File Allocation Table (FAT)

Learning Objectives

- Be able to describe the FAT on-disk layout and data structures in enough detail to begin to implement them in a basic FUSE file system (more on FUSE later)
- Be able to describe the scalability and performance limitations of the FAT design
- Be able to relate the design to the computing context of the time
- Be ready to talk about more complex file system designs that overcome some of the FAT limitations

Overview of Data Structures

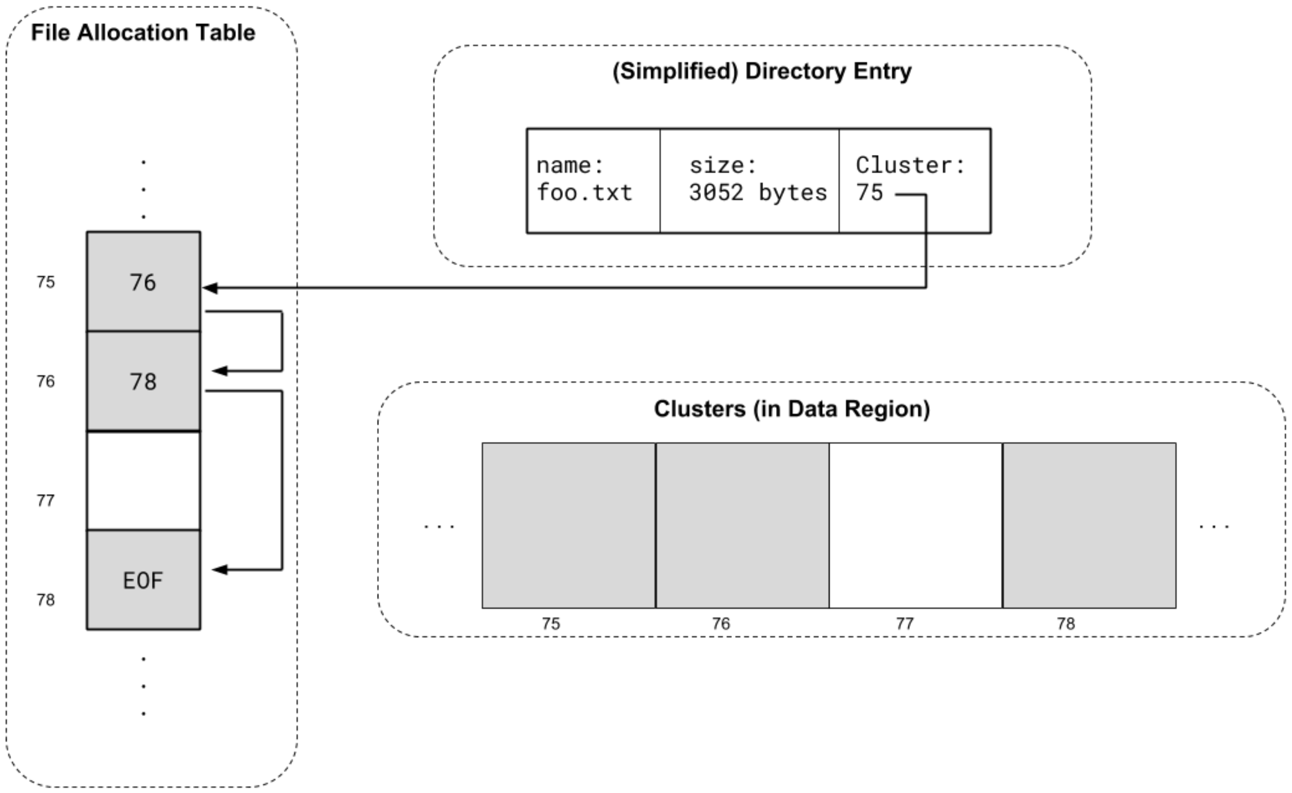
Directory entries (one per file) store a file's name, its size, and the address of the first block with the file's contents (a pointer to the *cluster*).

- **directory entries** have a fixed size (32 bytes)
- key fields: file size, address of the first cluster, {c|a|m}time, name
- if first byte is `0xe5` , then the directory entry is not allocated
- directories are just files with a special format: stored in clusters in data region as a table of directory entries
- FAT has no concept of an inum: the way to address a directory entry is by it's full pathname
- directory entries have an "*attributes*" field that is a series of flags:
  - *directory* (does this directory entry refer to a directory?)
  - *long file name* (does this directory entry use the long file name layout VFAT)
  - *volumne label*: one file per FAT file system should have the volume attribute set
  - *read only*: file can't be written/changed
  - *hidden*: files with this attribute aren't listed by default
  - *system*: this file is a system file
  - *archive*: set when a file is modified (created or written to) so that system can identify files that have changed since the last backup

Clusters a cluster is the FAT allocation and addressing unit

- a power-of-2-sized group of consecutive sectors, with a maximum cluster size of 32KiB.
- clusters can form a chain, which can be followed by tracing through the FAT
- directory entries contain a pointer to the first cluster in a file
- The FAT entry for an allocated cluster contains the address of the next cluster in the chain (like a linked list), with the exception of the FAT entry for the last cluster in a cluster chain, which contains the EOF marker

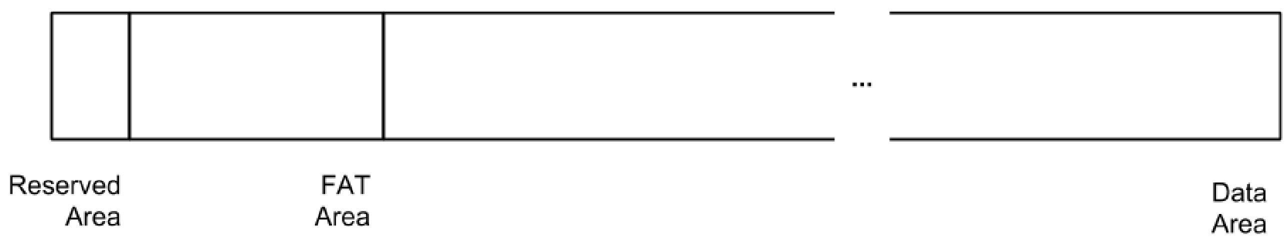
FAT Data Structure Example



- The contents of file `foo.txt` begin in cluster 75. The entry in the FAT at index 75 shows that the next cluster is 76, which then takes us to cluster 78, which is the end-of-file. Cluster 77 is not allocated.

FAT Disk Layout

FAT statically partitions the LBA space into 3 regions: the reserved area, FAT area, and Data area. This static partitioning lets us use math to directly index to specific data structures (specific FAT entries, the first section in cluster `X`, etc.)



FAT File System On-disk Layout: Statically partition the LBA space

Reserved area:

- The 1st (and 6th) sector on the partition store (copies of) the *boot sector*. Interesting/notable boot sector contents include:
  - locations/sizes of the reserved/FAT/Data areas. These can be derived from fields stored in the boot sector.
  - The starting address of the root directory's cluster

FAT Area: (File Allocation Table Area)

- the FAT has one table entry for each cluster
  - If the table entry is 0, cluster is unallocated
  - If the table entry is a special value (version dependent), cluster is damaged and should not be used
  - All other values indicate that the cluster is in use

Data area: (Clusters of sectors allocated to store file contents)

- Data is stored in clusters, which are fixed-size contiguous regions of sectors
- Clusters are allocated on demand to hold file data, and entries are added to the FAT to connect related clusters

1. *Question*: Knowing what we know about HDDs, how might the cluster size affect file system performance?
2. *Question*: Why would you choose a small cluster size? A large cluster size?

Cluster (De)Allocation

There are a few ways we might want to allocate new clusters. When you implement FAT, you may choose among them depending on your optimization goals. The simplest are *first available cluster*, and *next available cluster*.

- Next available cluster: keep track of the most recently allocated cluster. When allocating, start your search at that index, and scan the FAT until you find a cluster with value 0 (an unallocated cluster entry).
- To deallocate a cluster: simply set the FAT cluster entry to value 0.

1. *Question*: When deleting a cluster, should the FAT file system clear the cluster's contents, or just the cluster's allocation status (set its metadata entry in the FAT to unallocated)?

Long file Names

FAT directory entries use an "8.3" naming structure: 8 characters in the file's name and 3 characters in the file's extension (e.g., `billfile.txt`). File names longer than that must use a special naming convention to tie together directory entries.

- If a file name is longer than 8.3, or it uses special characters, a long-file-name (LFN) directory entry is added. Files with a LFN entry also have a short-file-name directory entry (SFN), which contains the normal directory entry entities like size, {c|m|a} time, first cluster's address, etc.
- LFN entries use a special attribute value to indicate that the are not SFN directory entries, and are chained together in reverse order, ending with an SFN. Below is a LFN FAT entry's layout

Bytes	**Contents**
0	Sequence number
1-10	Characters 1-5 of name
11	File attributes (`0x0F` special value to indicate this is an LFN entry)
12	Reserved
13	Checksum
14-25	Characters 6-11 of name
26-27	Reserved
28-31	Characters 12-13 of name

Below is an example taken from Brian Carrier's "File System Forensic Analysis" textbook, page 240, figure 9.15. The example shows directory entries for are three files: one has a long name, one has a short name, and one has been deleted. (Checksums for LFN directory entries are calculated based on the SFN, but I chose to omit them from this example for clarity.)

Attr: (normal file)	Name: RESUME-1.RTF	Cluster 9
Attributes: `0x0F` (LFN)	SeqNum: 2	Name.rtf
Attributes: `0x0F` (LFN)	SeqNum: 1	My Long File
Attributes: (normal file)	Name: MYLONG~1.rtf	Cluster 26
Attributes: (normal file)	Name: _ILE6.txt	Cluster 48

Hand-in Question

The FAT file system allocates data in units called clusters. What data structure is used to traverse the clusters that comprise a given file, and what is the big-O performance (in terms of file size)?