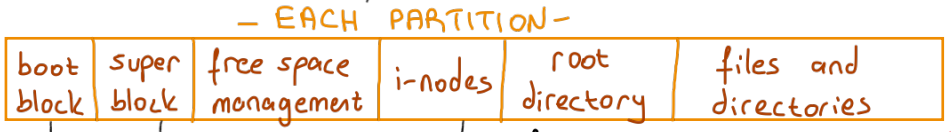
file system implementation

* each partiton can have different file system & os

master boot record
↳ first sector
↳ loads boot loaders in partions

↳ outlines and describes the partitions

– EACH PARTITION –



↳ info about files (not in FAT)

loads OS in kernel space

↳ key info about file system which is read into memory at boot time

FAT (file allocation table)

- window's file system
- linked list implemented on an array
- stores free block list and file linkage

↳ first sector
↳ vol name, # sectors, bytes, clusters...

optional copies of FAT1

↳ FAT12 and FAT16 only

	per entry	# of entries	one entry in bytes
FAT12 =	12 bits →	2^{12}	. $2^{(16\text{ bits})} = 2^{13}$ byte = 8 KB →
FAT16 =	16 bits →	2^{16}	. $2^{(16\text{ bits})} = 2^{17}$ byte = 128 KB
FAT32 =	28 bits →	2^{28}	. $4^{(32\text{ bits})} = 2^{30}$ byte = 1 GB

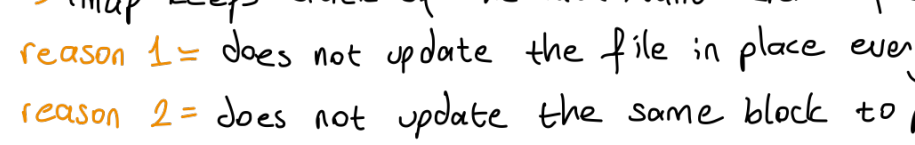
max volume for 512B = 2^9 byte cluster
$2^{12} \cdot 2^9 = 2^{21} = 2$MB

FAT directory entries = name + attribute (time, start, size...) + disk block id → soft link for the pointers to same file

metadata

- FAT table entries are initially zero.
- -1 end of list (file A: 3→5→2→-1)

UFS (Unix File System)



file system meta data

– EACH CYLINDER GROUP –

- number of inodes and data blocks are fixed at file system creation

inode = each file has one, fits in a single block, all file attributes + pointers to data blocks

↳ data block pointers = direct, indirect, double_indirect, triple-indirect
for small data ↳ they point to an array of data block pointers

↳ number of hard links = more than one directory can contain and point to same file, so its inode, only when the number is zero, delete the file

example⇒
block size = 4K
block pointers = 4 byte
direct blocks = 12

direct = 12 block → 12

indirect = 1K → 1K; 4K/4 = 1K; 1K = 1024 block pointers

double = 1K * 1K ⇒ 1M blocks

triple = 1K * 1K * 1K ⇒ 1G blocks

max file

* uses bitmap for empty blocks

file = inode + inode blocks + data blocks

directory entries = inode number + record length (whole) + name length + name + space
↳ basically inode number + name (no attributes) → hard link for the pointers

log structured file system-LSFS

↳ other file systems update metadata frequently

↳ Lsfs, do not overwrite blocks, instead each new write is appended at the end of the log segment

↳ imap keeps track of the last/valid version of the inodes

reason 1 = does not update the file in place every time (time consuming)

reason 2 = does not update the same block to prevent disk damage