week3.md

School of Engineering and Applied Science, Ahmedabad University

Operating System Lab - CSE 341

Faculty: *Prof. Mansukh Savaliya*

Project Progress Report: *Week 2*

Group Number: *3*

Group Members:

Name	Enrolment Number
Yashil Depani	AU1841005
Vidish Joshi	AU1841019
Manav Patel	AU1841037

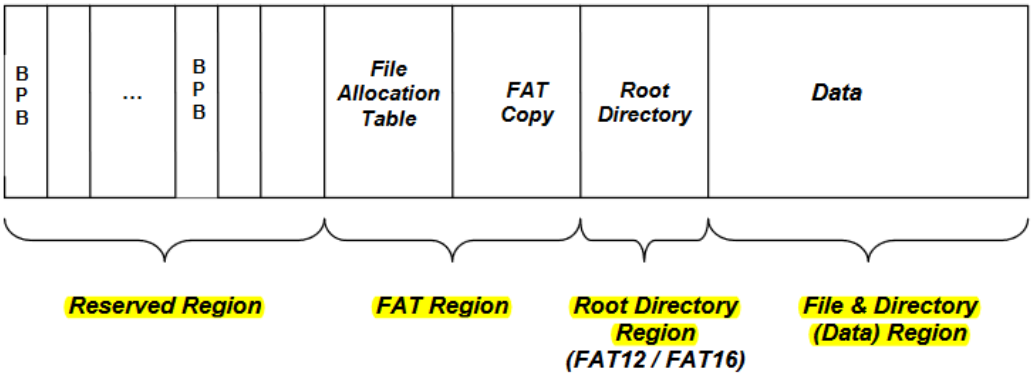
OSProject - toaruOS

An operating systems project on understanding and solving issues for the open source OS - toaruOS.

Index

Working on FAT File system

We want to write a code that can operate on FAT File system such as listing files, directories, opening directories, files and also editing them. In essence, we want the Operating System to be able to operate on FAT32 fs.

> The below figure illustrates the four regions in a volume formatted FAT:
>
> 
>

As we can see from the images, a FAT file system is composed of four basic regions, which are laid out in this order on the volume:

```
1 - Reserved Region
2 - FAT Region
```

```
3 – Root Directory Region (doesn’t exist on FAT32 volumes)
4 – File and Directory Data Region
```

The image has been taken reference from Microsoft Documnetation of FAT32.

This documentation contains important and required information for FAT file system. This information provides us the necessary relationships, equations and formulae to be able to access the entries in the above seen 4 regions. After understanding this we started our coding part through which we would operate on a File Structure following FAT32 System.

To give an example, these are 4 equations depicting relationships between 4 parameters of a FAT fs:

```
fat_begin_lba = Partition_LBA_Begin + Number_of_Reserved_Sectors
cluster_begin_lba = Partition_LBA_Begin + Number_of_Reserved_Sectors + (Number_of_FATs * Sectors_Per_FAT)
sectors_per_cluster = BPB_SecPerClus
root_dir_first_cluster = BPB_RootClus
```

Source: https://www.pjrc.com/tech/8051/ide/fat32.html

The definitions for the parameters used in above equations can also be found in the Microsoft Documentation and last week's report, some examples of which can be found in the last week's report.

We can use such relationships while coding to access the directories, files and directory structure with to create a functioning file system that supports the standard file system operations such as creating a new file, a new directory, removing them, get information, etc.

For this we require a directory structure following FAT32 System and so we started working on `.img` files. This .img files should follow FAT32 File System. Now, the question is. how to create/get such files?

Creating/Getting IMG File

Creating `.img` files following FAT32 File system is not that straight forward. We surfed a lot on this for 2-3 days. One cannot simply convert a directory into `.img` file. And to create a `.img` file of a directory structure that follows FAT32 fs seems even more difficult.

We didn't get any slightest hint on the internet regarding how to create/convert such files; neither in form of readily-available tools nor in the form of working codes. But we have managed to get some `.img` files online. We tried many methods available online in the form of codes and/or OS based steps to create `.img` files with supported FAT32 fs. Here are some of the famous sites out of the several we tried to use:

Site: https://superuser.com/questions/668485/creating-a-fat-file-system-and-save-it-into-a-file-in-gnu-linux

Site: https://unix.stackexchange.com/questions/37072/linux-fat32-and-etc-fstab

Site: https://sathyasays.com/2008/11/01/how-to-mount-fat32ntfs-partitions-with-read-and-write-support-in-linux-using-command-line/

Many of these websites did not work at all *for us* whereas for some it did get converted but the raw structure was not following the above image.

Having not being able to successfully create `.img` files with desired specifications, we then searched for already creates `.img` files supporting FAT32 file system. Here also we faced issues as we did find image files but very few of them supported fat file system. Following is the link to a GitHub repo we found containing 6 FAT32 image files.

Link: https://github.com/procount/fat32images

This is link to repo containing all our creates, tried or found image files:

Link: https://github.com/PatelManav/imagefiles

Why is all this needed for us? For writing the code , we strictly need to stick with on how Microsoft defines the files structure. Which bytes represent what regions and how to access it. Example of how the sectors, clusters and regions are defined, how much bytes each field requires are present in last week's report referenced from the Microsoft documentation. `.img` files allows us to replicate the FAT32 fs in our current OS and in the OS's file system without having to create a partition for it in our device's memory. Due to all these, we agreed on going with `.img` files which we have managed from internet for time being, while still looking for ways to create `.img` files by ourselves (creating our own image file will allow us to create directory structure of our liking and test with several test cases).

The CODE

We started the coding part as well.

The following image shows the defined parameters that are required to maintain and access a FAT file system from a drive or an .img file. These parameters are part of the *BIOS Parameters Block* or the *BPB* in the reserved region of the FAT32 fs.

```
char BS_OEMName[8];
int16_t BPB_BytesPerSec;//The amount of bytes in each sector of the fat32 file image
int8_t BPB_SecPerClus;//The amount of sectors per cluster of the fat32 file image
int16_t BPB_RsvdSecCnt;//Amount of reserved sectors in the fat32 image
int8_t BPB_NumFATs;
int16_t BPB_RootEntCnt; //Root entry count
int32_t BPB_FATSz32;
int32_t BPB_RootClus;//Rootcluster location in the fat32 image

int32_t RootDirSectors = 0; //Amount of root directory sectors
int32_t FirstDataSector = 0;//Where the first data sector exists in the fat32 file image.
int32_t FirstSectorofCluster = 0;//First sector of the data cluster exists at point 0 in the fat32 file image.

int32_t currentDirectory;//Current working directory
char formattedDirectory[12];//String to contain the fully formatted string
char BPB_Volume[11];//String to store the volume of the fat32 file image
```

The following image contains code block containing the basic I/O required to input commands and process them by calling necessary functions which contain the logic to operate on the DATA region of the file system through the FAT table.

```
void getInput()
{
    printf("CMD> ");

    memset(cmd_str, '\0', MAX_COMMAND_SIZE);

    while (!fgets(cmd_str, MAX_COMMAND_SIZE, stdin))
        ;

    int token_count = 0;

    char *arg_ptr;

    char *working_str = strdup(cmd_str);

    char *working_root = working_str;

    memset(&token, '\0', MAX_NUM_ARGUMENTS);

    memset(&token, '\0', sizeof(MAX_NUM_ARGUMENTS));
    while (((arg_ptr = strsep(&working_str, WHITESPACE)) != NULL) && (token_count < MAX_NUM_ARGUMENTS))
    {
        token[token_count] = strndup(arg_ptr, MAX_COMMAND_SIZE);
        if (strlen(token[token_count]) == 0)
        {
            token[token_count] = NULL;
        }
        token_count++;
    }
    free(working_root);
}
```

However, this code is not complete and also contains some errors. We are working to resolve them. We also made some progress for the functions in the code that will handle the FAT32 regions as mentioned above. At present, we are able to open *some* of the `.img` files successfully.

Plan for next week

Going ahead with code. Successfully implement the code section that allows us to *open* and *close* the `.img` files. Open them and get the data present in it in token arrays for, i.e read what is inside the directory structure. if possible implement command such as `ls` which would parse through the FAT table, go to the memory addresses pointed in the table entries and output the subdirectories and files present inside the root directory.

Contribution

Reading/understanding the Microsoft documentation - All 3 members

Writing the code:

Writing the part in code dealing with defining parameters and IO ops: All 3 members

Writing the part to handle the FAT tables and addresses - All 3 members

Finding tools/codes to create `.img` files - All 3 members

Finding created FAT32 `.img` files - All 3 members

Working around creating and manipulating .img file - All 3 members