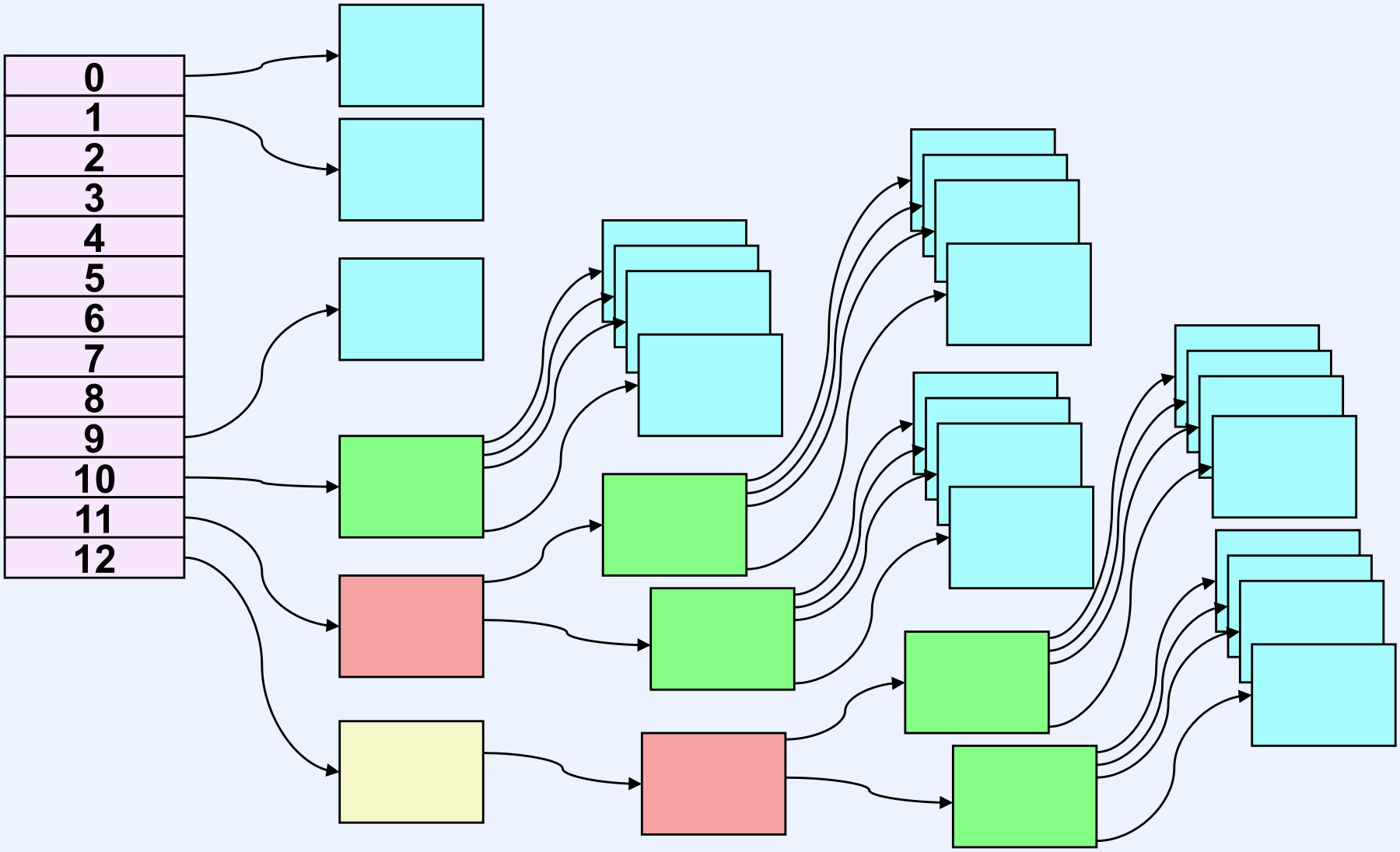
File Systems Part 2

Disk Map



Friday's Quiz

Suppose a new file is created. (At this point it occupies zero blocks.) Then one byte is written to it at byte offset $(266 \times 2^{10}) + 1$. Assume the block size is 2^{10} and block addresses occupy four bytes. How many blocks are required to represent the file, not counting its inode?

a) more than 270

b) 270

c) 3

d) 1

Quiz 1

Suppose one now writes to all locations in the file, from its beginning up to the location written to in the previous slide (byte offset $(266 \times 2^{10}) + 1$). How many blocks are required to represent the file, not counting its inode?

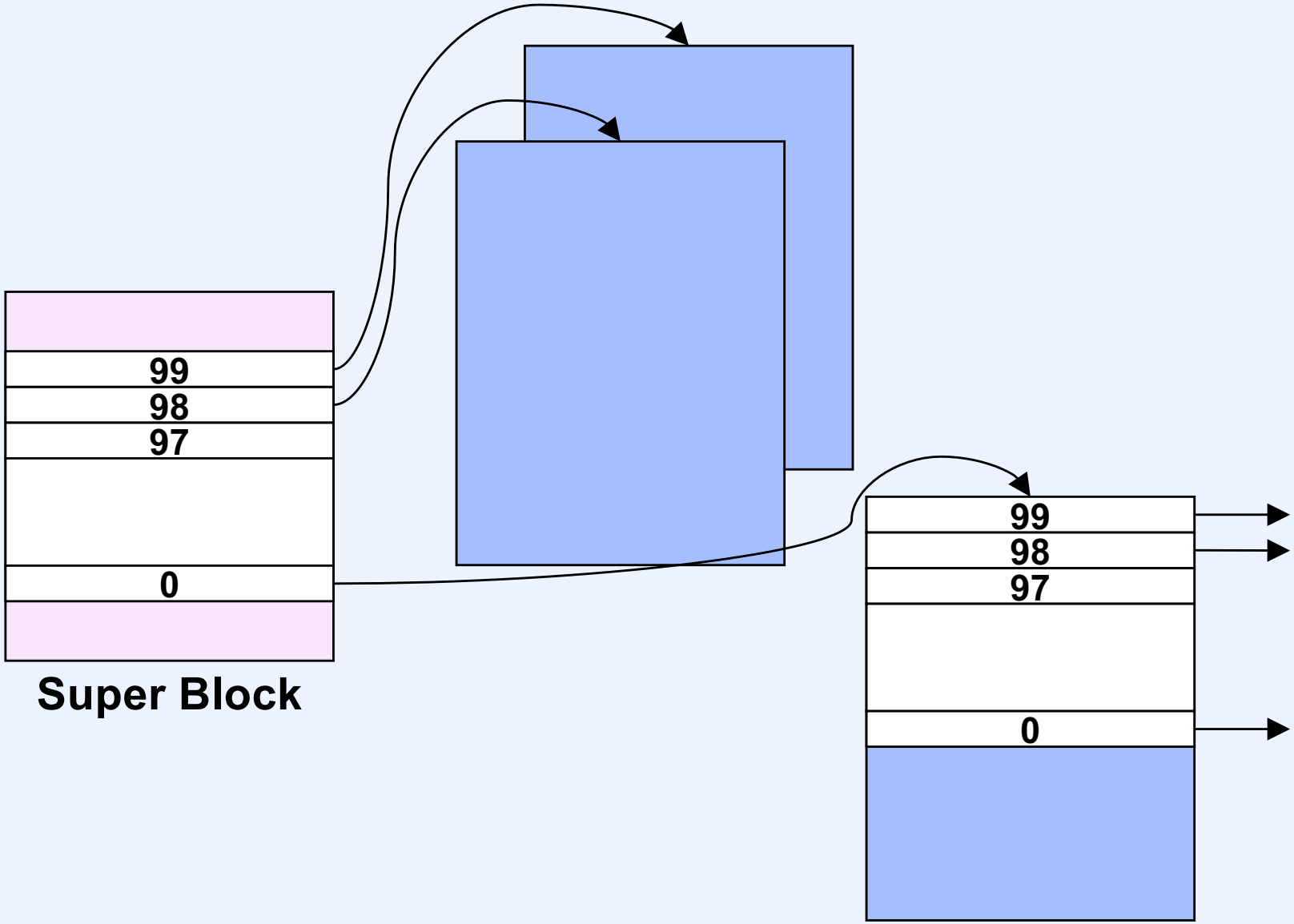
a) more than 270

b) 270

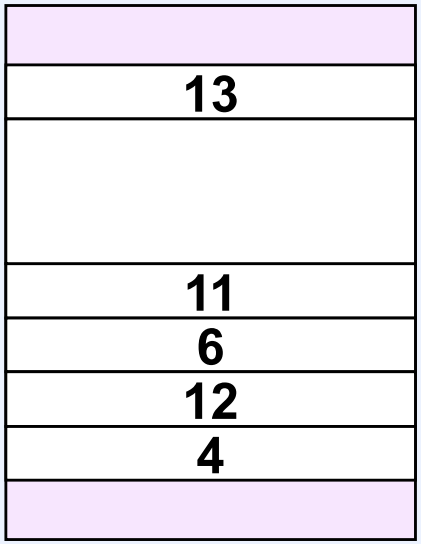
c) 3

d) 1

S5FS Free List



S5FS Free Inode List

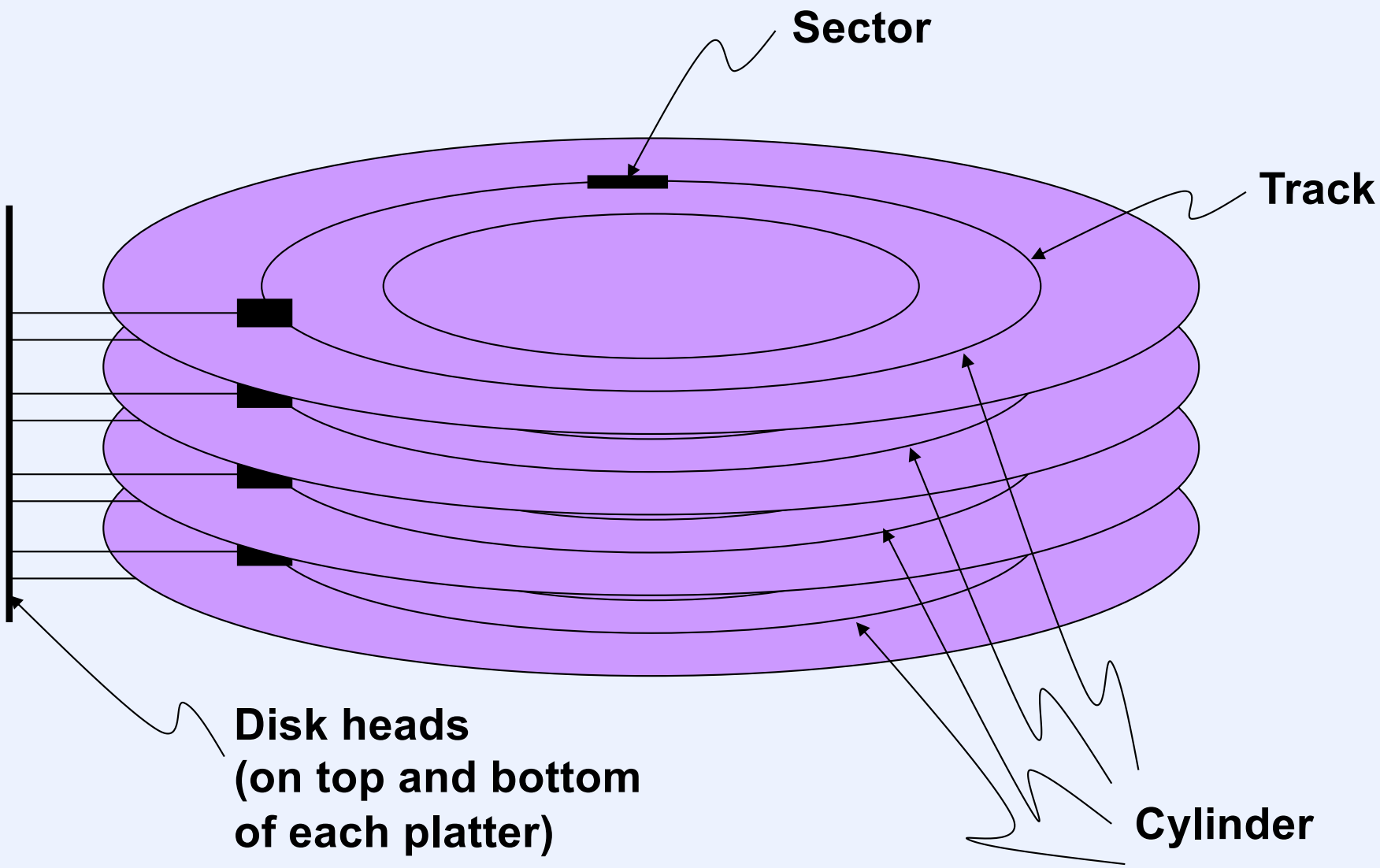


Super Block

16	0	
15		
14		
13	0	
12	0	
11	0	
10		
9	0	
8		
7		
6	0	
5		
4	0	
3		
2		
1		

I-list

Disk Architecture



CS167 Disk Drive

Rotation speed	10,000 RPM
Number of surfaces	8
Sector size	512 bytes
Sectors/track	500-1000; 750 average
Tracks/surface	100,000
Storage capacity	307.2 billion bytes
Average seek time	4 milliseconds
One-track seek time	.2 milliseconds
Maximum seek time	10 milliseconds

S5FS on CS167
(A Marketing Disaster …)

- CS167's maximum transfer speed?
  - 63.9 MB/sec
- S5FS's average transfer speed on CS167?
  - average seek time:
    - < 4 milliseconds (say 2)
  - average rotational latency:
    - ~3 milliseconds
  - per-sector transfer time:
    - negligible
  - time/sector: 5 milliseconds
  - transfer time: 102.4 KB/sec (.16% of maximum)

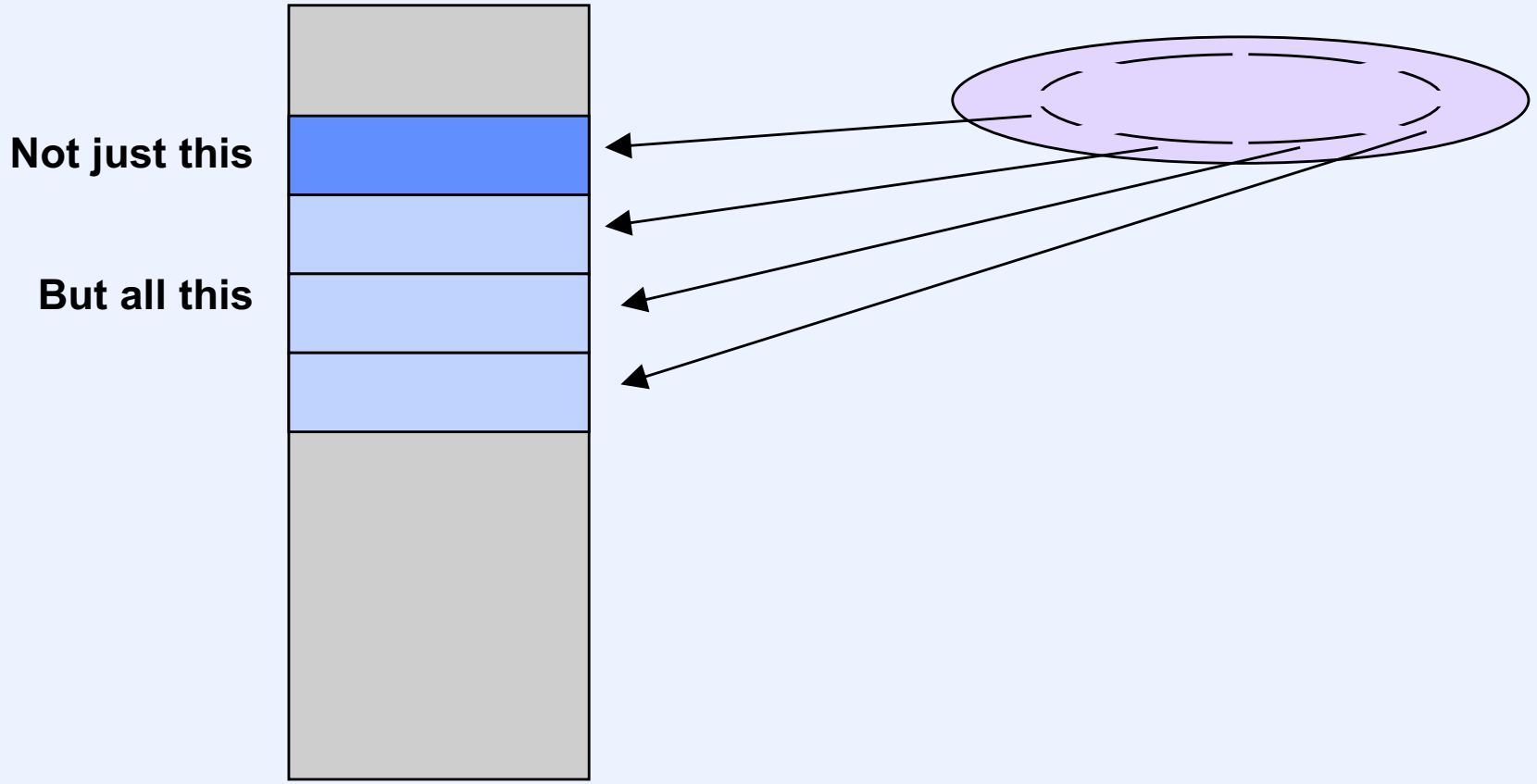
What to Do About It?

- Hardware
  - employ pre-fetch buffer
    - filled by hardware with what's underneath head
    - helps reads; doesn't help writes
- Software
  - better on-disk data structures
    - increase block size
    - minimize seek time
    - reduce rotational latency

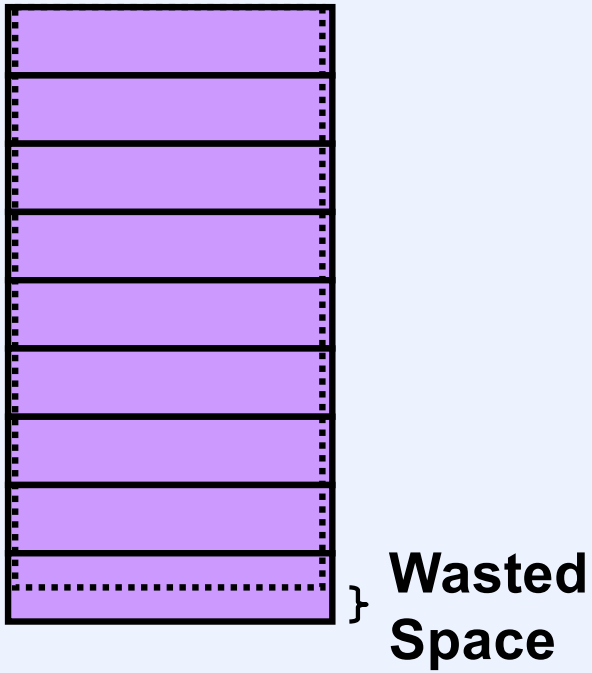
FFS

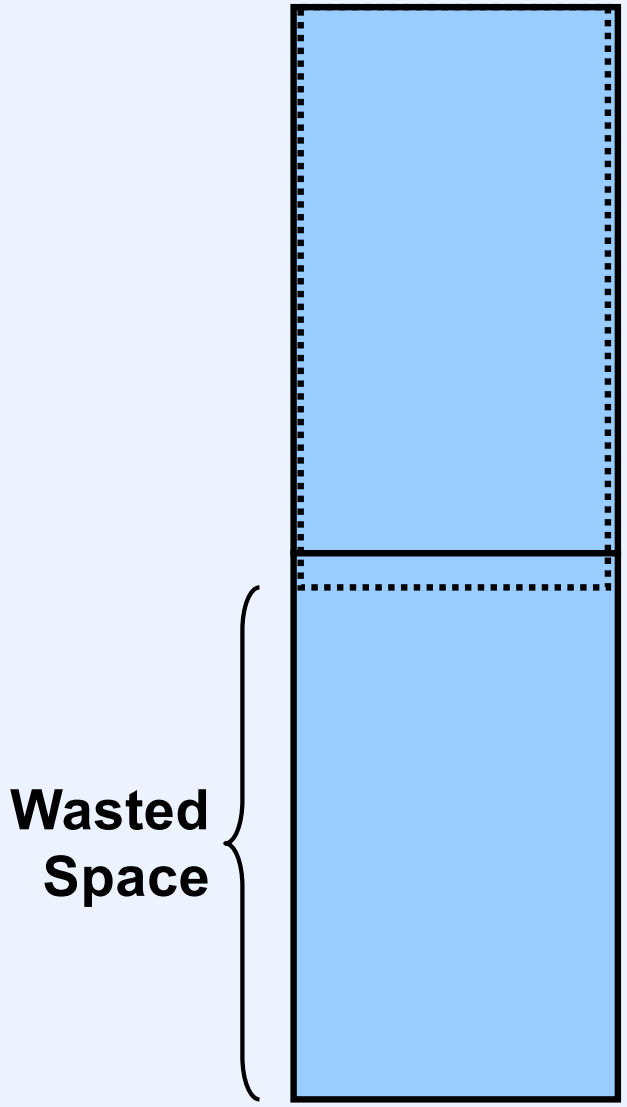
- Better on-disk organization
- Longer component names in directories
- Retains disk map of S5FS

Larger Block Size

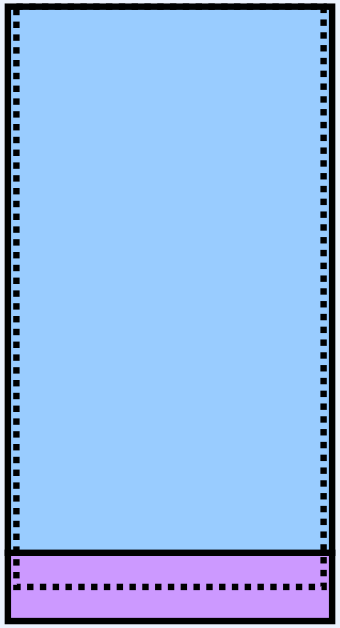


The Down Side …





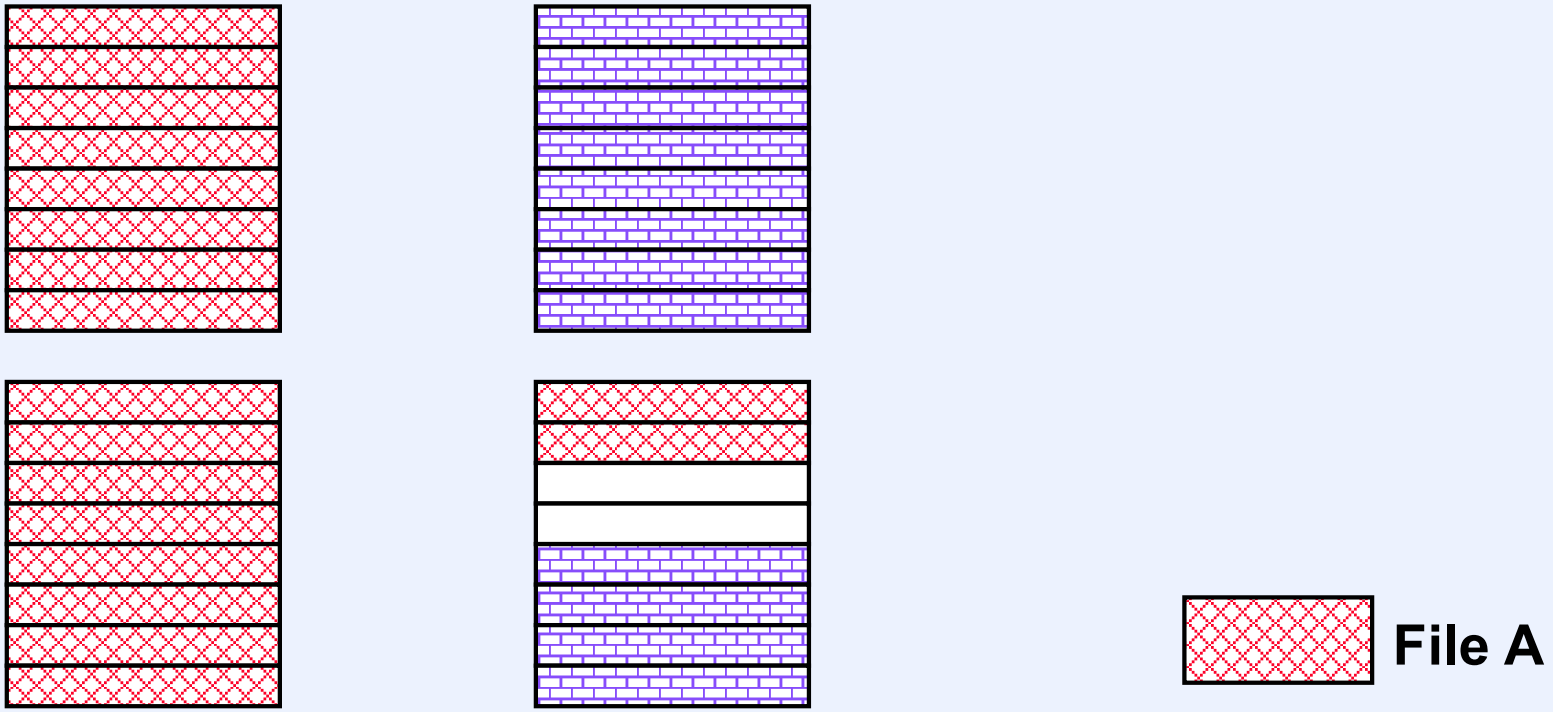
Two Block Sizes …



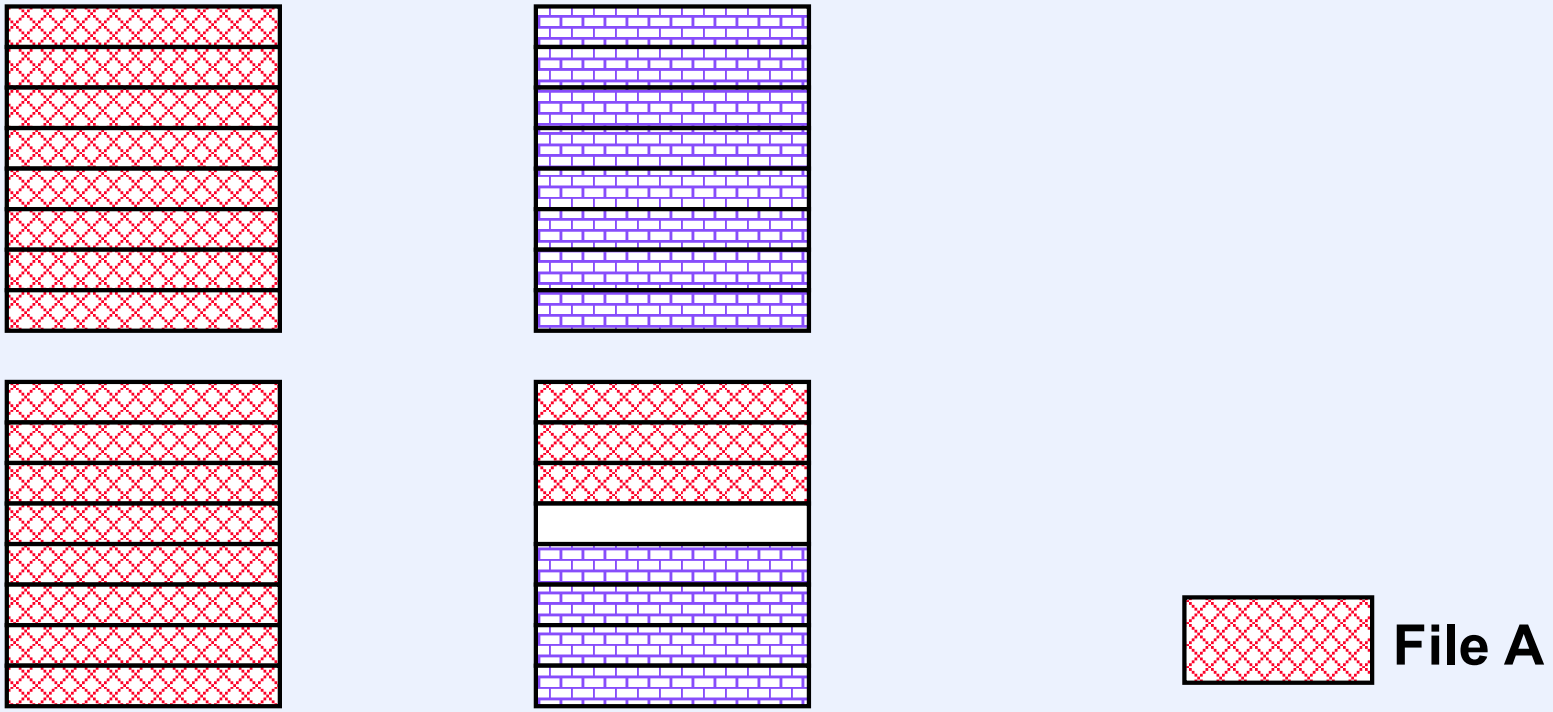
Rules

- File-system blocks may be split into fragments that can be independently assigned to files
  - fragments assigned to a file must be contiguous and in order
- The number of fragments per block (1, 2, 4, or 8) is fixed for each file system
- Allocation in fragments may only be done on what would be the last block of a file, and only for small files

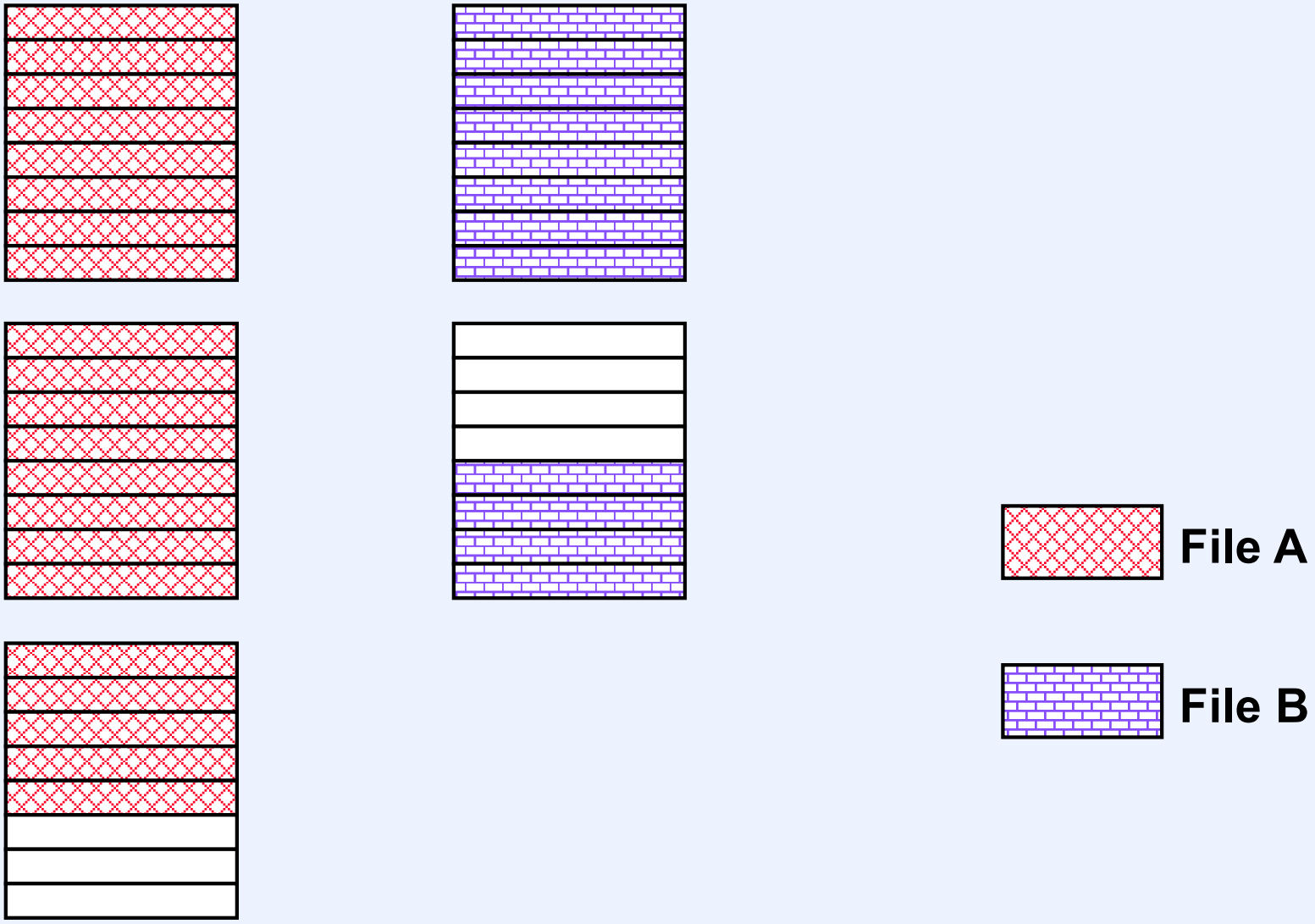
Use of Fragments (1)



Use of Fragments (2)



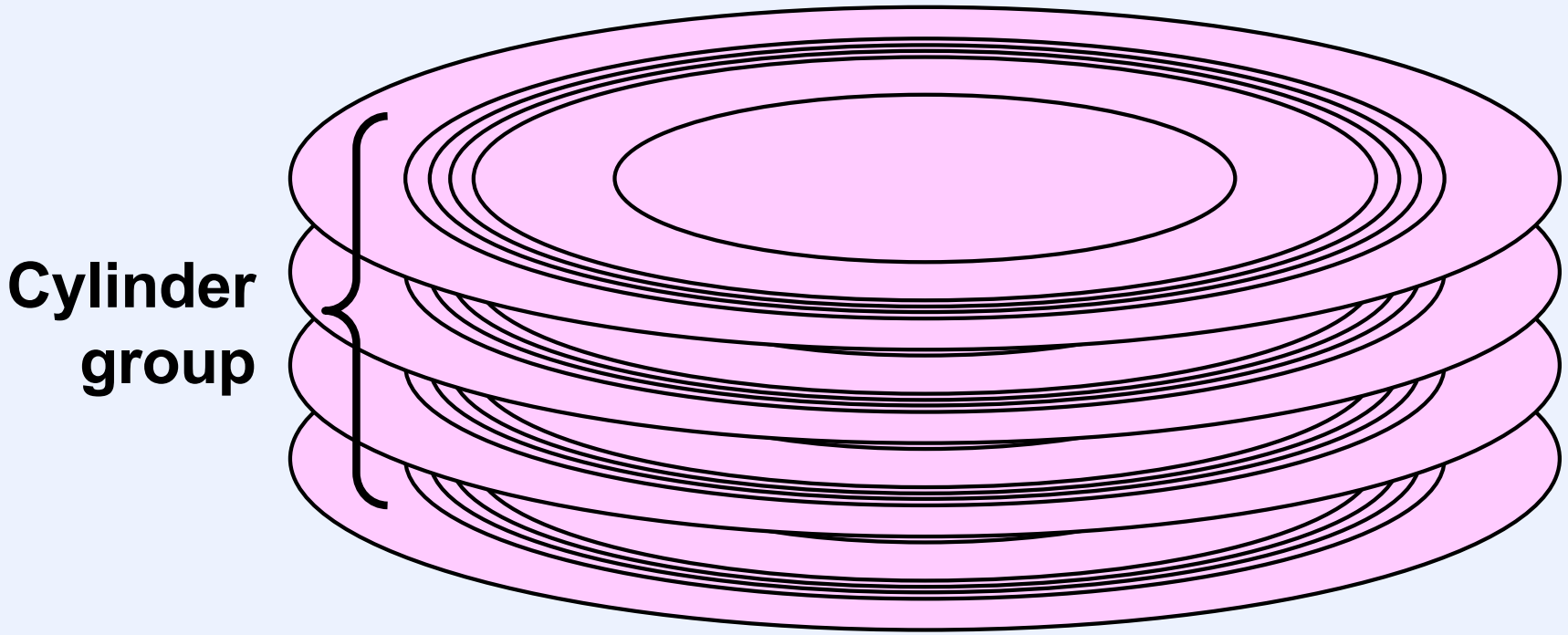
Use of Fragments (3)



Minimizing Seek Time

- Keep related things close to one another
- Separate unrelated things

Cylinder Groups



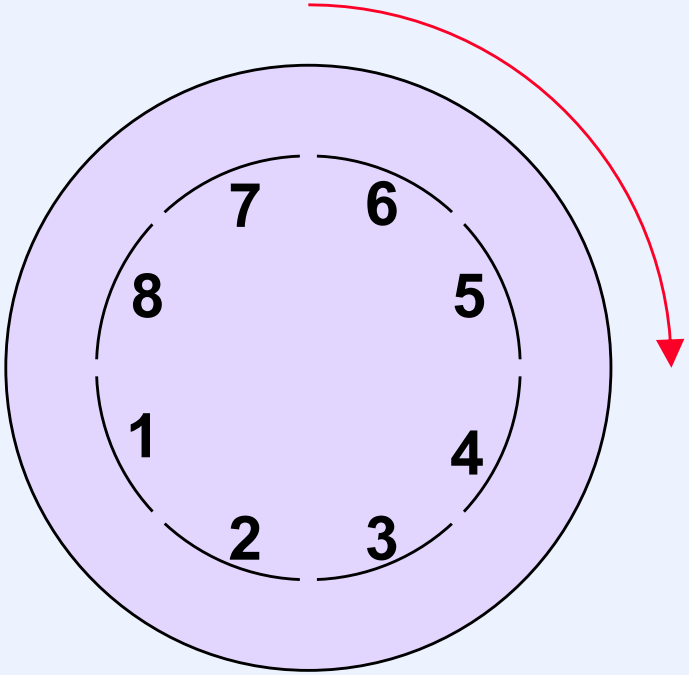
Minimizing Seek Time

- The practice:
  - attempt to put new inodes in the same cylinder group as their directories
  - put inodes for new directories in cylinder groups with "lots" of free space
  - put the beginning of a file (direct blocks) in the inode's cylinder group
  - put additional portions of the file (each 2MB) in cylinder groups with "lots" of free space

How Are We Doing?

- Configure CS167 with 20 cylinders per group
  - 2-MB file fits entirely within one cylinder group
  - average seek time within cylinder group is ~.3 milliseconds
  - average rotational delay still 3 milliseconds
  - .12 milliseconds required for disk head to pass over 8KB block
  - 3.42 milliseconds for each block
  - 2.4 million bytes/second average transfer time
    - 20-fold improvement
    - 3.7% of maximum possible

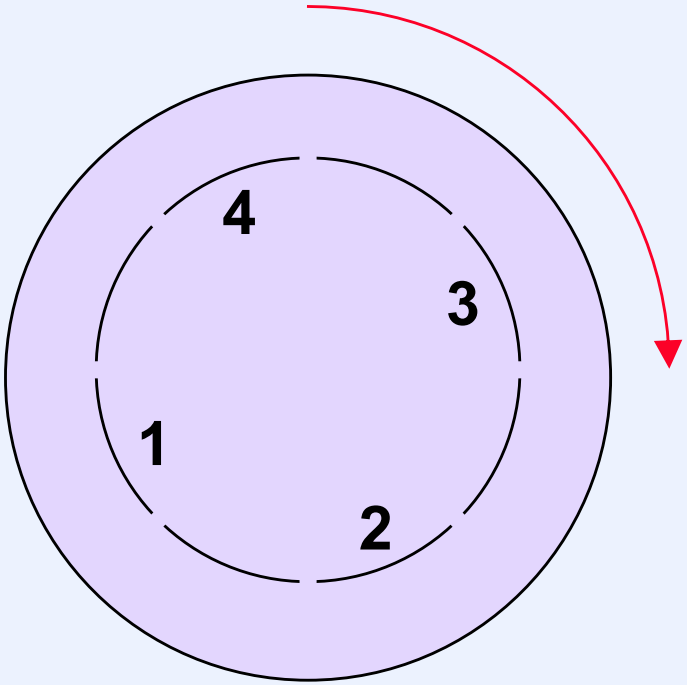
Minimizing Latency (1)

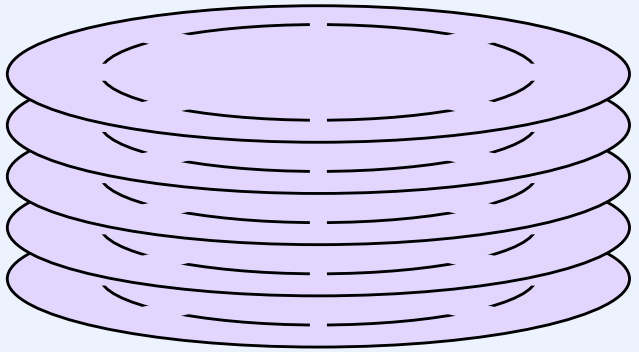


Numbers

- CS167 spins at 10,000 RPM
  - 6 milliseconds/revolution
- Disk I/O done one block at a time
- 100 microseconds required to field disk-completion interrupt and start next operation
  - typical of early 1980s
- Each block takes 120 microseconds to traverse disk head
- Reading successive blocks is time-consuming!

Minimizing Latency (2)





How’re We Doing Now? (part 1)

- Time to read successive blocks (two-way interleaving):
  - after request for second block is issued, must wait 20 microseconds for the beginning of the block to rotate under disk head
    - factor of 300 improvement!

How’re We Doing Now? (part 2)

- Same setup as before
  - 2-MB file within one cylinder group
  - actually fits in one cylinder
  - block interleaving employed: every other block is skipped
  - .3-millisecond seek to that cylinder
  - 3-millisecond rotational delay for first block
  - 50 blocks/track, but 25 read in each revolution
  - 10.24 revolutions required to read all of file
  - 32.4 MB/second (50% of maximum possible)

Quiz 2

If file access is one (8KB) block at a time and we employ 2-way block interleaving, can we do better than the 50% of maximum transfer speed achieved by FFS?

a) yes – we can get arbitrarily close to 100%

b) yes, but the limit is somewhere between 50% and 100%

c) no – we've reached the limit

Further Improvements?

- S5FS: 0.16% of capacity
- FFS without block interleaving: 3.8% of capacity
- FFS with block interleaving: 50% of capacity
- What next?

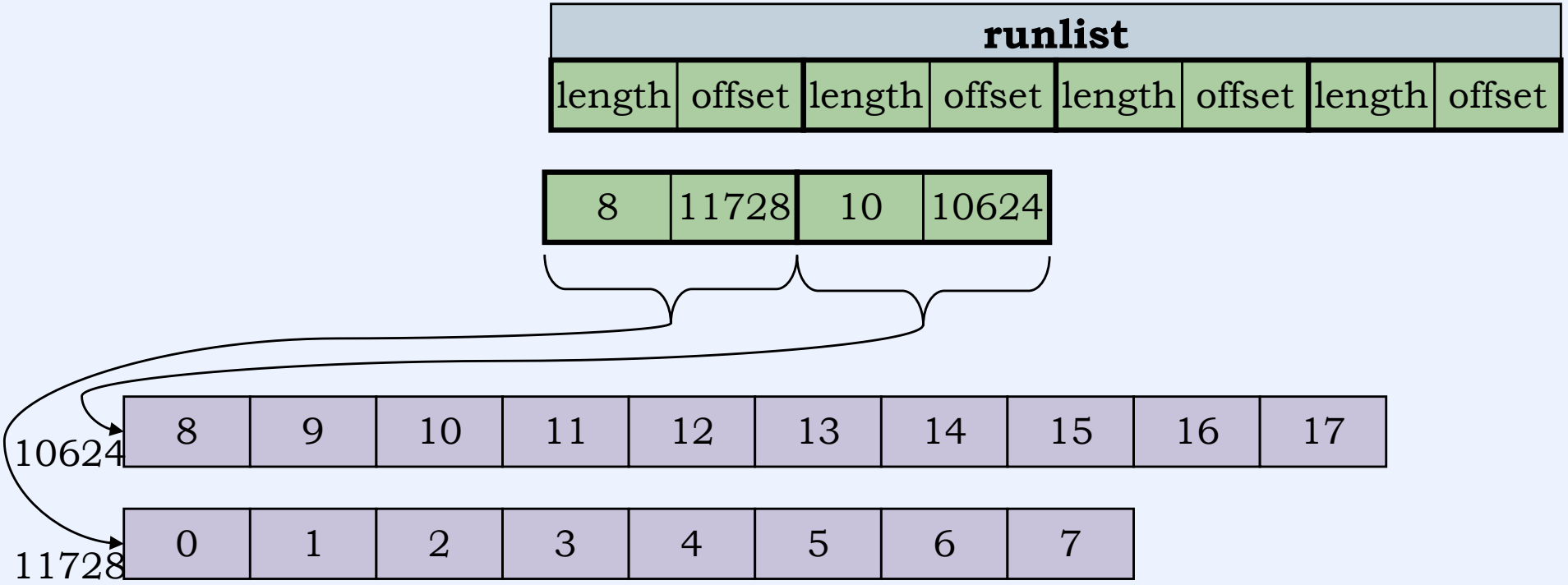
Larger Transfer Units

- Allocate in whole tracks or cylinders
  - too much wasted space
- Allocate in blocks, but group them together
  - transfer many at once
  - wasted space, but not as bad

Block Clustering

- Allocate space in blocks, eight at a time
- Linux's Ext2 (an FFS clone):
  - allocate eight blocks at a time
  - extra space is available to other files if there is a shortage of space
- FFS on Solaris (~1990)
  - delay disk-space allocation until:
    - 8 blocks are ready to be written
    - or the file is closed

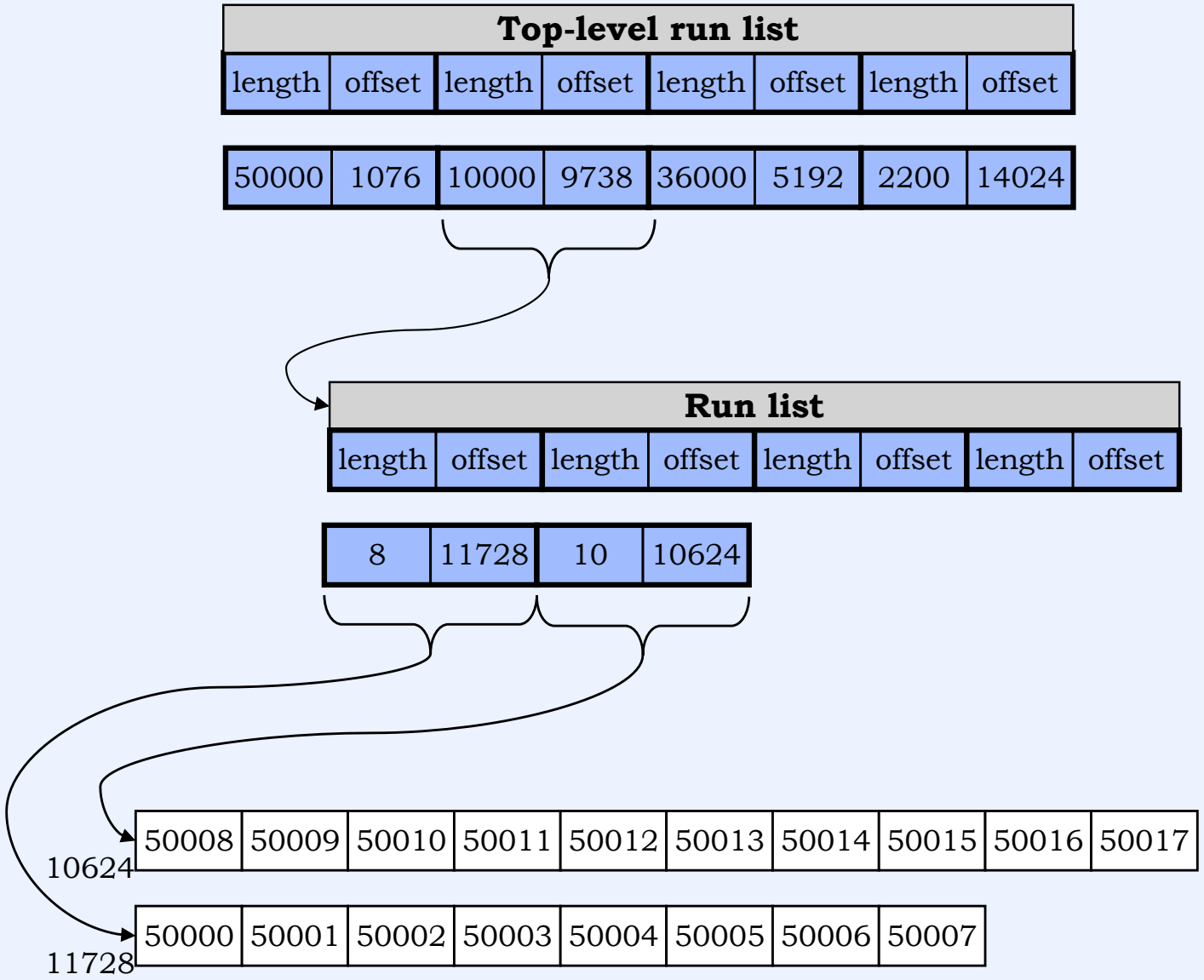
Extents



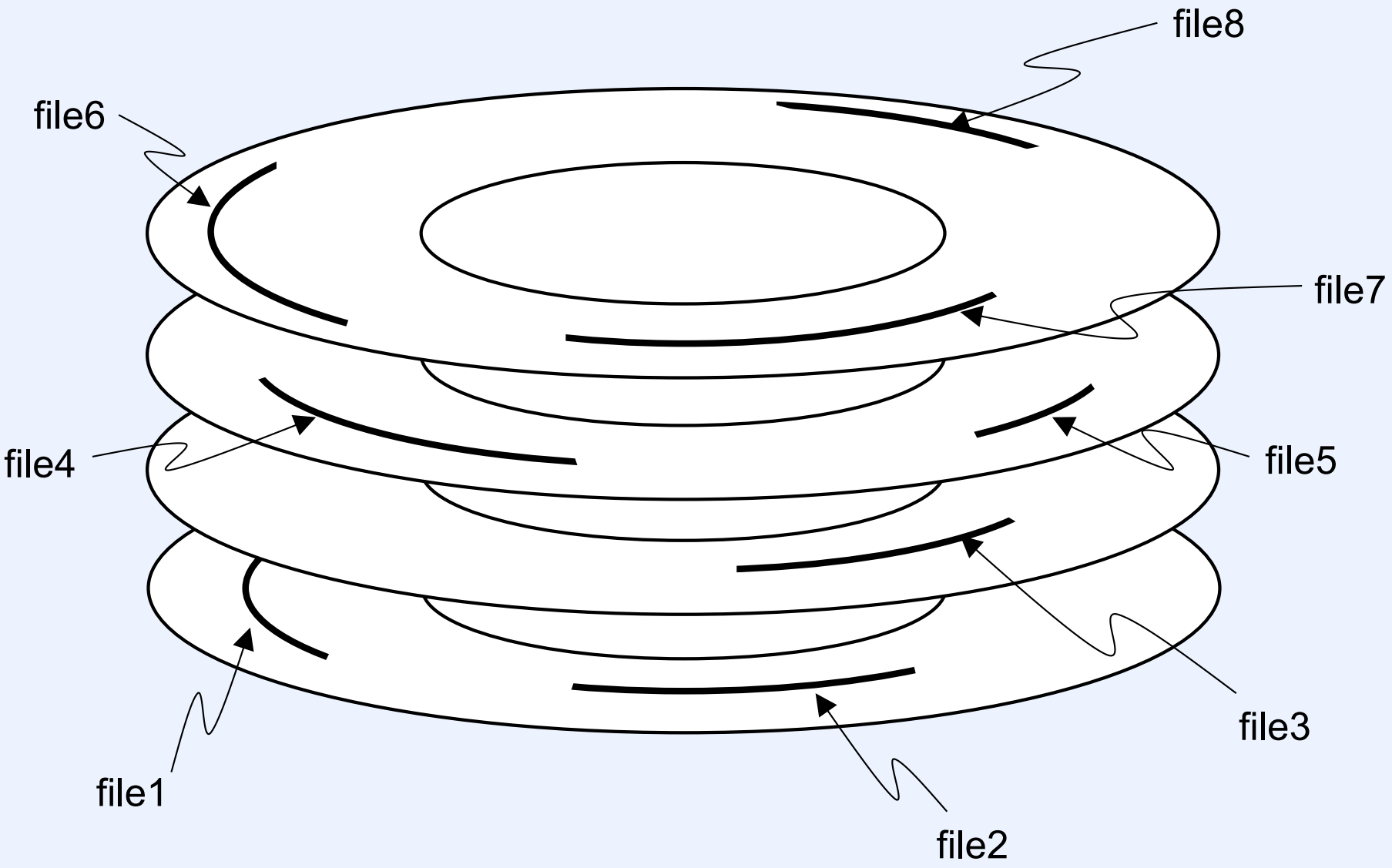
Problems with Extents

- Could result in highly fragmented disk space
  - lots of small areas of free space
  - solution: use a *defragmenter*
- Random access
  - linear search through a long list of extents
  - solution: multiple levels

Extents in NTFS



Good File Layout



Quiz 3

So far we've been attempting to optimize *logical locality* in file systems: bytes that are logically near one another within a file are physically near one another on disk.

a) This is important on both single-user systems and busy (multi-client) file servers

b) This is important on single-user systems, but of marginal importance on busy (multi-client) file servers

c) This is of marginal importance on single-user systems