Project 3: FAT32 File System
Part 2

Modifying the FAT32 Image File

- **Whenever opening the image file, make sure to use rb+ for the mode**
  - b -> open file as binary file
  - r+ -> allows reading and updating
  - Opening with wb or wb+ 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 rm’ing the old one and either untar’ing the original tar file or copying a clean version from another folder into the working directory)

Terminology

- Reminder: in the FAT32 setting, a folder's "directory entries" (DIR entries) refer to the entries which represent the files and subdirectories within the directory
  - Distinguish between files and directories by using the DIR_Attr field
  - (DIR_Attr = 0x10 means the entry refers to a directory/folder)

Function: creat FILENAME

- To create a new file, need to find a place to put it
- Iterate through FAT table and look for the 0x0 entry which marks an empty cluster -> 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: mkdir DIRNAME

- Very similar to creat
- Need to set DIR_Attr = 0x10, also set the other directory related fields specified in FATspec

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: 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

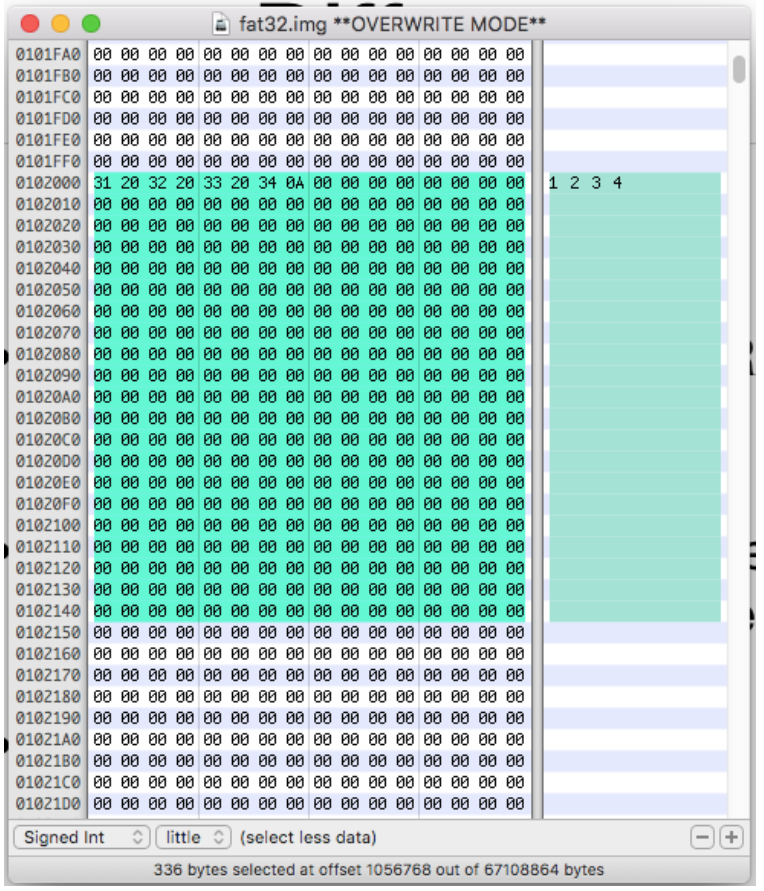
- After checking for valid FILENAME and MODE arguments,
  - Get FILENAME’s first_cluster_number (remember you must combine the HI and LO)
  - Check if the file is open by searching through the linked list
  - If first_cluster_number is in the linked list, the file is already open, return an error
  - 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, delete the entry
  - If it isn’t, print an error

Function: read FILENAME

- Remember that data stored in a file is stored in RAW format inside the FAT32 image file
- This means you must print the whole data inside the file as a string
- The example on the right shows a text file with the contents “1 2 3 4”



Function: read FILENAME

- To read in the file contents,
  - First, check that the read the file is open and that it was opened with MODE r, rw, or wr
- Iterate through all clusters of the file printing the contents at each cluster
  - Read entire cluster into char array and print contents
  - You may want to use fread and puts functions

Function: read FILENAME OFFSET SIZE

- Given a FILENAME, read SIZE bytes of the FILENAME starting at OFFSET
- At *each* cluster, need to calculate which bytes to read based on SIZE and OFFSET!
- Edge cases: OFFSET > sizeof(FILENAME)
  - Print error
- SIZE > sizeof(FILENAME),
  - Print entire file
- OFFSET + SIZE > sizeof(FILENAME)
  - Prints sizeof(FILENAME) - OFFSET bytes (from offset to end of file)

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:
  - OFFSET / bytes_per_sec = cluster offset
  - OFFSET % bytes_per_sec = byte offset within cluster

Function: write FILENAME OFFSET SIZE STRING

- Check MODE to ensure file was opened with write permissions
- Initialize a char array of SIZE bytes
- Need to write STRING to the file using fwrite
- Once again, the file might span multiple clusters
  - You may need to allocate new clusters or remove existing clusters

Function: write FILENAME OFFSET SIZE STRING

- Edge cases:
- FILENAME is a large file (say 1024 bytes), and you are supposed to write 20 bytes at OFFSET 100.
  - Overwrite the 20 bytes since OFFSET 100, and then 0 out the rest of the contents of the file
  - If there are clusters after the current cluster, unlink them (similar to rm and rmdir)
- OFFSET > sizeof(FILENAME)
  - Print error
- OFFSET < sizeof(FILENAME) but OFFSET + SIZE > sizeof(FILENAME)
  - Write the STRING on the (OFFSET/sector_size) cluster at OFFSET%sector_size point of the cluster, overwrite whatever is there
  - 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 STRING < SIZE, pad 0s (ASCII value 0) to the rest of the char array
  - If STRING > SIZE, just use the first SIZE bytes of STRING, put them in the char array and write to file

When in doubt, check the FATspec!