Lecture 7 File Systems

Last Day: File Directory Structures

- Tree structures
- Shared Files
- Reference counters
- Garbage collection

Today: UNIX File system

- Index Nodes (Inodes)
- Accessing data
- Layout of blocks on disk

See handout on file systems

UNIX

- Unix is an Operating System (O.S.)

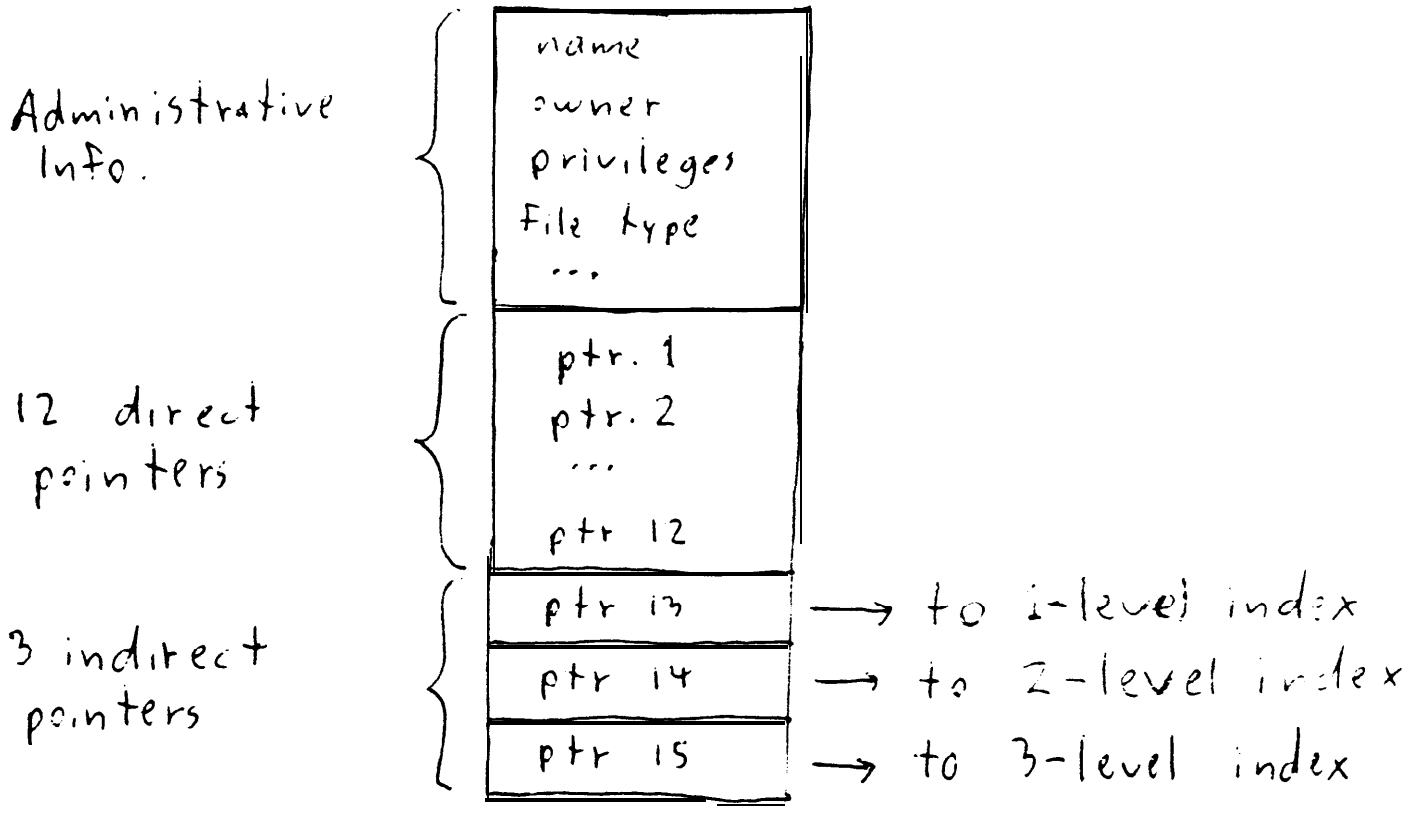
- As such, it frees the user from many details

- eg. The Unix file system (part of the O.S. manages the disk's free pool, allocates disk blocks to new files, & manages the file directory and active file table.

Inodes

- In Unix, each entry in the file directory is called an **Inode** (index node).
- There is one inode for each file on disk
- An inode records administrative information about the file.
- It is also the start of several multi-level indices for the file.

I node



- Pointers 1-12 point directly to file blocks.

- Pointers 13, 14, 15 point to the roots of multi-level indices, which eventually lead to file blocks.

UNIX 4.2 BSD

- In the 4.2 BSD implementation of Unix, block size is 4 KB (ie, 2^{12} bytes) (where 1 KB = 2^{10} bytes).

- And, pointers are 32 bits (4 bytes) each.

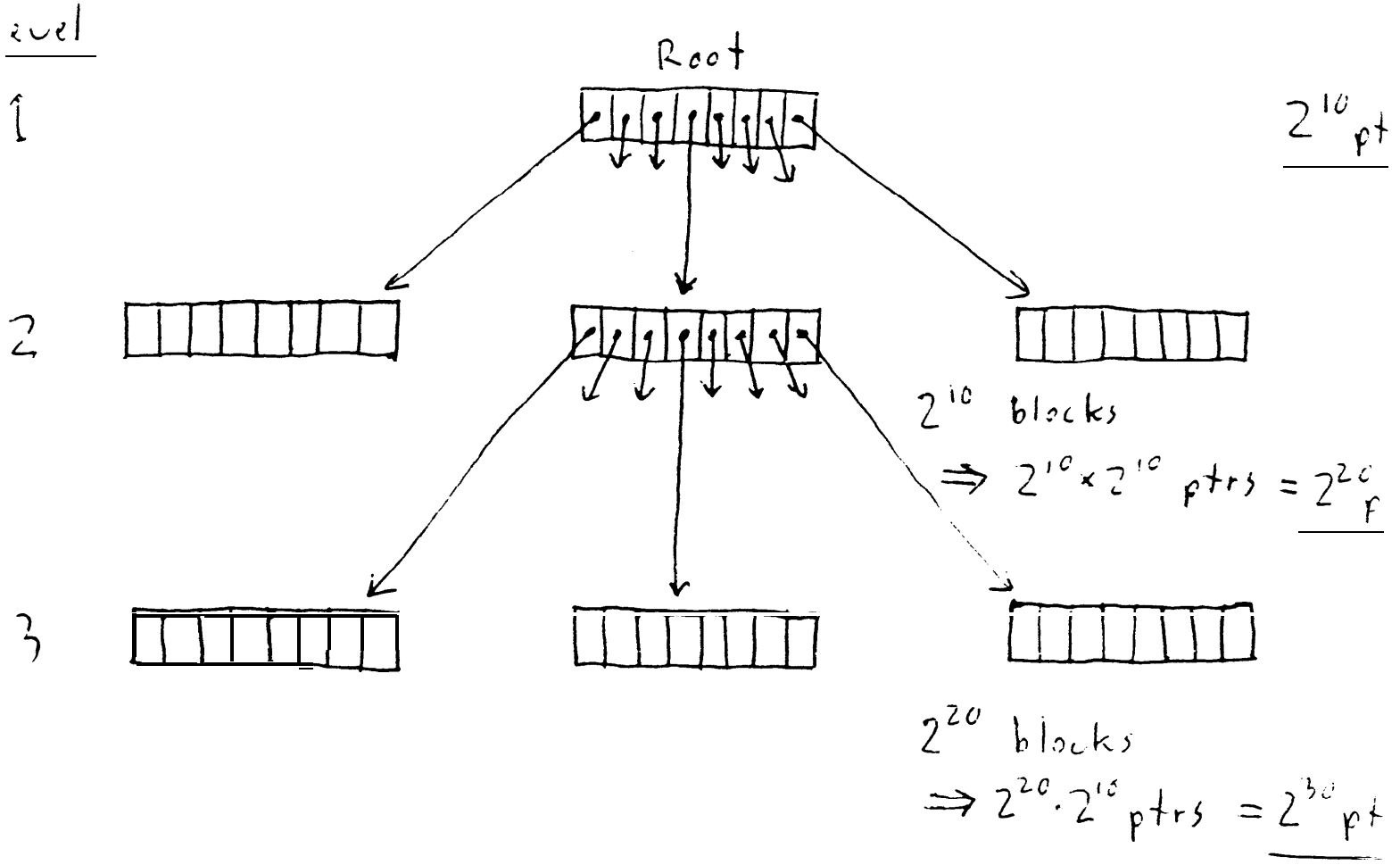
- ∴ 1 block can store 2^{10} pointers.

ie, $$\frac{2^{12} \text{ bytes/blk}}{4 \text{ bytes/ptr}} = \frac{2^{12} \text{ bytes/blk}}{2^2 \text{ bytes/ptr}}$$

$$= 2^{12-2} \text{ ptrs/blk}$$

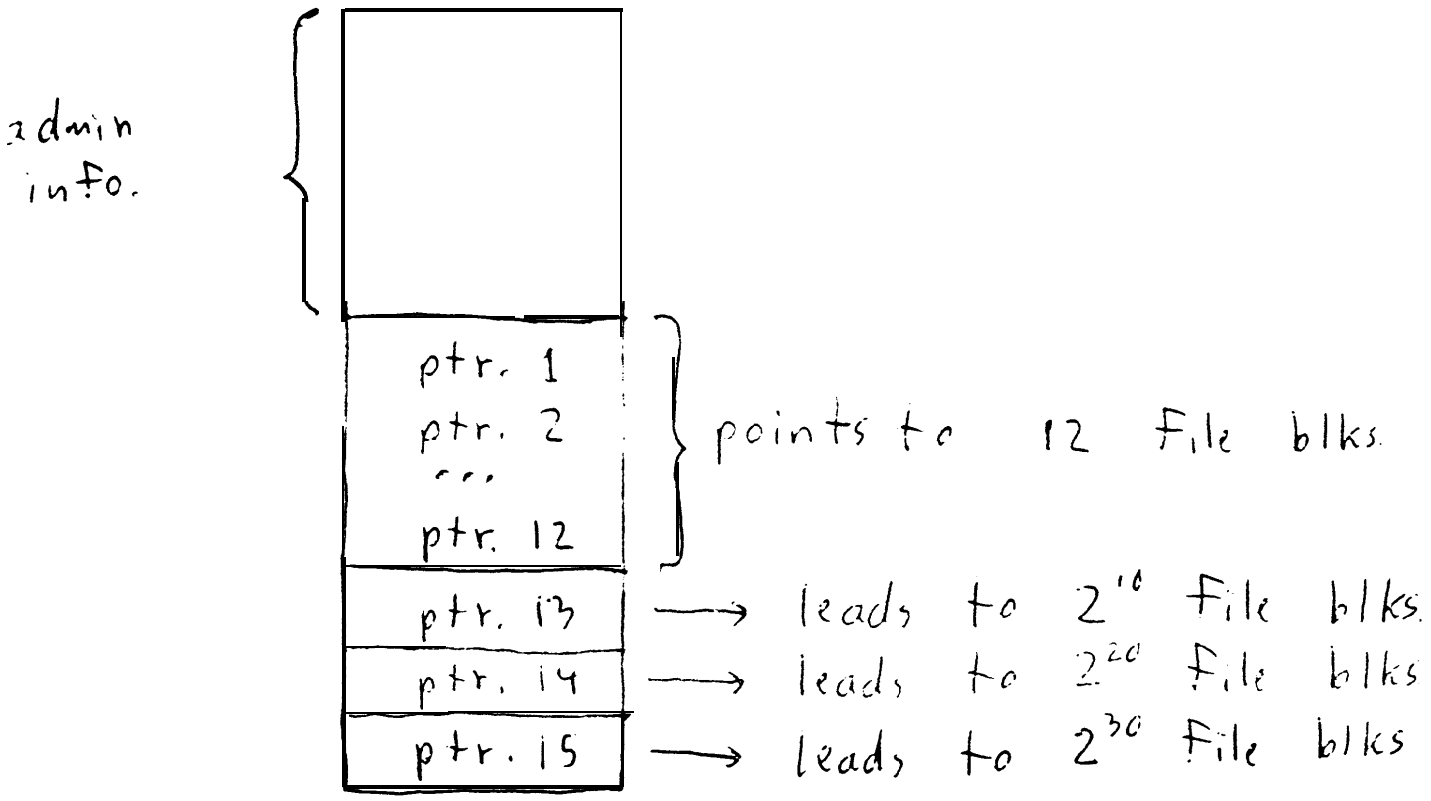
$$= 2^{10} \text{ ptrs/blk}$$

A Multi-Level Index

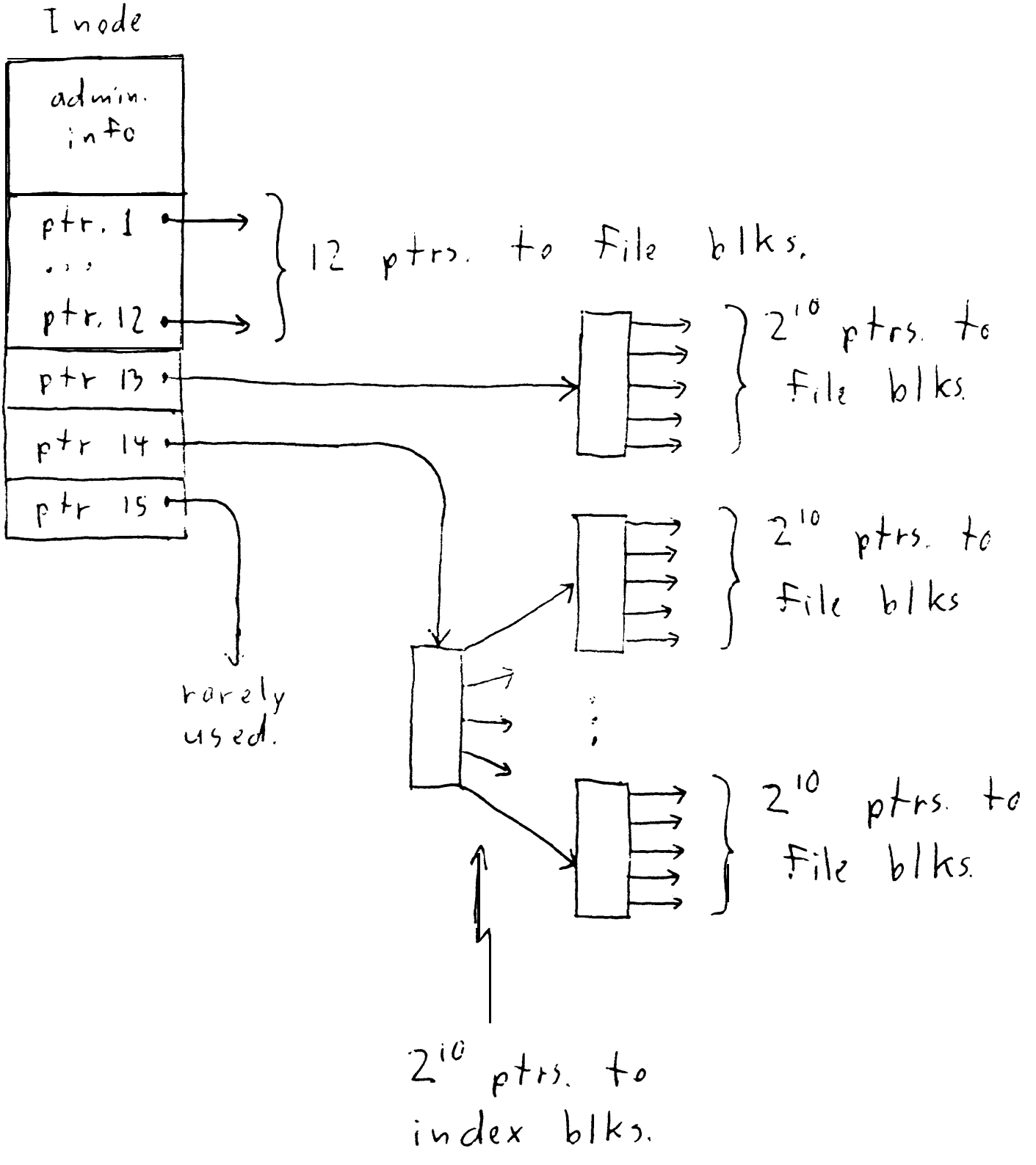


- Level k has $2^{10 \cdot k}$ pointers.
- ie, A k-level index can point to $2^{10 \cdot k}$ File blocks.

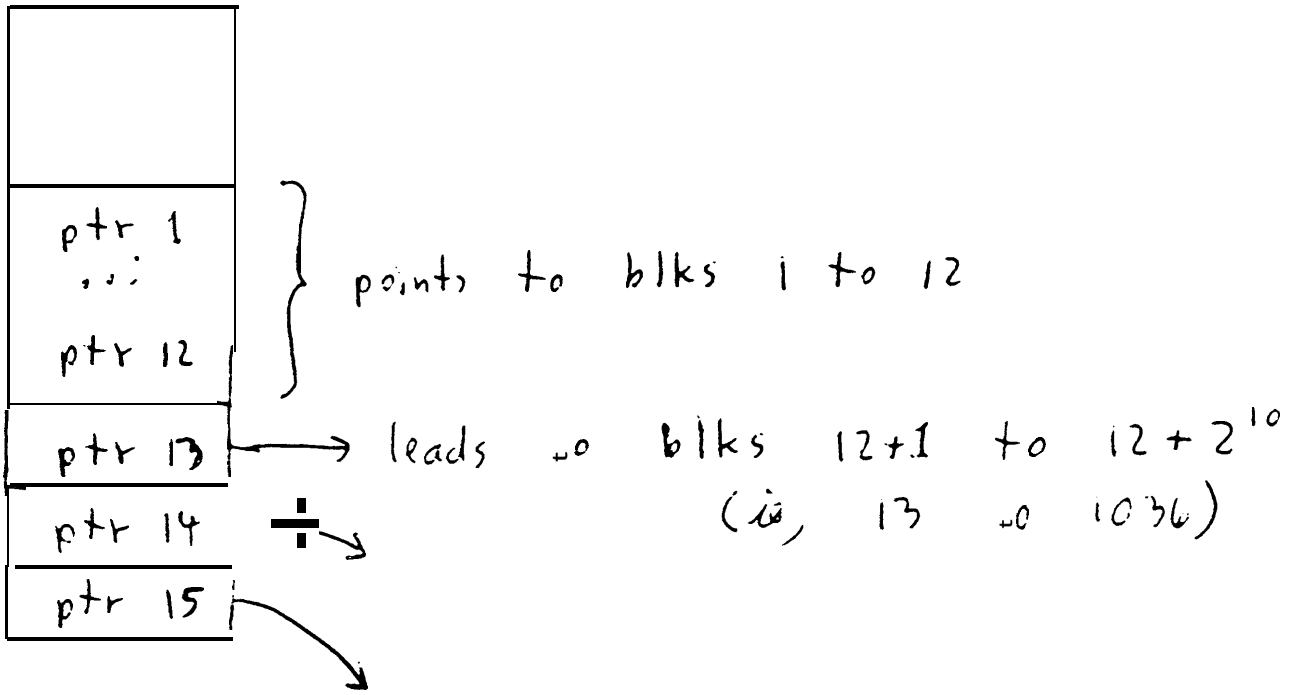
An Inode


admin info.
ptr. 1
ptr. 2
...
ptr. 12
points to 12 File blks.
ptr. 13
leads to 2^{10} File blks.
ptr. 14
leads to 2^{20} File blks
ptr. 15
leads to 2^{30} File blks

In Detail


Inode
admin. info
ptr. 1
...
ptr. 12
12 ptrs. to file blks.
ptr 13
ptr 14
ptr 15
2^{10} ptrs. to file blks.
2^{10} ptrs. to file blks
2^{10} ptrs. to file blks.
rarely used.
2^{10} ptrs. to index blks.

Access Speed



Note: If the inode is in main memory, then

- Blk. 2 of the file can be reached in 1 disk access, via ptr 2.

- Blk. 200 of the file can be reached in 2 disk accesses, via ptr. 13

- Blk. 2000 of the file can be reached in 3 disk accesses via ptr 14

With this organization,

- short files (12 blks or less) can be accessed directly from the inode (i.e., in 1 disk access).

- Medium Files (up to $12 + 2^{10}$ blks) can be accessed in 2 disk accesses or less.

- Large Files (up to $12 + 2^{10} + 2^{20}$ blks) can be accessed in 3 disk accesses or less.

- Very large files (up to $12 + 2^{10} + 2^{20} + 2^{30}$) can be accessed in 4 disk accesses or less.

Retrieving Data: Examples

- Each Unix block holds $2^{12} = 4096$ bytes
- So, if the user's program asks for byte number 107, then Unix will retrieve block # 1, and then return the 107^{th} byte in the block to the user.
- In general, byte N is stored in block $\lceil N/4096 \rceil$
- To retrieve this block, Unix follows the appropriate pointer in the file's inode.

Examples

- To get byte 9000, Unix retrieves block $\lceil 9000/4096 \rceil = \lceil 2.2 \rceil = 3$, using pointer 3 in the inode.

- To get byte 26,000, Unix retrieves block $\lceil 26{,}000/4096 \rceil = \lceil 6.3 \rceil = 7$, using pointer 7 of the inode.

- To get byte 90,000, Unix retrieves block $\lceil 90{,}000/4096 \rceil = \lceil 21.9 \rceil = 22$, using pointer 13 of the inode (which leads to blocks 13 through 1036).

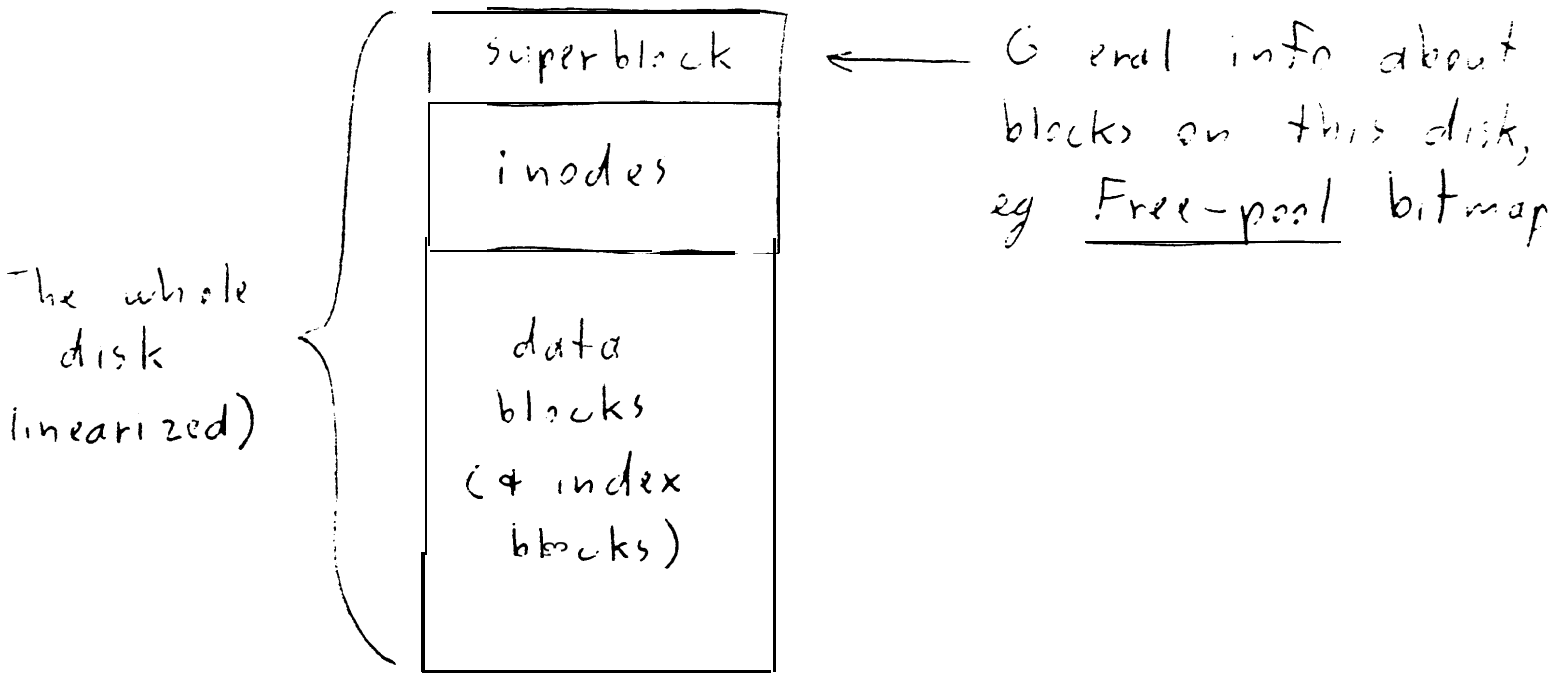
Examples (cont.)

- To get byte 900,000, Unix retrieves block $\lceil 900{,}000 / 4096 \rceil$, ie, block 220, using pointer 13 (which leads to blocks 13 through 1036).

- To get byte 9,000,000, Unix retrieves block $\lceil 9{,}000{,}000 / 4096 \rceil$, ie, block 2198, using pointer 14.

Layout of Blocks on Disk in Unix

If disk space is viewed as a one dimensional array of blocks, then Unix organizes disk space as follows:



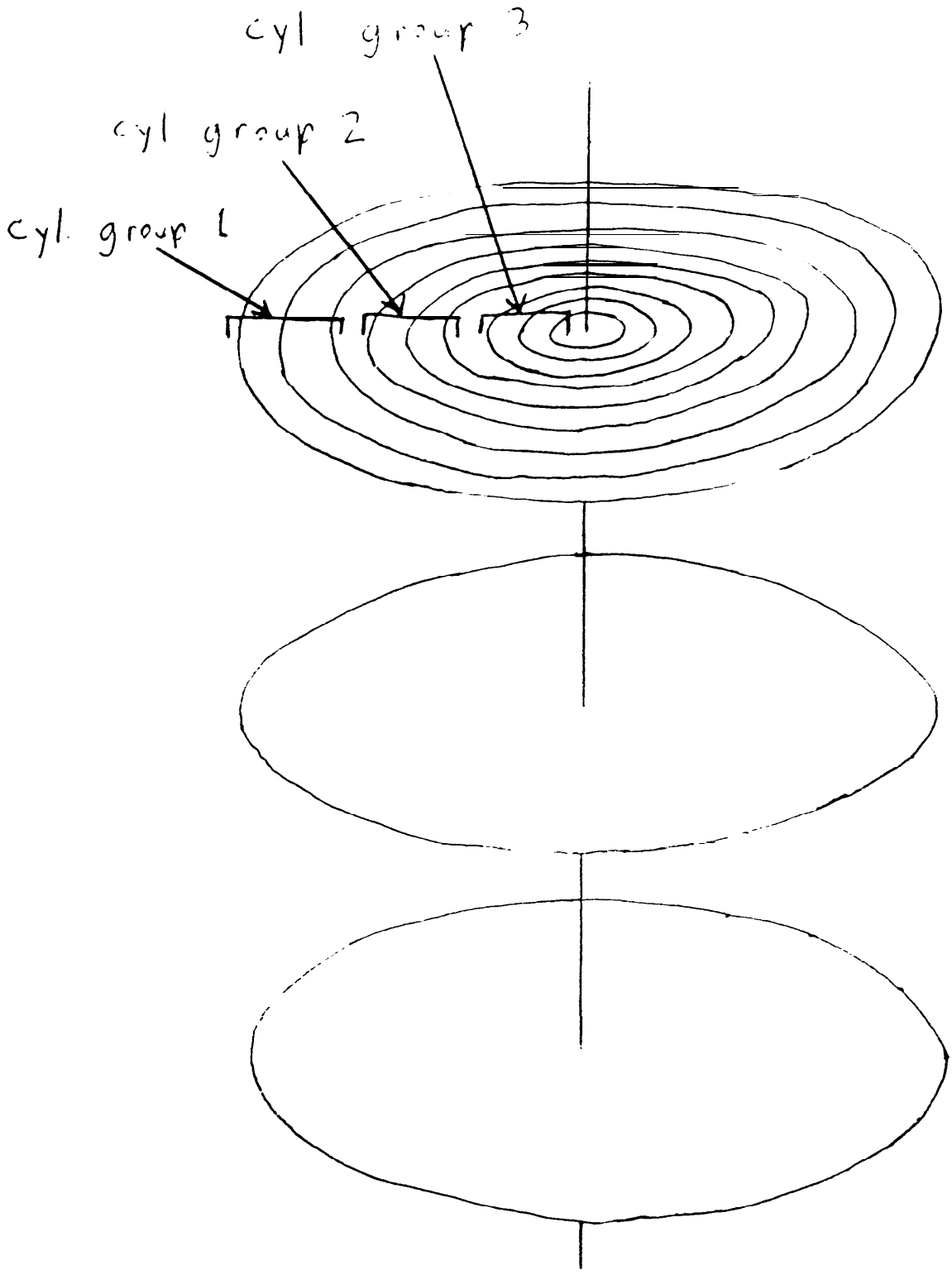
Problems

① Reliability. System crashes and O.S. bug may destroy the superblock, so that information about which blocks are free is lost.

② Efficiency: Data blocks of a file may be scattered all over disk, so processing a file can require many disk seeks (ie, slow)

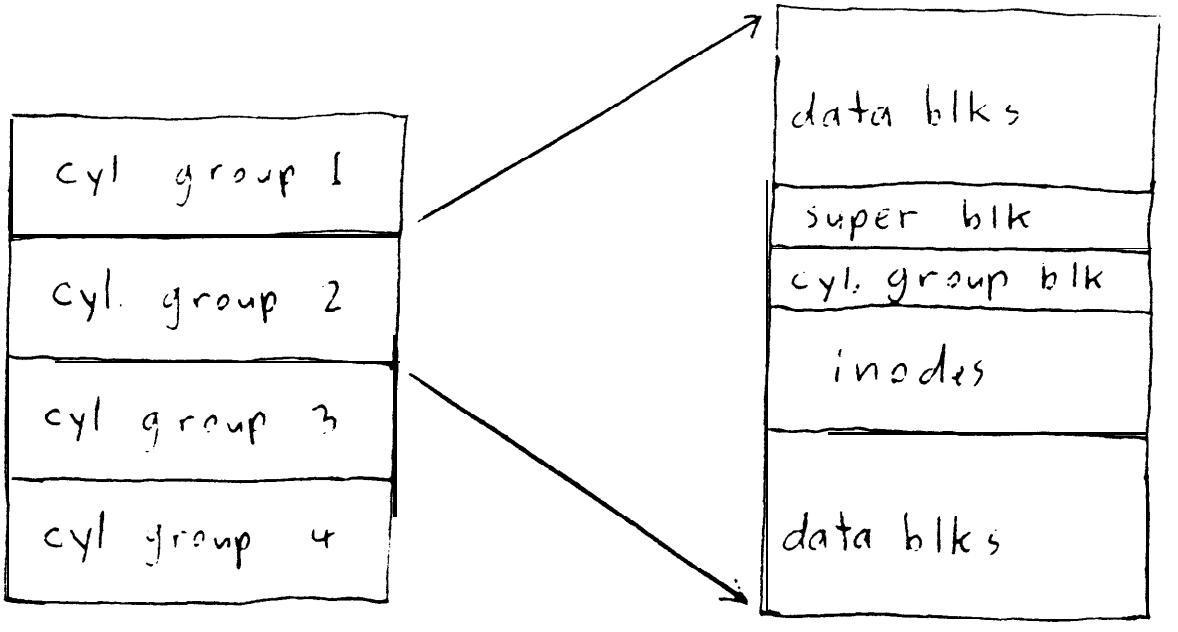
- To alleviate these problems, the 4.2 BSD implementation of Unix divides a disk into cylinder groups.

- A cylinder group is all the blocks in a number of consecutive cylinders.



Note: Seek time (disk head movement) within a cylinder group is small.

1-Dimensional View



- Each cyl group has a copy of the super block. This enhances reliability

- Each cyl group has its own cyl group block, describing the state of the blocks (free or used) in the group.

- Since the cyl. group block and super-block are accessed very often, they are stored in the center of a cylinder group.