CI583: Data Structures and Operating Systems
File systems

Outline

Overview

File systems provide a simple means of permanent storage.

The design decisions taken when producing a file system are of crucial importance to the performance of the overall system.

We will begin by analysing a couple of early file systems (useful to illustrate the basic ideas in a simple way) before moving on to modern improvements.

Overview

An ideal file system should satisfy these criteria:

1. Easy to use: the abstractions used (file, directory, permissions, etc) should be easy to understand,
2. High performance: fast access to data, efficient use of storage space,
3. Permanence: the system should be reliable (files not easily corrupted, for instance) and able to recover well from failures,
4. Security: users should have the right level of control over who can see their data and what they can do with it.

FAT

We start with the legacy FAT (File Allocation Table) system since it uses a design which is very easy to describe (and implement).

FAT was first designed in the late 70s for use on floppy discs. It satisfies our first criteria (ease of use) but none of the others!

FAT

Surprisingly enough, however, it is still in use in settings such as digital cameras and solid-state memory drives.

It is a good choice for these devices because it is supported by every OS.

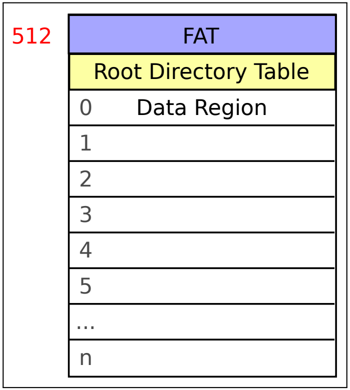
FAT

For now, we will consider a disk to be *a single region of storage* divided into blocks (also called clusters) of equal size, such as 512 bytes.

This simplifies the discussion a lot, but when we consider file system performance we need to take the storage medium into account.

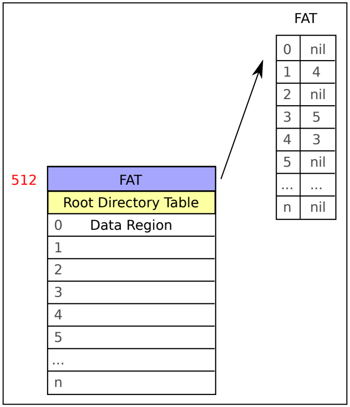
FAT

After a boot block, which contains the instructions to load the operating system's binary image into memory, the first block in a FAT drive is occupied by the File Allocation Table.



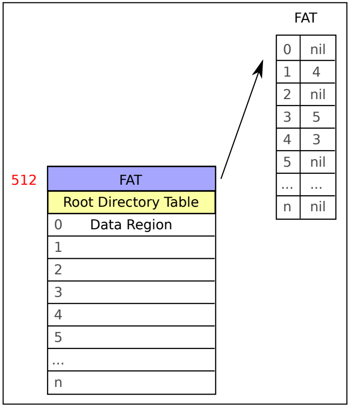
FAT

We can think of the FAT as an array with an entry for each block in the data region. Entry 0 in the FAT corresponds to data block 0 and so on (as far as we're concerned).



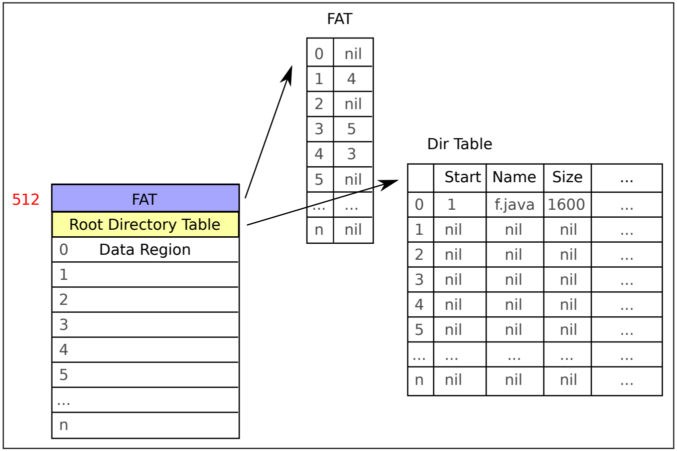
FAT

The FAT entry for each data block gives its successor, creating a linked list. The successor may be nil.



FAT

The root directory table lists the files in the top-level directory of the file system. Amongst other things, this tells us the starting block for each file. Where are the blocks for `f.java`?



FAT

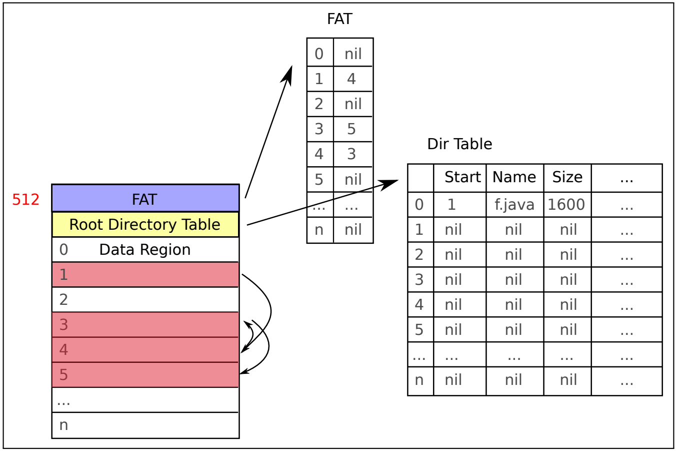
0	nil
1	4
2	nil
3	5
4	3
5	nil
...	...
n	nil

Dir Table

	Start	Name	Size	...
0	1	f.java	1600	...
1	nil	nil	nil	...
2	nil	nil	nil	...
3	nil	nil	nil	...
4	nil	nil	nil	...
5	nil	nil	nil	...
...	...	...	...	...
n	nil	nil	nil	...

FAT

The root directory table lists the files in the top-level directory of the file system. Amongst other things, this tells us the starting block for each file. Where are the blocks for `f.java`?



FAT

0	nil
1	4
2	nil
3	5
4	3
5	nil
...	...
n	nil

Dir Table

	Start	Name	Size	...
0	1	f.java	1600	...
1	nil	nil	nil	...
2	nil	nil	nil	...
3	nil	nil	nil	...
4	nil	nil	nil	...
5	nil	nil	nil	...
...	...	...	...	...
n	nil	nil	nil	...

FAT

The size of the root directory is allocated when the disk is formatted, so there is a fixed limit on the number of files that can appear in the root directory.

Sub-directories are just special files which list the files they contain, similarly to the root directory table. The difference from normal files is that they can't be modified directly by users.

File systems don't get much simpler than FAT and that is actually a benefit: very well-supported, easy to implement and relatively robust (thanks to having an extra FAT on floppy discs).

FAT

The drawbacks are that FAT is way too slow for modern use, especially for random access within files. If we read some data starting in the middle of the file, we may have to follow a long series of pointers in the FAT.

FAT is inefficient for storing sparse files – a file that consists of 10kB of 0s takes up as much space as one full of varied data.

FAT imposes severe limits on the length of filenames, the number of files in a directory and the size of disks. It is prone to internal and external fragmentation.