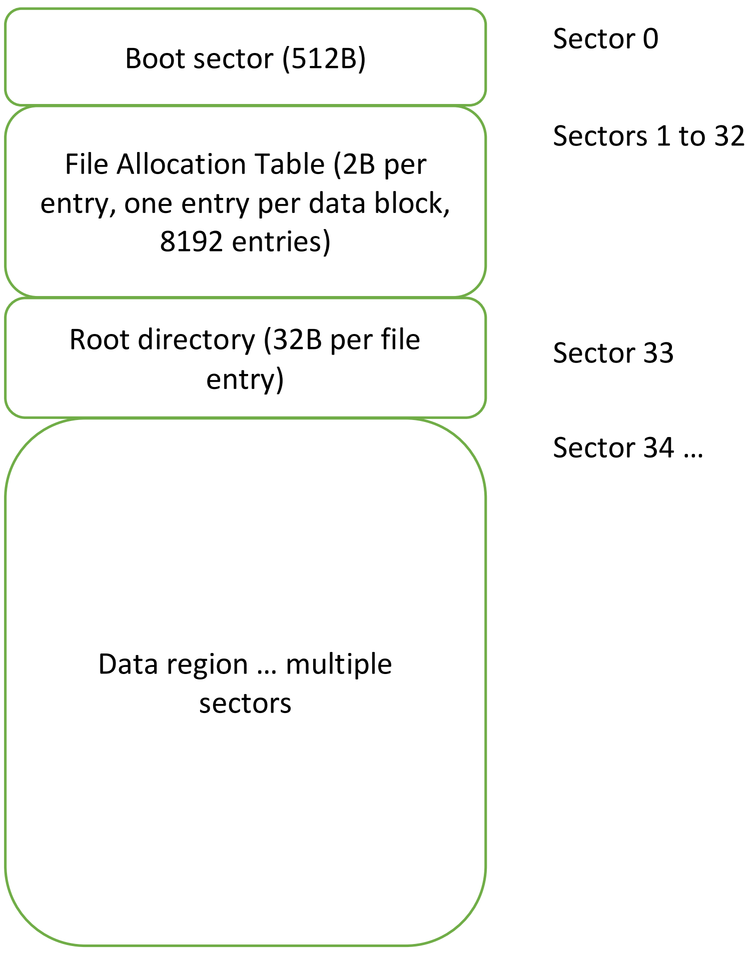
Mon 27 Feb 2018
Operating Systems (H) Lab
The FAT File System

Learning Objectives

- To understand the structure underlying the FAT file system
- To create a blank FAT file system image
- To programmatically update a FAT file system with new files
- To compare the relative merits of C and Python for systems programming

Task 1 – Create a Blank FAT file system image

Open the create_fatfs.py script. This is incomplete python code that you need to fix up, in order to generate a FAT image file called example.img. First, here is a schematic diagram of the FAT file system.


Boot sector (512B)
Sector 0
File Allocation Table (2B per entry, one entry per data block, 8192 entries)
Sectors 1 to 32
Root directory (32B per file entry)
Sector 33
Sector 34 …
Data region … multiple sectors

We want to create a FAT16 image, with 8192 sectors of data. Each sector is 512B in size, so this will create a 4MB disk image. The Python script already generates the correct data for the boot sector.

(1) You need to add in the FAT table entries (8192 of them) immediately after the boot sector. The first four bytes of the FAT table are 0xf8 0xff 0xff 0xff – to occupy the first two entries of the table. All subsequent 8190 entries are 16-bit 0 values.

Use code like **`f.write( bytearray([0x00, 0x00]) )`**
to generate the FAT entries. However note it is cheaper to generate and write one large bytearray rather than lots of little ones.

(2) Now you need to write the root directory. How many files are allowed in the statically-allocated root directory? Check the boot sector data to see the answer. Each root directory entry occupies 32 bytes. For now, set them all to 0. So you need [0] * (32 * NUM_ROOT_ENTRIES) and write this to the file.

(3) Now you need to allocate the data region of the file. The image should have 8192 sectors overall, but 1 is taken up by the boot sector, and the root directory occupies some sectors too. (Don't count the FAT sectors – they are 'invisible'). So how many sectors are left for the data region? Fill these with all 0 values. Remember that each sector is 512 bytes long.

(4) Now you can run your Python script and create a FAT file image. You should be able to mount this (if you have root permissions – perhaps with sudo mount –t vfat –o loop example.img /mnt) or dd it to a USB disk – perhaps with dd if=example.img of=/dev/sdb - or GNU mtools).

Task 2 – Write a file to the FAT FS using a C program

Now look at the add_file.c – this program reads in the example.img disk image. You need to look at the two TODO comments. (1) You need to add the directory metadata for the file that is being created. (2) You need to add the FAT entry for the file that is being created.

Since the file is a single sector file (shorter than 512 bytes) then you only need one FAT entry. What would you do if you wanted to write a longer file, that spans multiple sectors?