11/17/23

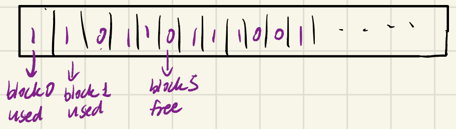
File System Implementation

-> need to track block usage on disk

-> block bitmap (stored on disk)

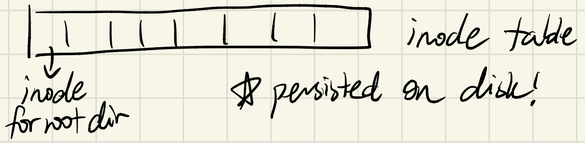
block size is configurable

-> can be 1 sector, 8 sectors or more



-> need to track metadata

-> reserved space for metadata (inode, file header, file record)



☆ persisted on disk!

☆ inode should fit within one unit of atomic write (sector / page)

-> superblock (metadata of the filesystem) ☆ stored at known loc on disk

-> loc of bitmap blocks

-> loc of inode table

Data Layout

☆ assume block size = 1 sector

☆ bytes ⇒ blocks conversion
(r/w offsets & size) ⟶ blocks

-> contiguous : tracks starting data blocks & # of blocks data takes up

example: 100 2

means that block 100 holds the first 512 bytes of data, block 101 holds the next 512 bytes

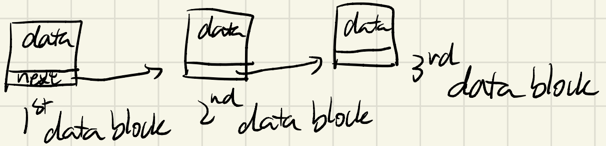
☆ How do you grow a file?

ith byte → data block

$= i / 512$ (block size) + start blk

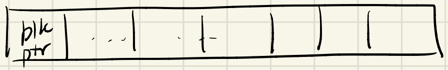
☆ block = allocation unit, a file with 600 bytes also takes up 2 data blocks

-> linked: metadata tracks first block and last block



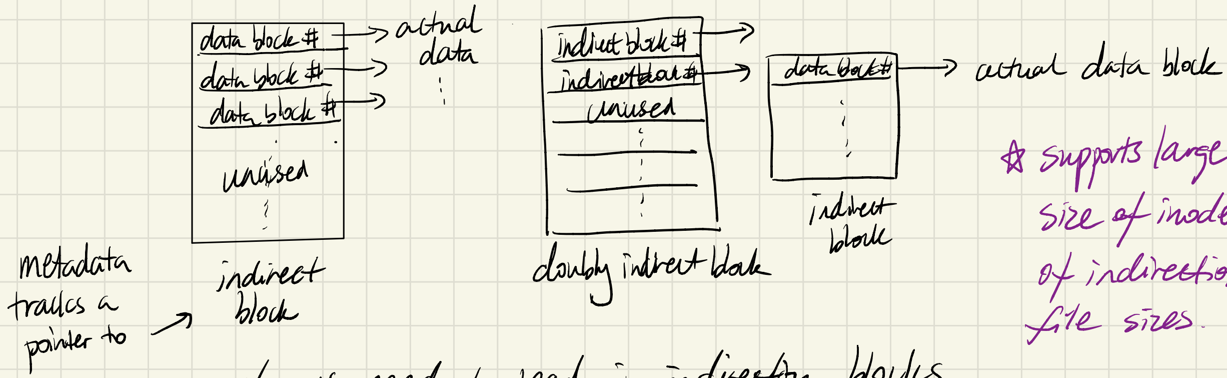
☆ data blocks no longer need to be allocated in contiguous chunk, can grow easily! but locating specific data blocks can take much longer (pointer chasing)

-> array: store an array of data block #



can quickly locate data block, what happen if the array fills up?

-> indexed/indirection: use a block separate block to store array of data block pointer



☆ supports large file w/out expanding size of inode, different levels of indirection supports different file sizes.

always need to read in indirection blocks to locate data blocks => poor performance for reading small files

-> combined approaches
-> multilevel indexed (FFS): 12 direct pointer, 1 indirect, 1 doubly indirect, 1 triplely indirect

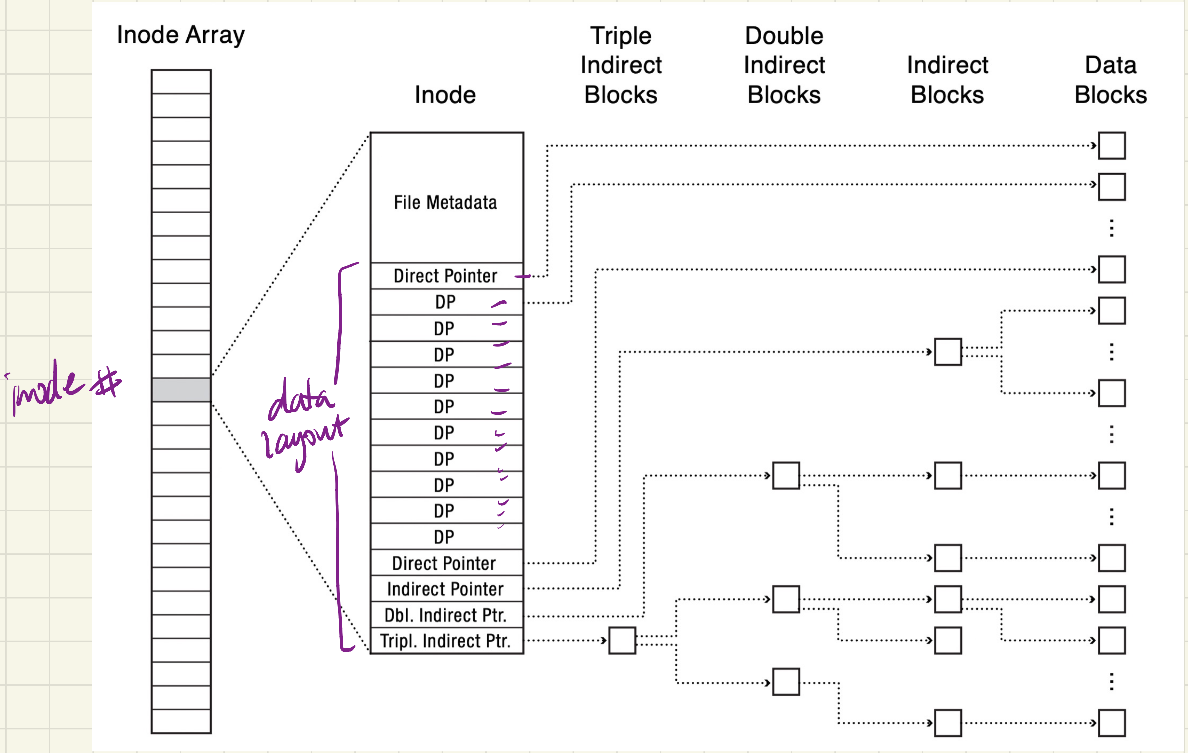
-> extents: tracks array/linked list of contiguous blocks
each extent tracks a contiguous section of blocks
extents sizes varies

Fast File System (FFS)

→ designed for disk

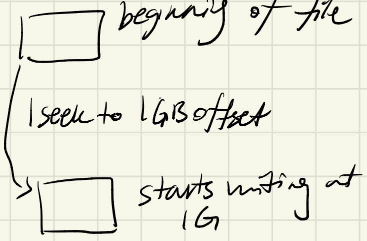
→ when configured w/ block size of 4kB & 32 bit disk address :

direct ptr → 4kB of data
indirect ptr → 4MB of data
double indirect → 4GB of data
triple indirect → 4TB of data



☆ only used entries have valid block ptr

lseek = POSIX API that lets you set offset past file size



gap w/ file