File Systems (3)

Dr. Jun Zheng

CSE325 Principles of Operating Systems

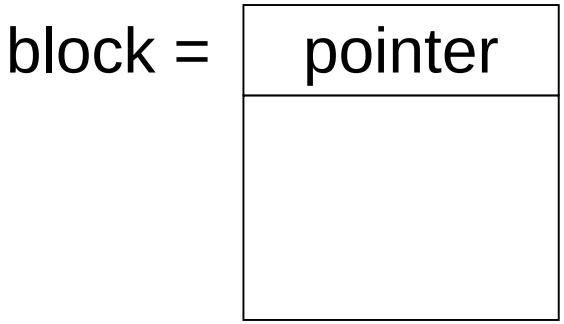
11/20/2019

Extent-Based Systems

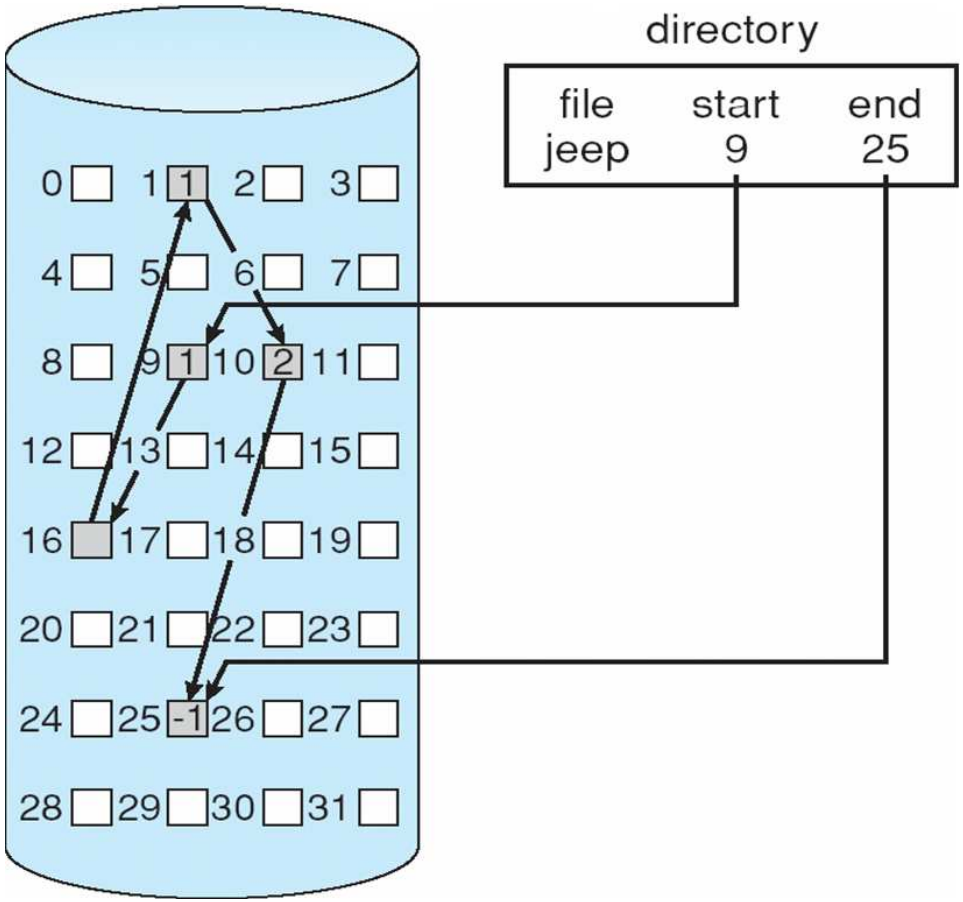
- Many newer file systems (i.e., Veritas File System) use a modified contiguous allocation scheme
- Extent-based file systems allocate disk blocks in extents
- An **extent** is a contiguous area of disks (a range of blocks)
  - Extents are allocated for file allocation
  - A file consists of one or more extents

Linked Allocation

- Each file is a linked list of disk blocks
  - Blocks may be scattered anywhere on the disk
- Each block contains pointer to next block



Linked Allocation



Pros

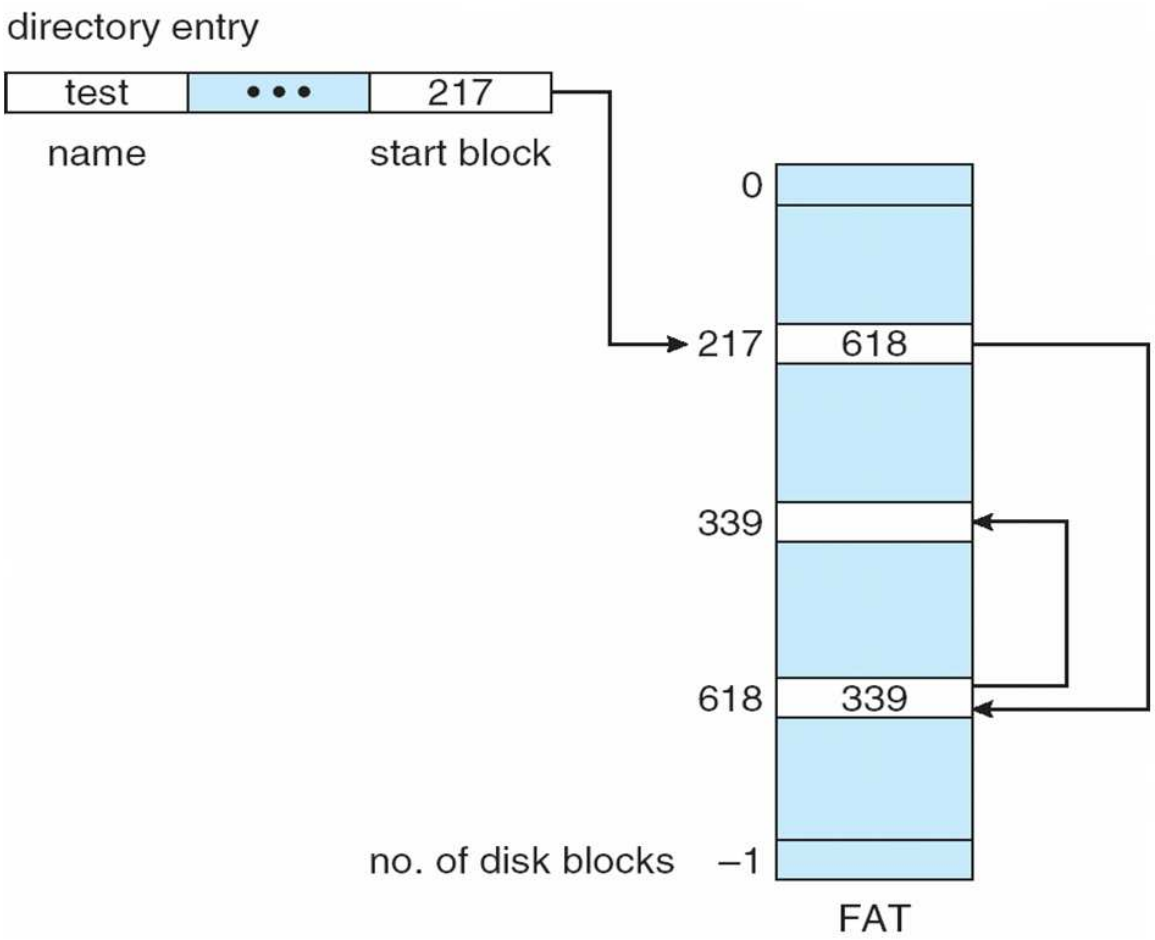
- Simple – need only starting address
- No external fragmentation

Cons

- No random access
- Seeks can be slow
- Space required for pointer
- Reliability

NEW MEXICO TECH
SCIENCE · ENGINEERING · RESEARCH · UNIVERSITY

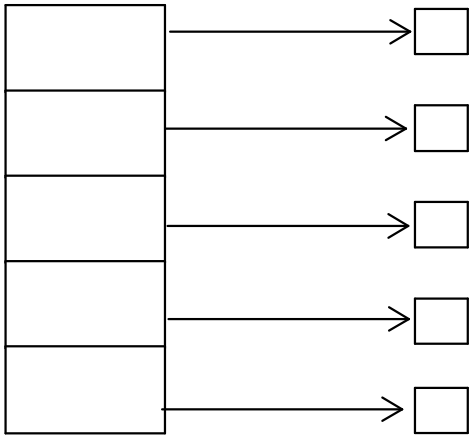
File-Allocation Table (FAT)



MS-DOS

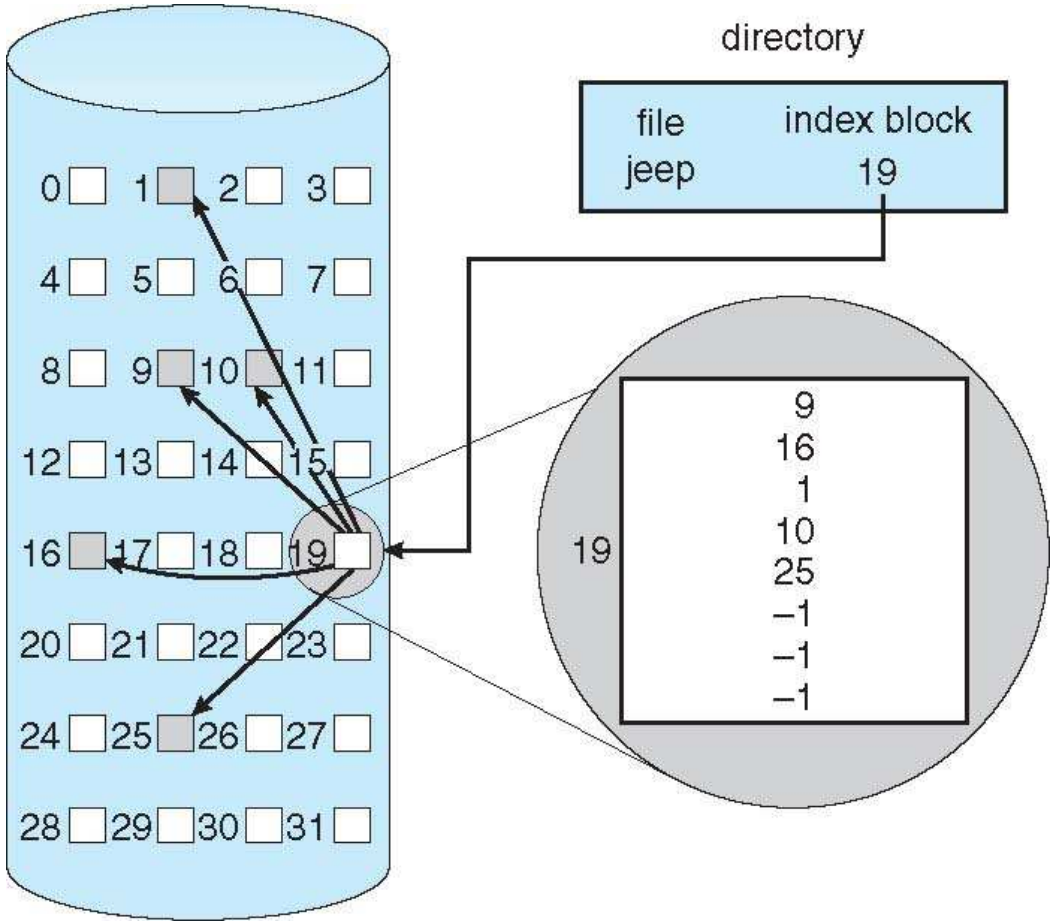
Indexed Allocation

- Each file has its own **index block**(s) of pointers to its data blocks
- Logical view



index table

Example of Indexed Allocation



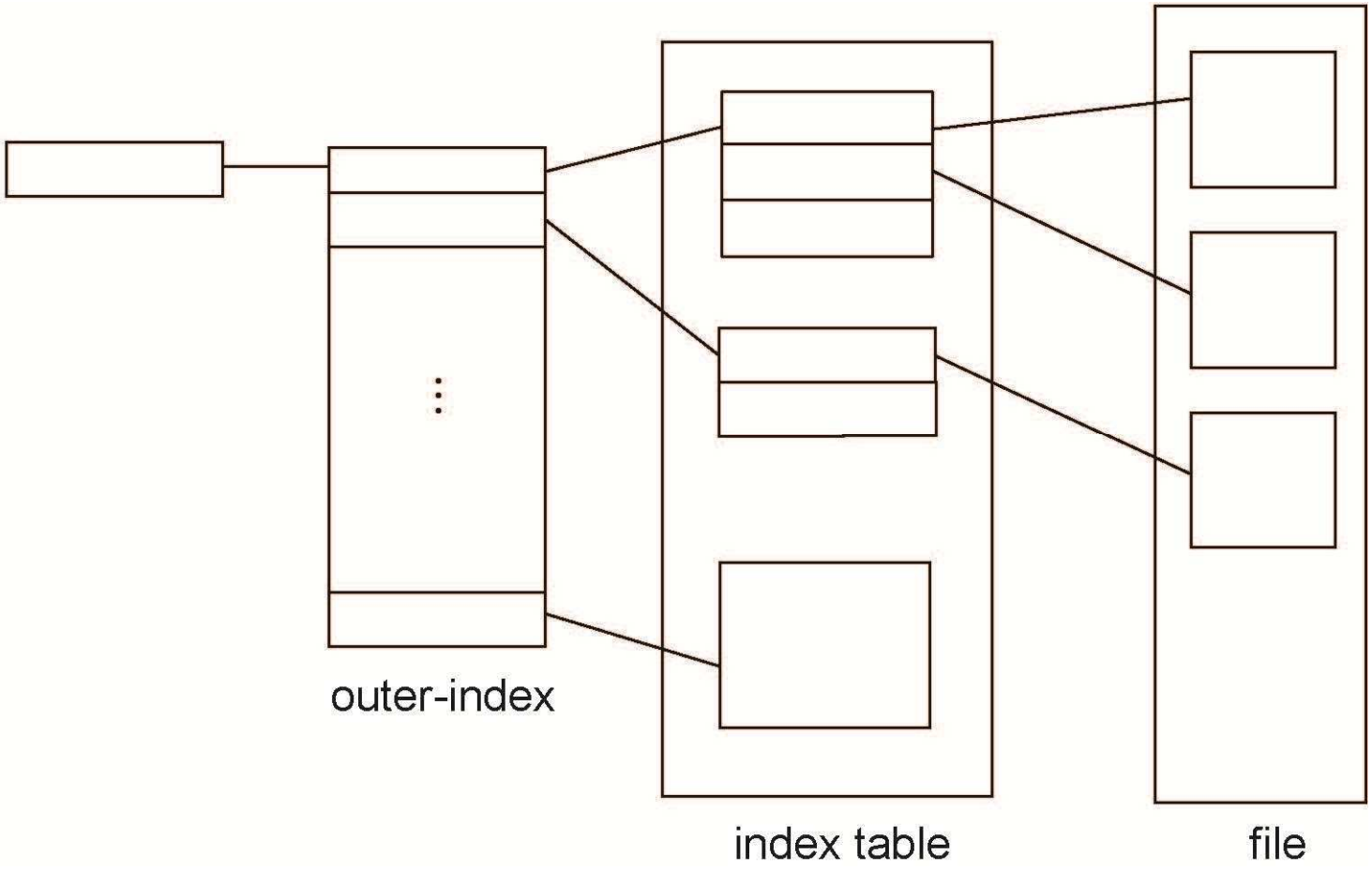
Pros

- Simplify seeks
- Random access
- Dynamic access without external fragmentation
- May link several index blocks together (for large files)

Cons

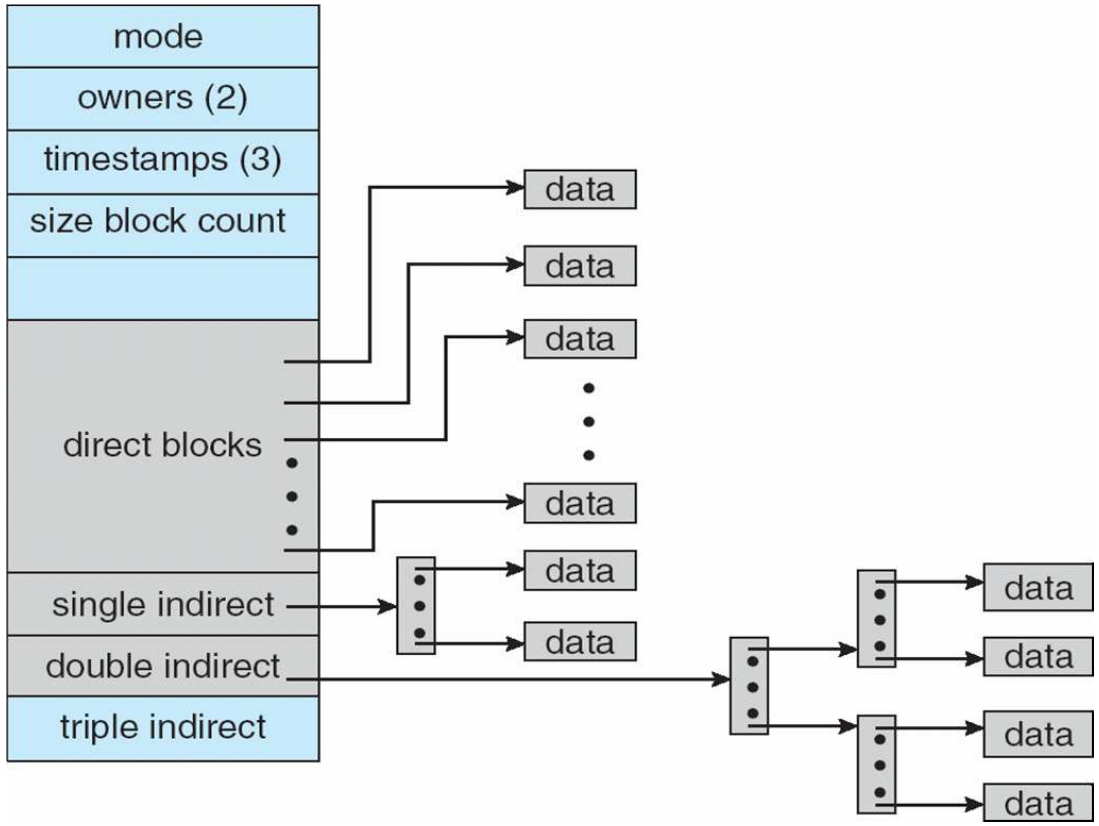
- Overhead of index block

Indexed Allocation – Mapping



Combined Scheme: UNIX UFS

- 4K bytes per block, 32-bit addresses



Question

- Consider a file system that uses inodes to represent files. Disk blocks are 8 KB in size, and a pointer to a disk block requires 4 bytes. This file system has 12 direct disk blocks, as well as single, double, and triple indirect disk blocks. What is the maximum size of a file that can be stored in this file system?

Answer

12 direct disk blocks *8KB = 96KB

1 single disk block = 2048 * 8KB = 16384KB

1 double disk block = 2048^2 * 8 KB = 33554432 KB

1 triple disk block = 2048^3 * 8 KB = 68719476736 KB

Total = 64.03 TB

In-Class Work 8

Consider a UNIX file system with 10 direct pointers, 1 indirect pointer, 1 double-indirect pointer, and 1 triple-indirect pointer in the i-node. Assume that disk blocks are 4K bytes and that each pointer to a disk block requires 4 bytes.

(1) What is the largest possible file that can be supported with this design? (show your work as expression, no need to calculate the numeric result)

(2) Assume that the operating system has already read your file into the main memory. How many disk reads are required to read the data block 800 into memory? Explain your answer.

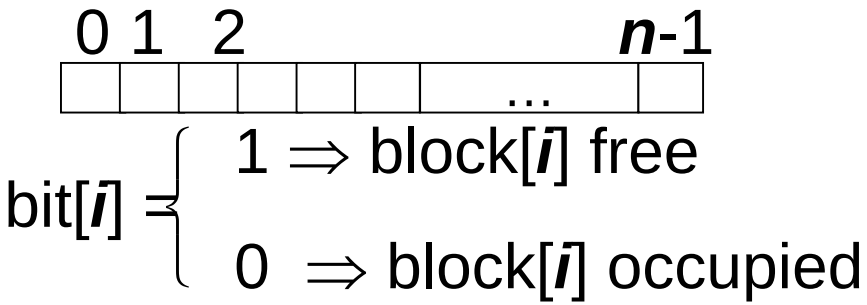
Answer

(1) 10*4KB + 1024*4KB + 1024*1024*4KB + 1024*1024*1024*4KB

(2) Data block number 800 falls in the set of blocks accessible from the single indirect block (block number 10-1033). One disk read is required to read the indirect block, one disk read is required to read the actual data, for a total of two disk reads.

Free-Space Management (1)

- File system maintains **free-space list** to track available blocks/clusters
  - (Using term "block" for simplicity)
- **Bit vector** or **bit map** (***n*** blocks)



Block number calculation (first free block)

(number of bits per word) *
(number of 0-value words) +
offset of first 1 bit

CPUs have instructions to return offset within word of first "1" bit

Free-Space Management (2)

- Bit map requires extra space
  - Example:

    block size = 4KB = 2^{12} bytes

    disk size = 2^{40} bytes (1 terabyte)

    $\boldsymbol{n}$ = $2^{40}/2^{12} = 2^{28}$ bits (or 256MB)

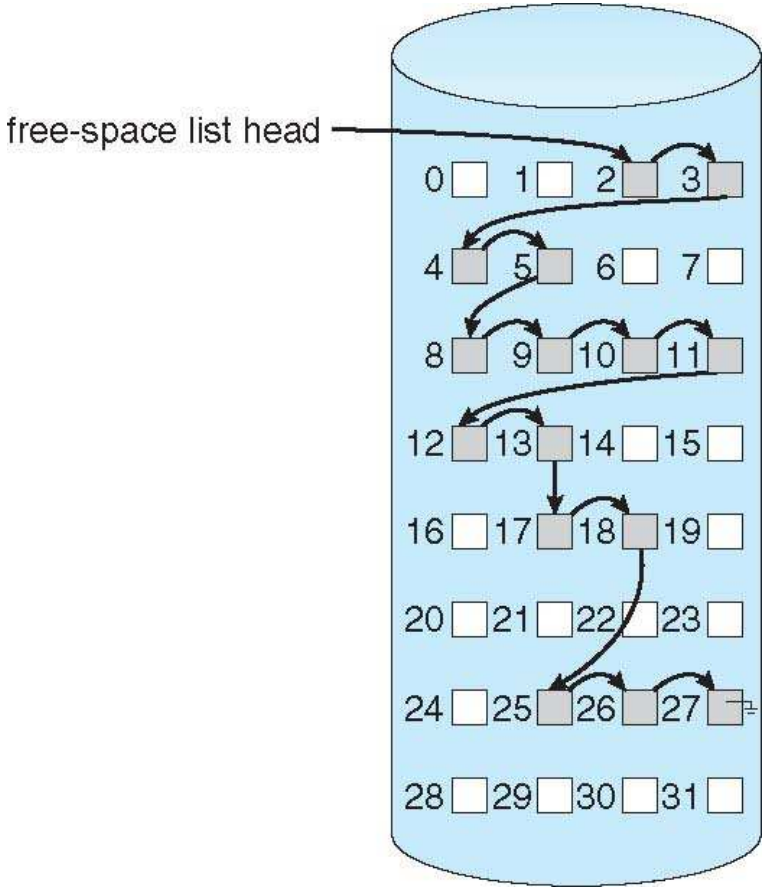
    if clusters of 4 blocks -> 32MB of memory

- Easy to get contiguous files

Linked Free Space List on Disk

Linked list (free list)

- Cannot get contiguous space easily
- No waste of space
- No need to traverse the entire list (if # free blocks recorded)

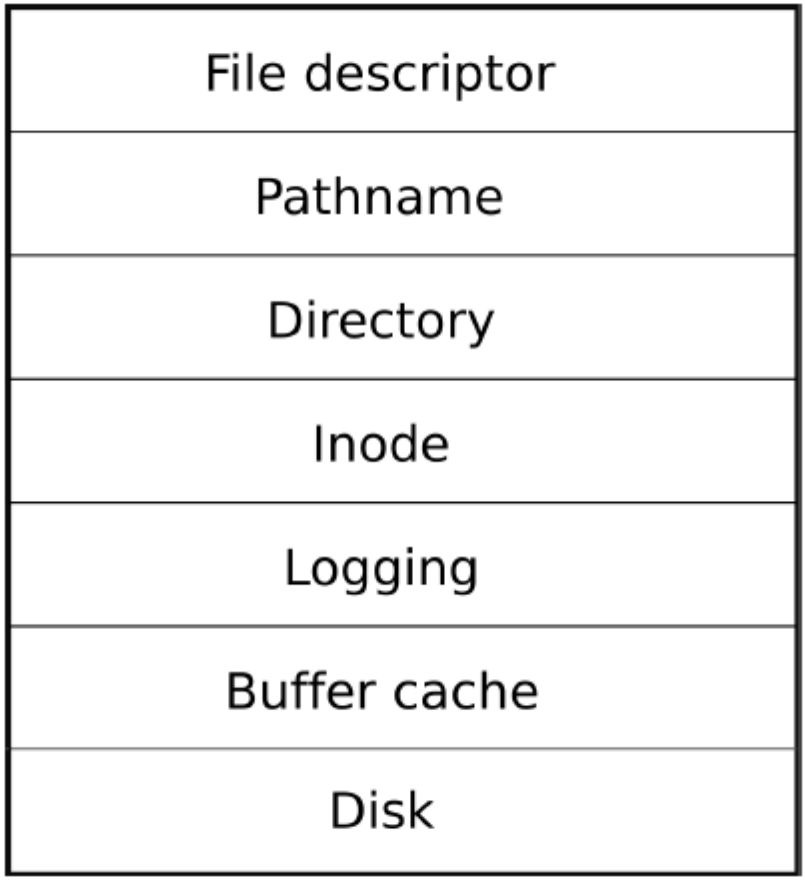


NEW MEXICO TECH
SCIENCE · ENGINEERING · RESEARCH · UNIVERSITY

Free-Space Management (3)

- Grouping
  - Modify linked list to store address of next *n-1* free blocks in first free block, plus a pointer to next block that contains free-block-pointers (like this one)
  - Find the addresses of a large number of free blocks quickly
- Counting
  - Because space is frequently contiguously used and freed, with contiguous-allocation algorithm, extents, or clustering
    - Keep address of first free block and count of following free continguous blocks
    - Free space list then has entries containing addresses and counts
    - The entries can be stored in a balanced tree for efficient lookup, insertion, and deletion.

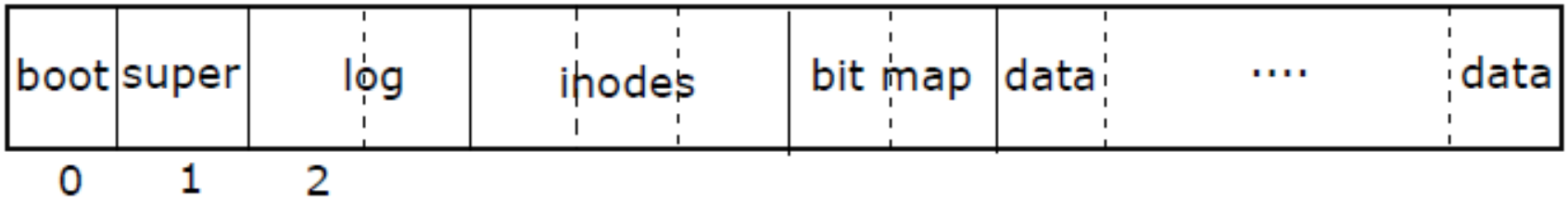
Layers of xv6 File System



Chapter 6 of xv6 book

NEW MEXICO TECH
SCIENCE · ENGINEERING · RESEARCH · UNIVERSITY

xv6 File System Structure



fs.h, fs.c, mkfs.c

```
struct superblock {
  uint size;         // Size of file system image (blocks)
  uint nblocks;      // Number of data blocks
  uint ninodes;      // Number of inodes.
  uint nlog;         // Number of log blocks
  uint logstart;     // Block number of first log block
  uint inodestart;   // Block number of first inode block
  uint bmapstart;    // Block number of first free map block
};
```

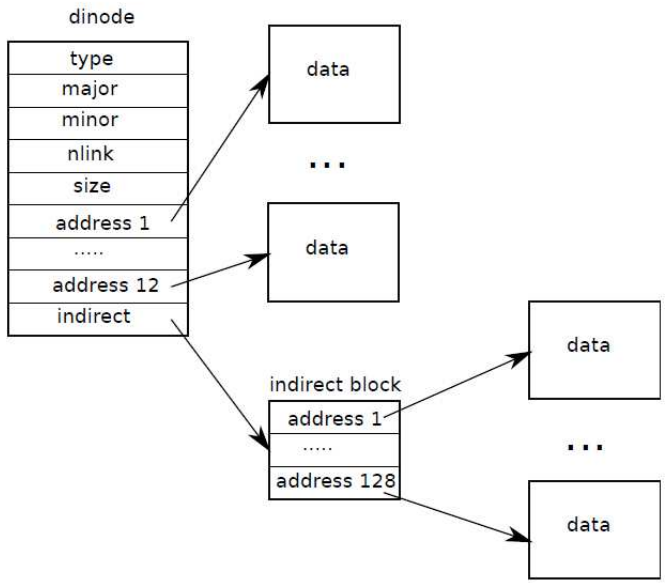
NEW MEXICO TECH
SCIENCE · ENGINEERING · RESEARCH · UNIVERSITY

On-disk inode structure

```
struct dinode {
  short type;           // File type
  short major;          // Major device number (T_DEV only)
  short minor;          // Minor device number (T_DEV only)
  short nlink;          // Number of links to inode in file system
  uint size;            // Size of file (bytes)
  uint addrs[NDIRECT+1];   // Data block addresses
};
```

NDIRECT = 12

NINDIRECT = BSIZE/4 = 512/4 = 128



NEW MEXICO TECH
SCIENCE · ENGINEERING · RESEARCH · UNIVERSITY