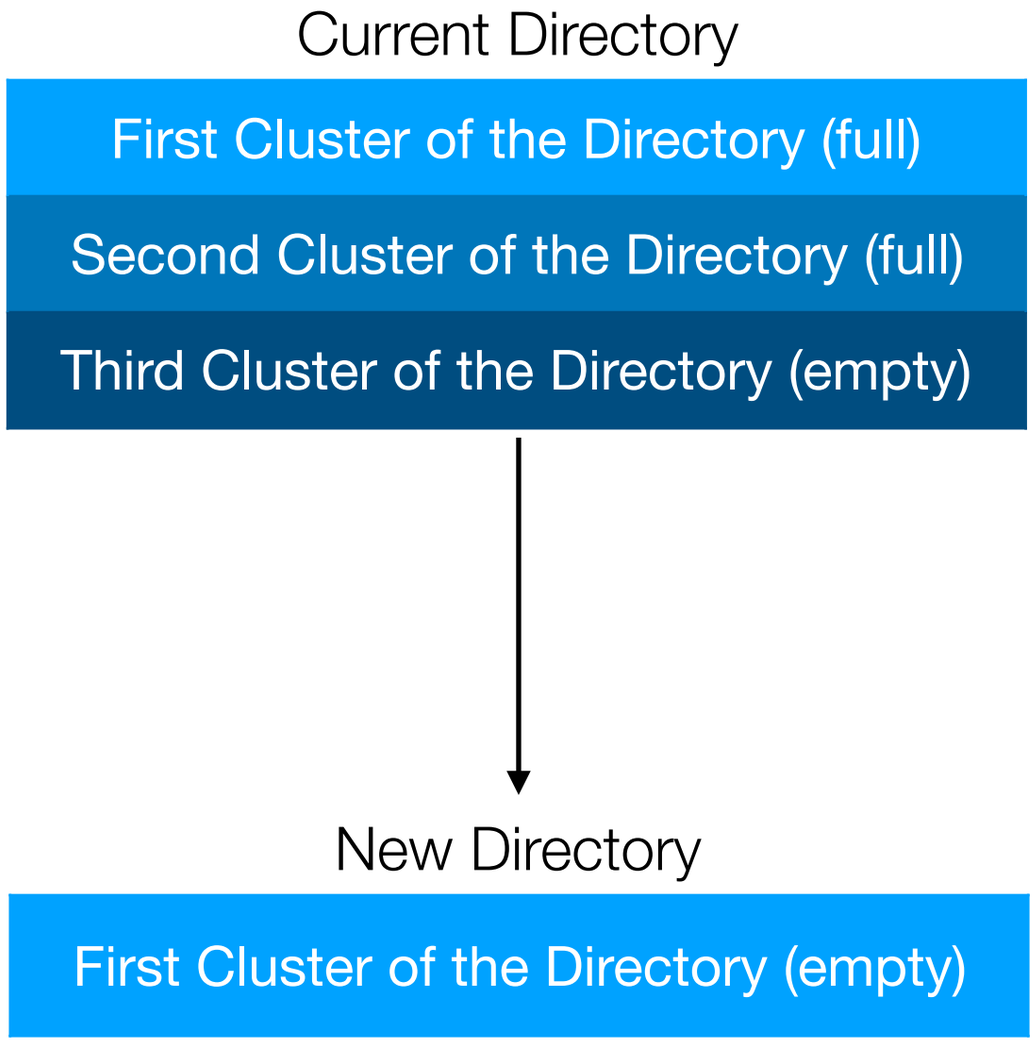
- It is hence, easier for us to have a function which can give us the number for an empty cluster from the FAT

How to get an empty cluster?

- As discussed previously, if FAT[i] = 0x00000000, then that is an empty cluster.
- So mkdir, (and creat) functions will first find an empty cluster in the FAT Table
- Make a function in your program such that it returns an empty cluster number from the FAT Table
- Function find_empty_cluster should:
- Iterate through the FAT Table
- If you get an ‘i’ such that FAT[i] == 0x00000000, then return that ‘i’
- Else return -1 (means we do not have any empty cluster -> FAT is full)

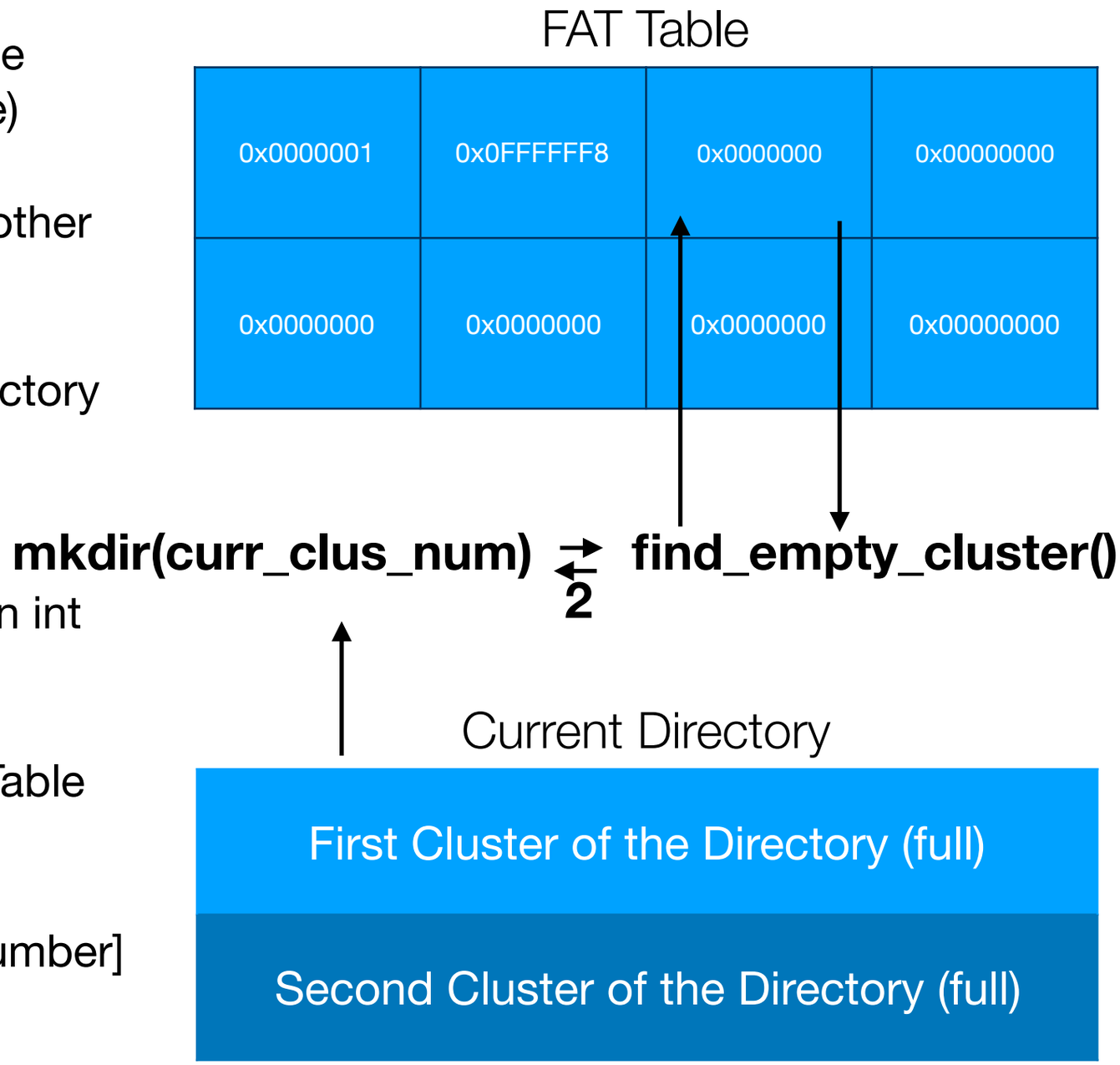
mkdir DIRNAME

- Makes a new directory in the file system.
- You have to make a directory entry in the present directory
- Then allocate an empty cluster for the directory
- And link the empty cluster number to the new directory entry



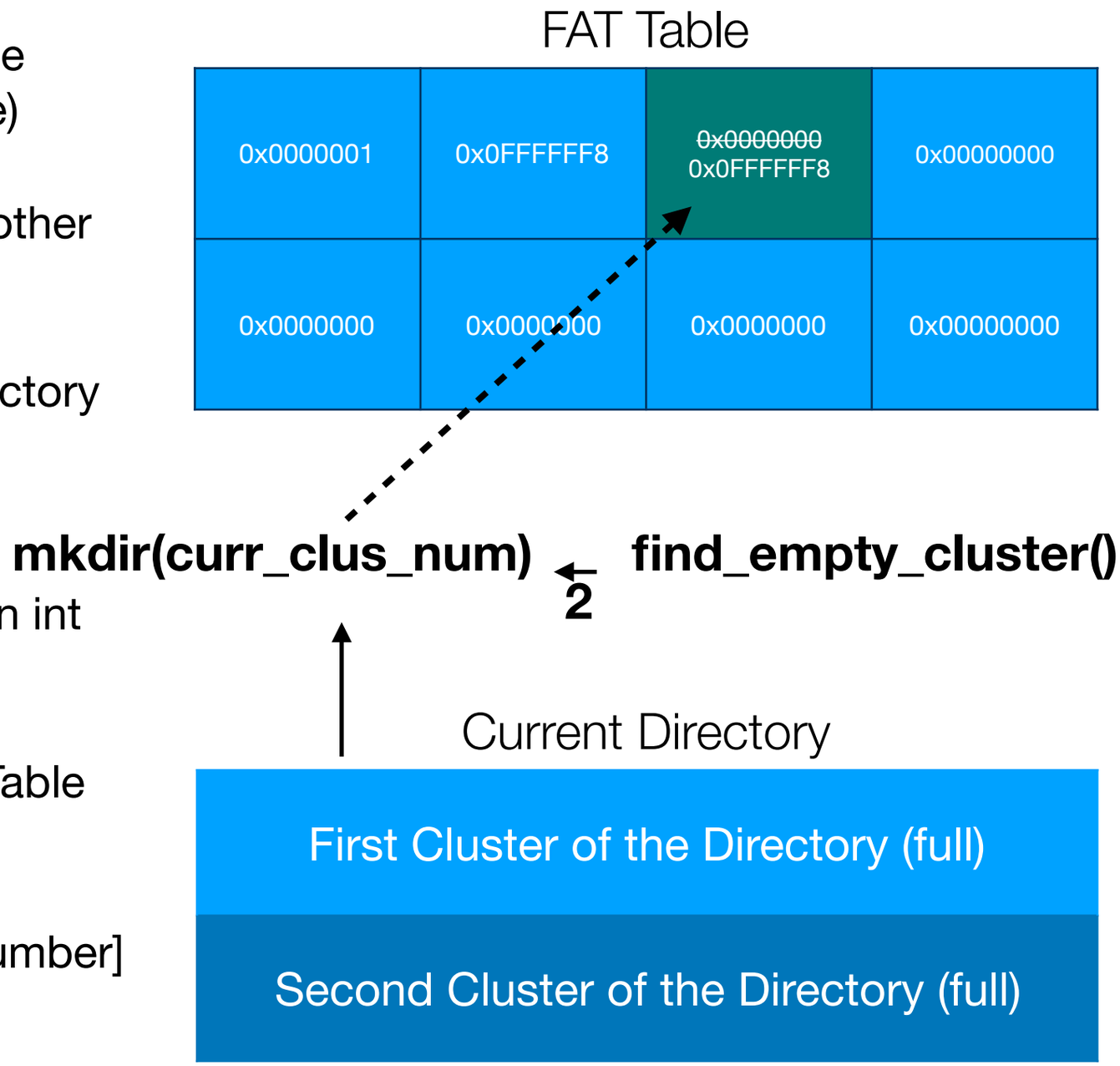
mkdir DIRNAME

- First iterate through the whole of the current directory (just like ls/cd/size)
- Then check if there is space for another directory entry
- If there is no space for another directory entry, call the find_empty_cluster function
- Lets say that the function returns an int value ‘i’
- If ‘i’ is NOT -1, then go to the FAT Table and set FAT[i] = 0x0FFFFFF8
- Now set the FAT[current_cluster_number] == Hex Code of ‘i’



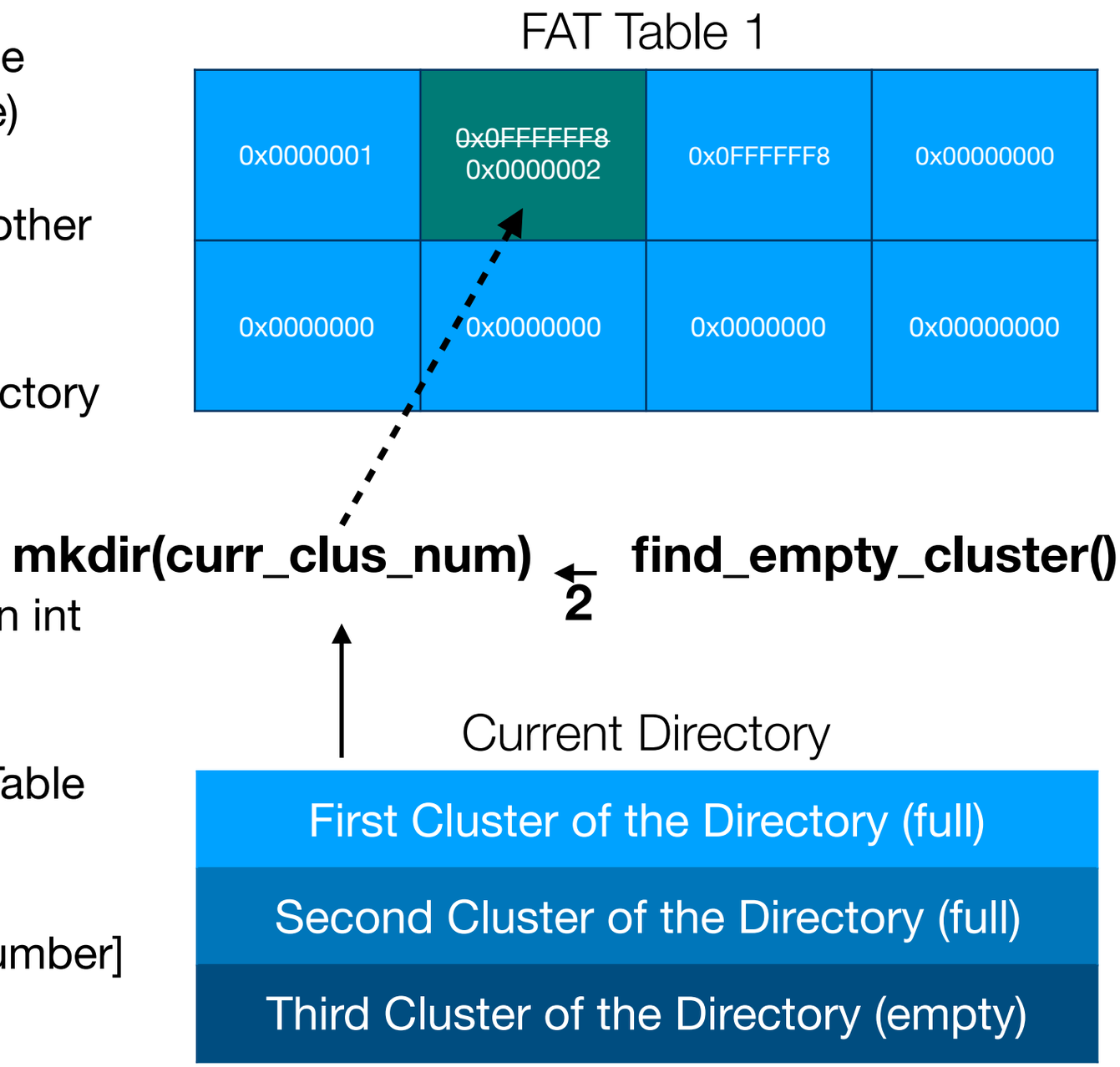
mkdir DIRNAME

- First iterate through the whole of the current directory (just like ls/cd/size)
- Then check if there is space for another directory entry
- If there is no space for another directory entry, call the find_empty_cluster function
- Lets say that the function returns an int value ‘i’
- If ‘i’ is NOT -1, then go to the FAT Table and set FAT[i] = 0x0FFFFFF8
- Now set the FAT[current_cluster_number] == Hex Code of ‘i’



mkdir DIRNAME

- First iterate through the whole of the current directory (just like ls/cd/size)
- Then check if there is space for another directory entry
- If there is no space for another directory entry, call the find_empty_cluster function
- Lets say that the function returns an int value ‘i’
- If ‘i’ is NOT -1, then go to the FAT Table and set FAT[i] = 0x0FFFFFF8
- Now set the FAT[current_cluster_number] == Hex Code of ‘i’



mkdir DIRNAME

- Now, using fwrite, write a directory entry inside the empty directory entry space in the current/new (as is the case) cluster

- Remember to account for the space above (which is present as the 2nd long directory entry)

- You don’t have to write anything in the 2nd long directory entry, as long as you set the short-directory entry fields properly.

- Remember to SET DIR_Attr = 0x10 (as mkdir implies it will be a directory)

mkdir DIRNAME

- Now, using fwrite, write a directory entry inside the empty directory entry space in the current/new (as is the case) cluster

- Remember to account for the space above (which is present as the 2nd long directory entry)

- You don’t have to write anything in the 2nd long directory entry, as long as you set the short-directory entry fields properly.

- Remember to SET DIR_Attr = 0x10 (as mkdir implies it will be a directory)

mkdir DIRNAME

- Now again call the find_empty_cluster function. And this time, the incorporate the returned value into the DIR_Clus fields of the new directory

- empty_cluster_number/0x100 = DIR_ClusHI

- empty_cluster_number%0x100 = DIR_ClusLO

- Also, you need to set other fields (however, not all fields are extremely important, except the ones mentioned here in the slides, and the ones you need for operations)

- Go through the Directory Structure in FATSpec.pdf

- However, you are not finished

mkdir DIRNAME

FAT Table

0x0000001	0x0000002	0x0FFFFFF8	~~0x00000000~~ 0x0FFFFFF8
0x0000000	0x0000000	0x0000000	0x00000000

mkdir(curr_clus_num) ← 3 **find_empty_cluster()**

Current Directory

First Cluster of the Directory (full)

Second Cluster of the Directory (full)

Third Cluster of the Directory (empty)

Current Directory Cluster No. 3

Long Directory Entry for DIRNAME (keep it blank if you want to)

Short Directory Entry for DIRNAME (DIR_Attr = 0x10, DIR_ClusHi = empty_cluster_number/ 0x100 and DIR_ClusLo = empty_cluster_number%0x100)

mkdir DIRNAME

- Now you have an empty cluster
- Go to this empty cluster and write 2 new directory entries
- The “.” and “..” entries are used to link a directory to itself and it’s parent
- Hence the first directory entry (“.”) will have DIR_Cluster = new_directory_cluster_number
- And the second directory entry (“..”) will have DIR_Cluster = parent_directory_cluster_number (in our case, this is current_cluster_number)
- DONE

mkdir DIRNAME

Current Directory

First Cluster of the Directory (full)

Second Cluster of the Directory (full)

Third Cluster of the Directory (empty)

mkdir(curr_clus_num)

Current Directory Cluster No. 3

Long Directory Entry for DIRNAME (keep it blank if you want to)

Short Directory Entry for DIRNAME (DIR_Attr = 0x10, DIR_ClusHi = empty_cluster_number/ 0x100 and DIR_ClusLo = empty_cluster_number%0x100)

New Directory Cluster No. 1

Short Directory Entry for "." DIRClusterNumber =empty_cluster_number, and DIR_Attr = 0x10

Short Directory Entry for ".." DIRClusterNumber =current_cluster_number, and DIR_Attr = 0x10

New Directory

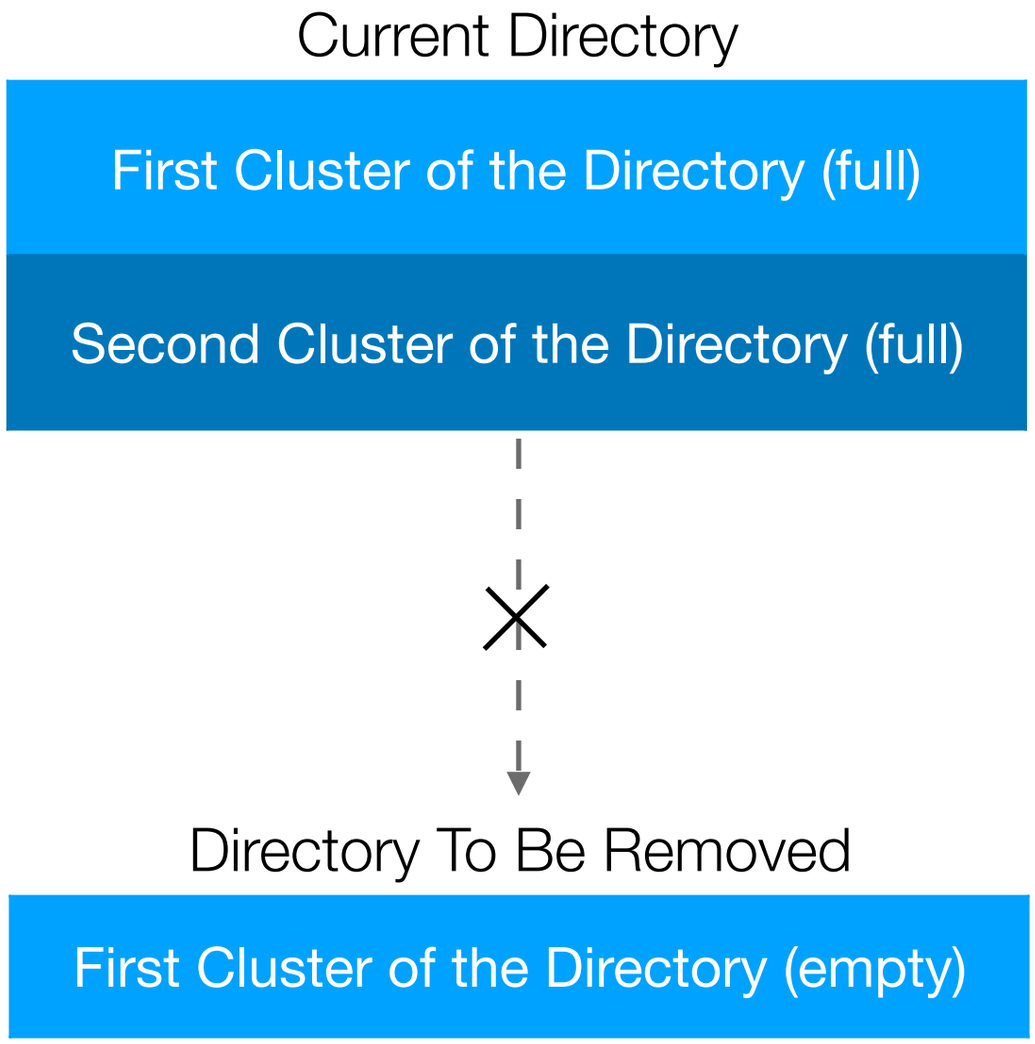
First Cluster of the New Directory

creat DIRNAME

- Almost exactly like mkdir
- Difference is you DON’T set DIR_attr to 0x10.
- You do not need to put the “.” and “..” entries in the new cluster
- DIR_Size will be 0

rmdir DIRNAME

- Removes a directory from the filesystem
- Can only be performed in empty directories
- You need to remove the directory entry in the present directory
- Then you need to erase all the data in the directory cluster
- Finally, unlink the directory and clusters and make FAT Table changes



rmdir DIRNAME

- Again iterate through the whole of the current directory (just like ls/cd/size)
- Check if DIRNAME matches any of the Directory Entry
- On match, go to the directory's first cluster
- As directory is empty, you don't need to think about traversing through to next cluster
- However, CHECK if directory is empty. See if the next directory entry after "." and ".." entries is 0x00 (See FATSpec.pdf Page 23)
- If and only if the next directory entry is 0x00 (NOT 0xE5), then use fwrite to put all 0s in the first 2 entries.(Basically FAT32DirectoryBlock with all values set to 0)

rmdir DIRNAME

Current Directory

First Cluster of the Directory (full)

Second Cluster of the Directory (full)

rmdir(curr_clus_num)

Current Directory Cluster No. 2

DIRNAME Long Entry

DIRNAME Short Entry

DIRNAME Cluster No. 1

~~Short Directory Entry for “.”~~
All Zeroes

~~Short Directory Entry for “..”~~
All Zeroes

DIRNAME

First Cluster of DIRNAME

rmdir DIRNAME

- Now, the DIRNAME_Cluster_Number (you got it from the directory entry of DIRNAME), needs to be unlinked

- Go to FAT[DIRNAME_Cluster_Number] and set it to 0x00000000

- Now you can remove the DIRNAME Directory Entry

rmdir DIRNAME

Current Directory

First Cluster of the Directory (full)

Second Cluster of the Directory (full)

rmdir(curr_clus_num)

Current Directory Cluster No. 2

DIRNAME Long Entry

DIRNAME Short Entry (Has DIR_Cluster_Number)

FAT Table

0x0000001	0x0000002	0x0FFFFFF8	~~0x0FFFFFF8~~ 0x00000000
0x0000000	0x0000000	0x0000000	0x00000000

rmdir DIRNAME

Current Directory

First Cluster of the Directory
(full)

Second Cluster of the Directory
(full)

rmdir(curr_clus_num)

Current Directory Cluster No. 2

~~DIRNAME Long Entry~~
All Zeroes

~~DIRNAME Short Entry (Has DIR_Cluster_Number)~~
All Zeroes

FAT Table

0x0000001	0x0000002	0x0FFFFFF8	0x00000000
0x0000000	0x0000000	0x0000000	0x00000000

rm FILENAME

- A bit more complicated than rmdir.
- We had the flexibility of only having to delete empty directories in rmdir
- For files, we do NOT have that flexibility
- So, what do we do?

rm FILENAME

- For rm, we do the first few steps (in getting to the directory entry) exactly as we did in mkdir.
- The differences diverge when we get to the first cluster of the file
- Here, iterate through the whole file till we have reached that last cluster
- Use a stack to push the cluster numbers (so that the most recently found cluster number is on the top).
- Using fwrite, overwrite all the data in the cluster on the top of the stack to 0s
- Then go to FAT[cluster_on_top_of_stack] and make it = 0
- Pop out the stack. And do the last 2 steps again and again, till stack is empty
- Remove the FILENAME directory entry.

rm FILENAME

Current Directory

First Cluster of the Directory
(full)

Second Cluster of the Directory
(full)

rm(curr_clus_num)

FILENAME Cluster 2

~~FILENAME Data~~
All Zeroes

Current Directory Cluster No. 2

FILENAME Long Entry

FILENAME Short Entry

FILENAME

First Cluster of FILENAME

Second Cluster of FILENAME

second_cluster | first_cluster

rm FILENAME

Current Directory

First Cluster of the Directory
(full)

Second Cluster of the Directory
(full)

rm(curr_clus_num)

FILENAME Cluster 1

~~FILENAME Data~~
All Zeroes

Current Directory Cluster No. 2

FILENAME Long Entry

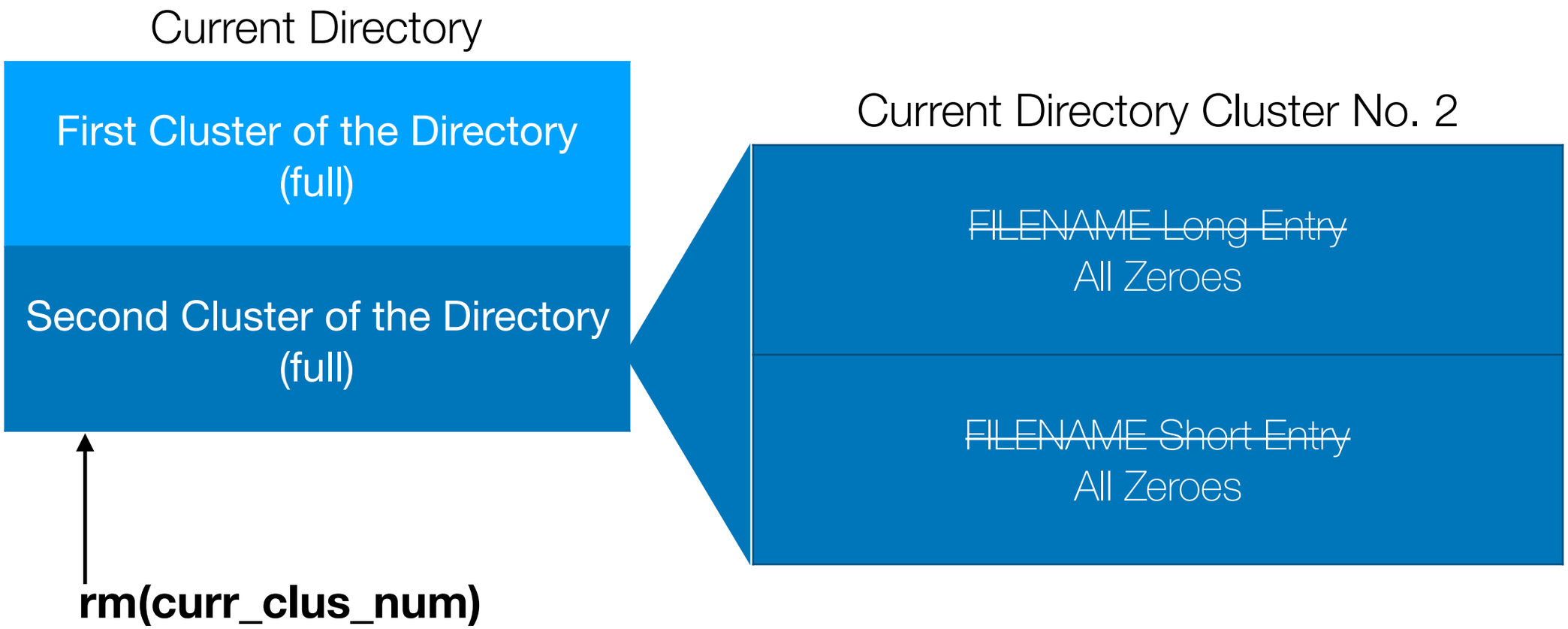
FILENAME Short Entry

FILENAME

First Cluster of FILENAME

first_cluster

rm FILENAME


Current Directory
First Cluster of the Directory (full)
Second Cluster of the Directory (full)
rm(curr_clus_num)
Current Directory Cluster No. 2
~~FILENAME Long Entry~~
All Zeroes
~~FILENAME Short Entry~~
All Zeroes

rm FILENAME

FAT Table On Slide 26

0x0000001	0x0000002	0x0FFFFFF8	0x00000004
~~0x0FFFFFF8~~ 0x00000000	0x0000000	0x0000000	0x00000000

FAT Table On Slide 27

0x0000001	0x0000002	0x0FFFFFF8	~~0x00000004~~ 0x00000000
0x00000000	0x0000000	0x0000000	0x00000000

Project 3: To Do

- Everyone knows you guys haven’t started your project. Please do.
- Start implementing these 4 functions
- Remember you have to use rb+ to modify the image properly
- Robust in this context means, after mkdir, or creat or rmdir, rm, you can use ls, cd, size as well as the other commands in the project properly and they will give correct output
- DO NOT OVERWRITE EVERYTHING USING WB or WB+ to open the file
- Please sign your groups up. It is getting way too late. I still have 30+ unassigned people

MAY THE CODE

BE WITH YOU