Discussing Unix v6 file system design principles

assign2 is variant of it

Filepath resolution - v. similar to resolving domain name to IP addr

File system: contiguous stretch of memory to store info

like RAM!

info in RAM is byte-addressable: must read 8 bits at a time

default unit of memory in hard drives are sectors (sector addressable)

~512 bytes (not always)

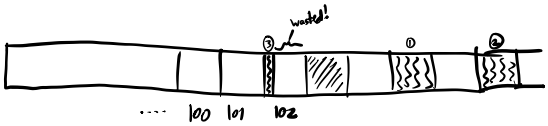
drive exports an API (hardware API) (software abstraction)

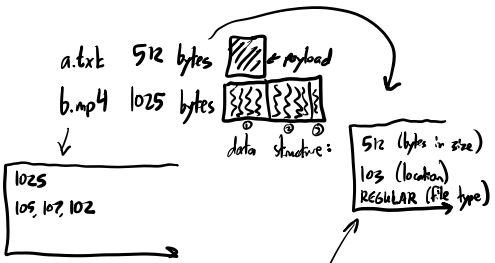
need to take this system and "invent" document storage

FS often invents this concept of "block" (in software, not hardware)

sectors are physical (hardware) storage units

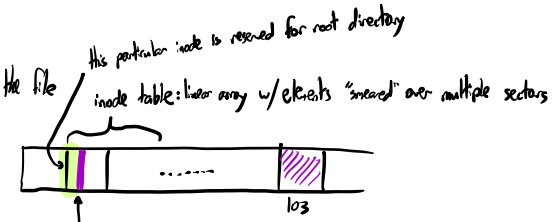
sometimes are the same size





- these are supposed to be fixed-size in a data structure... what happens when list of used sectors gets arbitrarily long?
- where do these get stored?
- how do we account for the name of the file

what we really need...




```
struct inode {
  size_t size;
  ushorts blocknums[8];
  byte type;
}
```

arbitrary

a.txt:
512
103
REGULAR

32 bytes, so fits "smug" in a sector w/ other inodes (like books on a shelf)

want these inodes as small as possible to live in persistent storage on hard drive

have to find inodes to find the data payloads

Friday: tackle large file problem (that "8" above limits us to size 4096 files)

support pathnames, too!

↳ but not every blocknum points directly to the data payload

...point to a sector that stores more block nums!