

CS 370: OPERATING SYSTEMS [FILE SYSTEMS]

An ode to the mighty iNode

A file control block
a keeper of the lore
a file's heart and so much more

tracing information about
blocks that
comprise a file

not just the *what*
but also the *how*
and the *where*

Shrideep Pallickara

Computer Science
Colorado State University

COMPUTER SCIENCE DEPARTMENT



1

Frequently asked questions from the previous class survey

- Who decides the size of a block?
- What if the file does not take up a whole block?
- Is the BCB at a standard location on every disk?
- What is metadata? Could you give an example?
- How do you “erase” data from CDs, DVDs, Blu-Rays?
 - Can you “re-burn” disks to store new data?
 - How does a DVD wear out even when someone is careful?
- If we have two different files with the same name in different directories, how is it handled?



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.2

2

Topics covered in this lecture

- Block Allocations
 - Indexed allocations
 - Linked allocations
- File Systems
 - Unix File System/FFS
 - FAT-32



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.3

3

What's in a name? That which we call a rose
By any other name would smell as sweet.

—Juliet
Romeo and Juliet (II, ii, 1-2)
(Shakespeare)

NOMENCLATURE

COMPUTER SCIENCE DEPARTMENT



4

Terminology

- Storage hardware arranges data in **sectors** (for magnetic disk) or **pages** (for flash)
- File systems often group together a *fixed number* of disk sectors or flash pages into a larger allocation unit called a **block**.
 - E.g.: format file system to run on a disk with 512b sectors to use 4 KB blocks
- Windows FAT and NTFS refer to blocks as **clusters**
- **File Control Block** (FCBs) organize info about blocks comprising a file
 - iNode in UFS and MFT Record in NTFS; Master File Table (MFT)



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.5

5

Allocation methods: Objective and approaches

- How to allocate space for files such that:
 - Disk space is utilized effectively
 - File is accessed **quickly**
- Major Methods
 - Contiguous
 - Linked
 - Indexed



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.6

6

INDEXED ALLOCATIONS

COMPUTER SCIENCE DEPARTMENT



7

Indexed allocations

- Bring all pointers together into one location
 - **index block**
- Each file has its **own** index block
 - Directory contains address of the index block



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.4

8

Indexed allocation supports direct access without external fragmentation

- Every disk block can be utilized
 - **No external fragmentation**
- Space wasted by pointers *is generally higher* than linked listed allocations
 - Example: File has two blocks
 - Linked listed allocations: 2 pointers are utilized
 - Indexed allocations: Entire index block must be allocated



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.9

9

iNODES

COMPUTER SCIENCE DEPARTMENT



COLORADO STATE UNIVERSITY

10

inode

- **Fixed-length** data structure
 - One per file
- Contains information about
 - **File attributes**
 - Size, owner, creation/modification time etc.
 - **Disk addresses**
 - File blocks that comprise file



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.11

11

inode

- The inode is used to encapsulate information about a large number of file blocks
- For e.g.
 - Block size = 8 KB, and file size = 8 GB
 - There would be a million file-blocks
 - inode will store info about the **pointers to these blocks**
 - inode allows us to access info for *all* these blocks
 - Storing pointers to these file blocks also takes up storage



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.12

12

Managing information about data blocks in the inode

- The first **few** pointers to the data blocks of the file stored in the inode
- If the file is large: **Indirect** pointer
 - To a block of pointers that point to additional data blocks
- If the file is larger: **Double indirect** pointer
 - Pointer to a block of indirect pointers
- If the file is huge: **Triple indirect** pointer
 - Pointer to a block of double-indirect pointers



COLORADO STATE UNIVERSITY

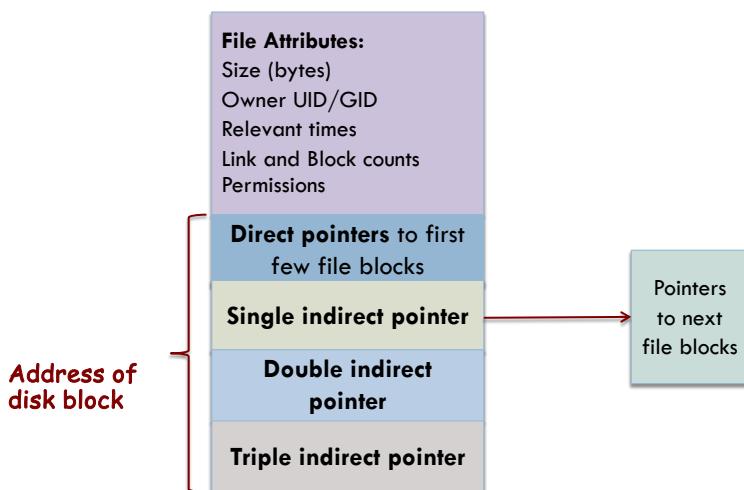
Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.13

13

Schematic structure of the inode



COLORADO STATE UNIVERSITY

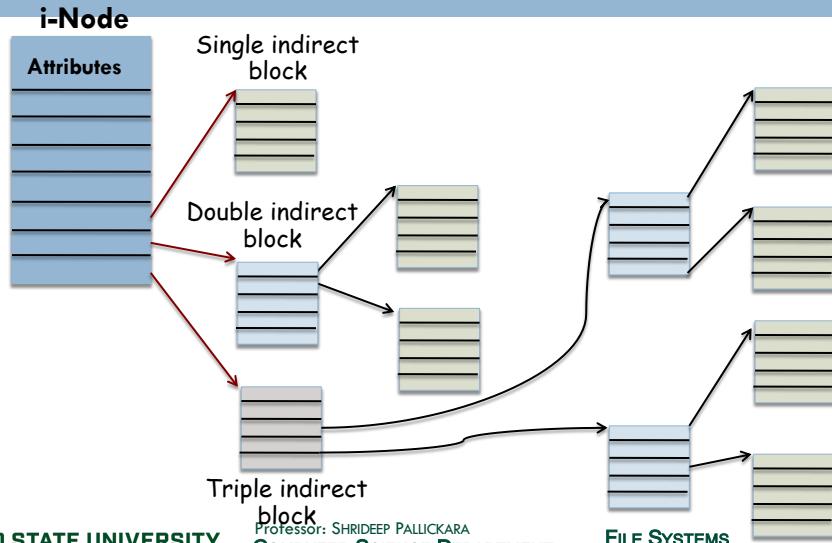
Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.14

14

i-Node: How the pointers to the file blocks are organized



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

L28.15

15

Disk Layout in traditional UNIX systems



An integer number of inodes fit in a single data block



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.16

16

Super Block describes the state of the file system

- Total size of partition
- Block size and number of disk blocks
- Number of inodes
- List of free blocks
- inode number of the root directory

- Destruction of super block?
 - Will render file system unreadable



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.17

17

A linear array of inodes follows the data block

- inodes are numbered from 1 to some **max**
- Each inode is identified by its inode number
 - inode number contains info needed to **locate** inode on the disk
 - Users think of files as filenames
 - UNIX thinks of files in terms of inodes



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

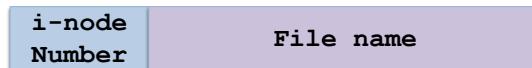
FILE SYSTEMS

L28.18

18

UNIX directory structure

- Contains only file names and the corresponding inode numbers



- Use `ls -i` to retrieve inode numbers of the files in the directory



COLORADO STATE UNIVERSITY

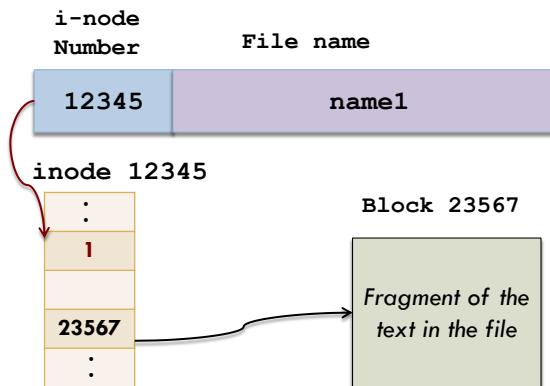
Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.19

19

Directory entry, inode and data block for a simple file



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.20

20

Looking up path names in UNIX

Example: /usr/tom/mbox

Root directory

1	.
1	..
4	bin
7	dev
14	lib
9	etc
6	usr
8	tmp

Looking up usr
yields i-node 6

i-node 6
is for /usr
Mode, size
.. attributes
132

i-node 6 says
that /usr is in
block 132

Block 132 is
/usr directory

6	.
1	..
19	bob
30	eve
51	jim
26	tom
45	zac

/usr/tom is in
i-node 26

i-node 26
is /usr/tom
Mode, size
.. attributes
406

i-node 26 says
that /usr/tom
is in block 406

Block 406 is
/usr/tom dir

26	.
6	..
64	grants
92	dev
60	mbox
81	docs
17	src

/usr/tom/mbox
is in i-node 60



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.21

21

Advantages of directory entries that have name and inode information

- Changing filename only requires changing the directory entry
- Only 1 physical copy of file needs to be on disk
 - File may have several names (or the same name) in different directories
- Directory entries are small
 - Most file info is kept in the inode



COLORADO STATE UNIVERSITY

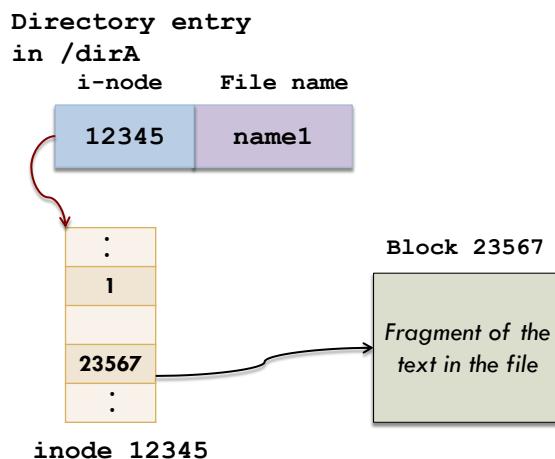
Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.22

22

Two hard links to the same file



COLORADO STATE UNIVERSITY

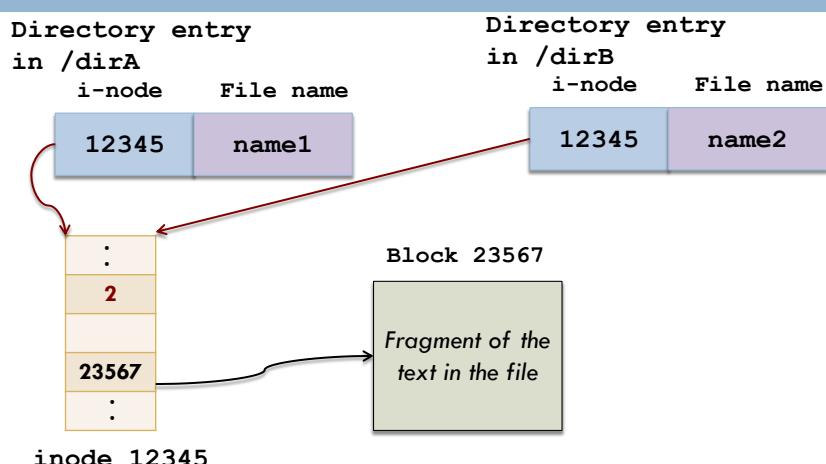
Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.23

23

Two hard links to the same file



COLORADO STATE UNIVERSITY

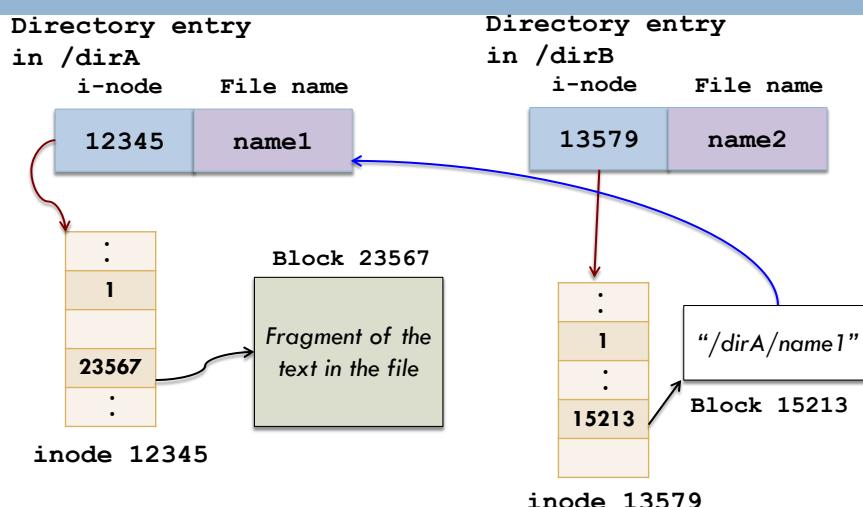
Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.24

24

File with a symbolic link



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.25

25

Maximum size of your hard disk (8 KB blocks and 32-bit pointers)

- 32-bit pointers can address 2^{32} blocks
- At 8 KB per-block
 - Hard disk can be $2^{13} \times 2^{32} = 2^{45}$ bytes (32 TB)



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.26

26

The case for larger block sizes

- Larger partitions for a fixed pointer size
- Retrieval is more efficient
 - Better system throughput
- Problem
 - Internal fragmentation



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.27

27

Limitations of a file system based on inodes

- File **must fit** in a single disk partition
- Partition size and number of files are **fixed** when system is set up



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.28

28

inode preallocation and distribution

- inodes are **preallocated** on a volume
 - Even on empty disks % of space lost to inodes
- Preallocating inodes and spreading them
 - Improves performance
- Keep file's data block **close** to its inode
 - Reduce seek times



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.29

29

Checking up on the iNodes: The df -i command (disk free)

- inode statistics for a given set of file systems
 - Total, free and used inodes

```
df -i /s/bach/*
Filesystem      Inodes  IUsed  IFree   IUse%
/dev/cciss/c0d1p1  12746752 948362 11798390  8%
/dev/cciss/c0d2p1  10240000 150436 10089564  2%
/dev/cciss/c0d3p1  10240000 812727 9427273  8%
/dev/cciss/c0d4p1  10240000 930080 9309920 10%
/dev/cciss/c0d5p1  10240000 496744 9743256  5%
/dev/cciss/c0d6p1  10240000 167900 10072100  2%
/dev/cciss/c0d7p1  10240000 748709 9491291  8%
/dev/cciss/c0d8p1  12681216 760002 11921214  6%
/dev/cciss/c0d9p1  12681216 394165 12287051  4%
```



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

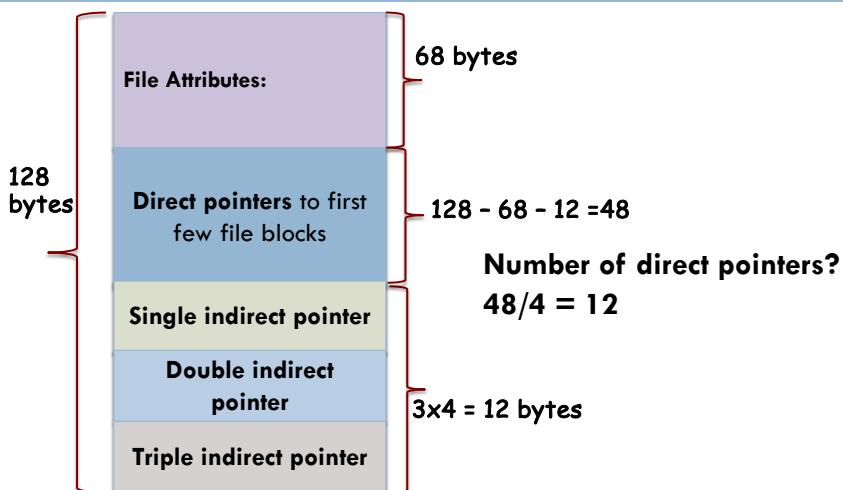
FILE SYSTEMS

L28.30

30

inode: A quantitative look

BLOCK Size = 8 KB and Pointers = 4 bytes



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.31

31

inode: A quantitative look

BLOCK Size = 8 KB and Pointers = 4 bytes

- 12 **direct** pointers to file blocks
- Each file block = 8KB
- Size of file that can be represented with direct pointers
 - $12 \times 8 \text{ KB} = 96 \text{ KB}$



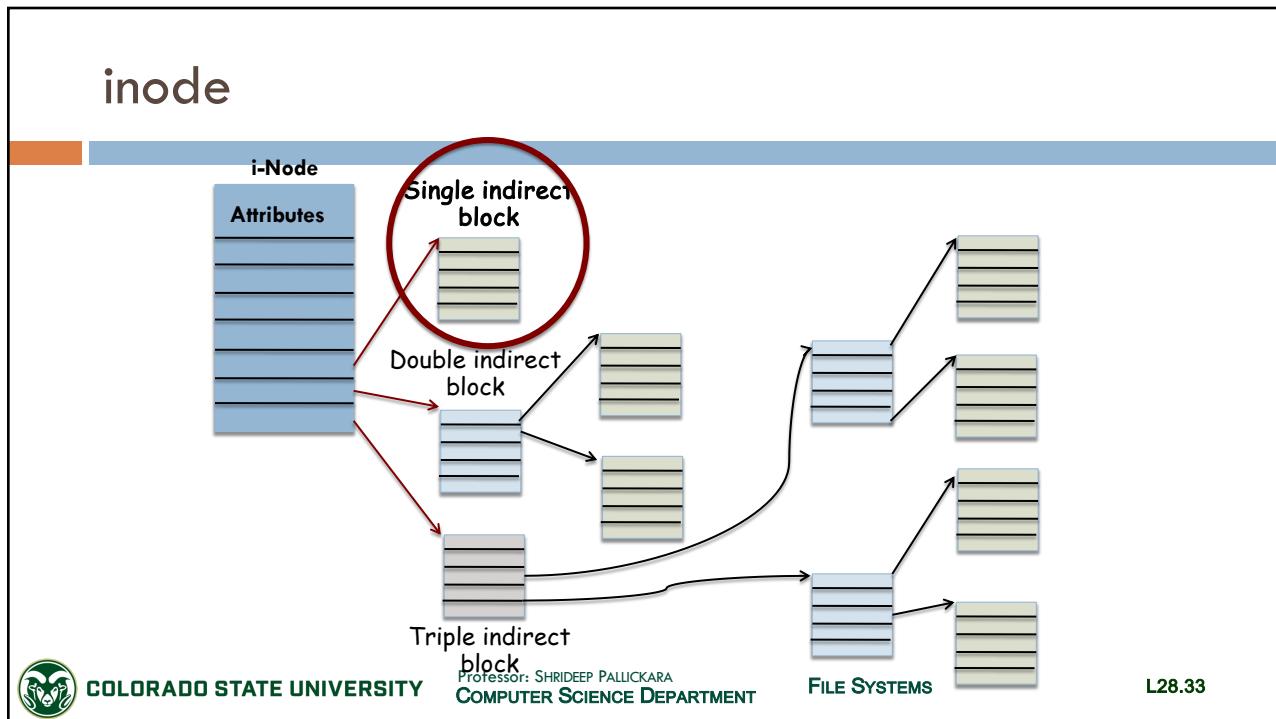
COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

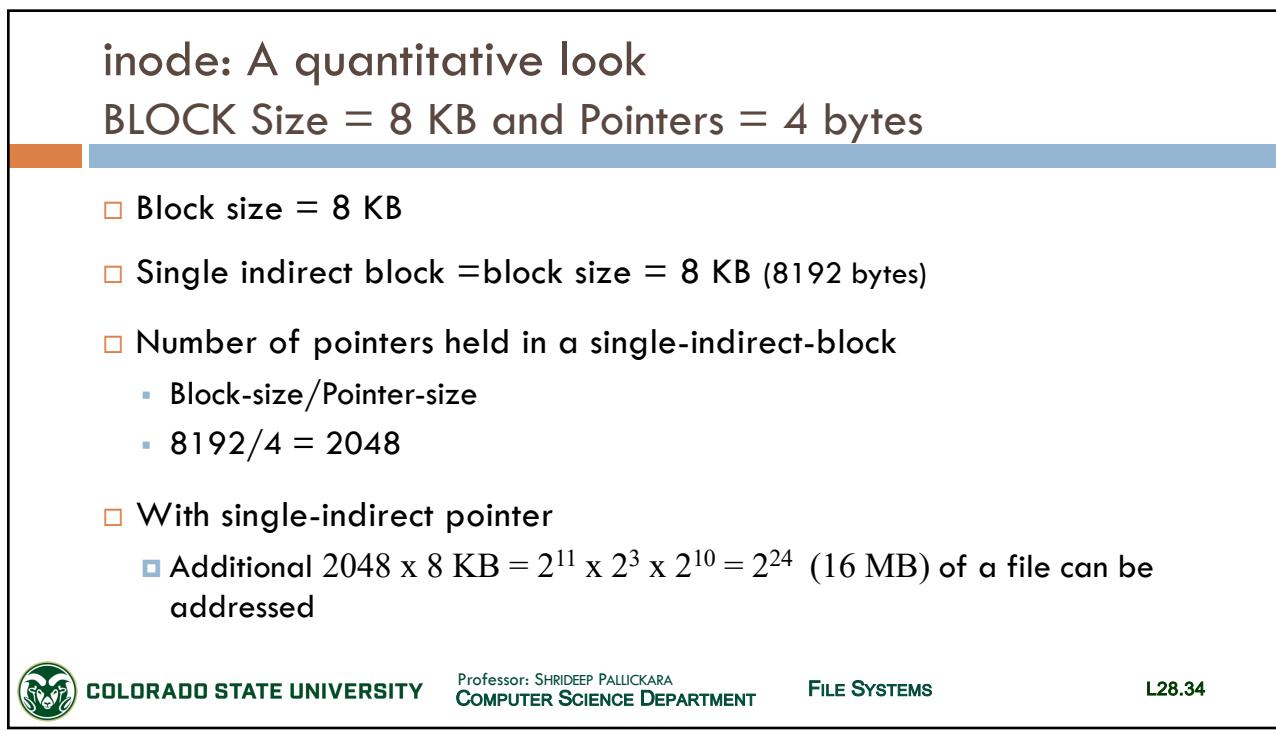
FILE SYSTEMS

L28.32

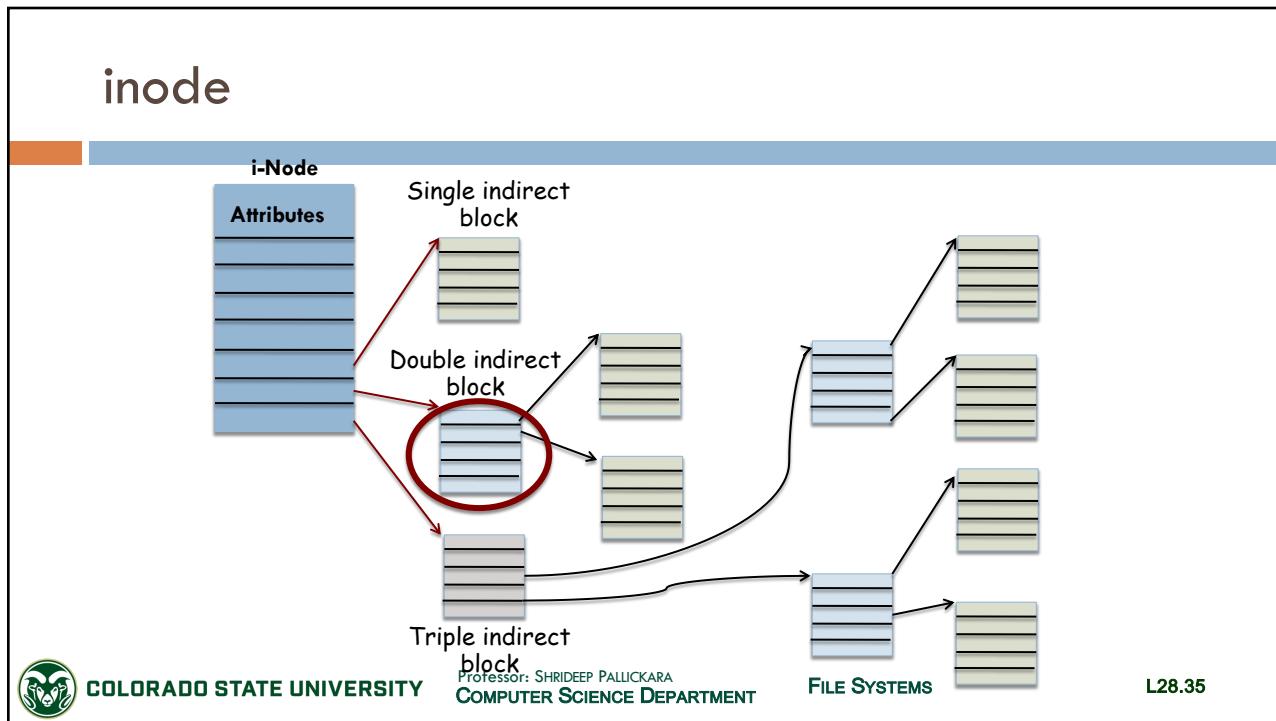
32



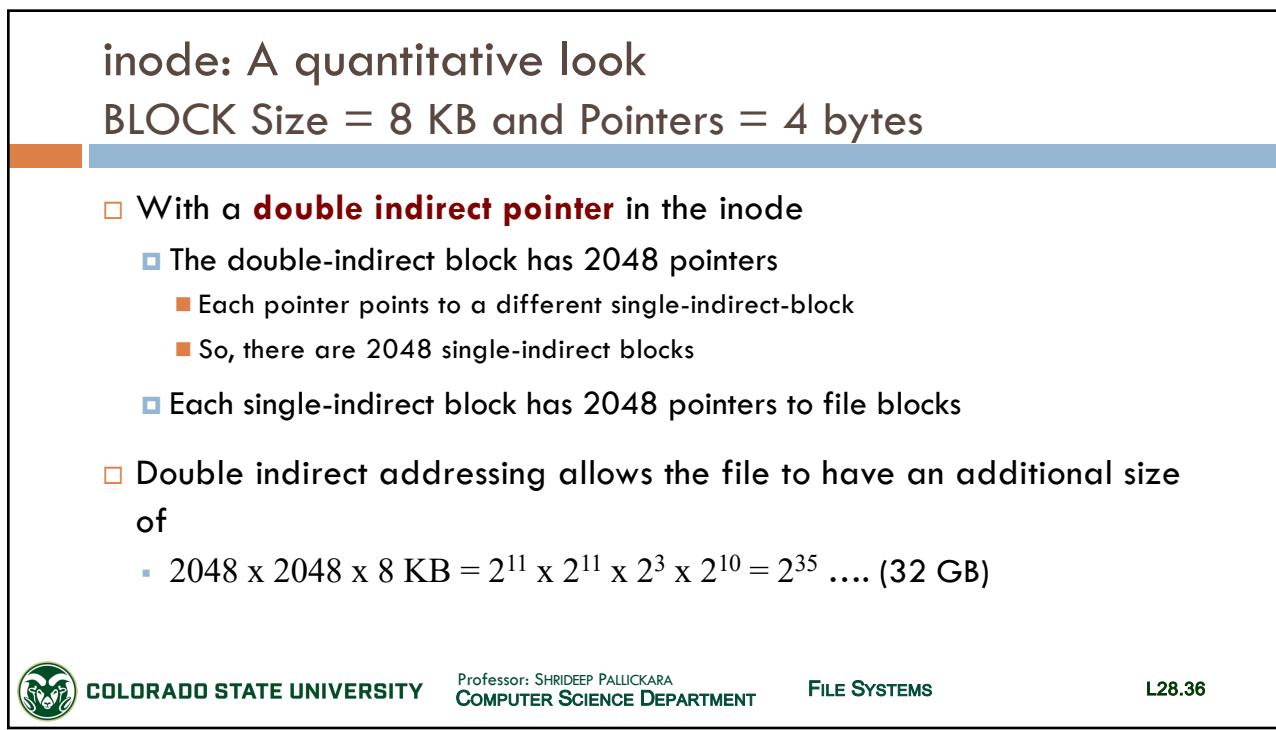
33



34



35



36

inode: A quantitative look

BLOCK Size = 8 KB and Pointers = 4 bytes

- **Triple indirect addressing**
 - Triple indirect block points to 2048 double indirect blocks
 - Each double indirect block points to 2048 single indirect blocks
 - Each single direct block points to 2048 file blocks
- Allows the file to have an additional size of
 - $2048 \times 2048 \times 2048 \times 8 \text{ KB} = 2^{11} \times 2^{35} = 2^{46} (64 \text{ TB})$



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.37

37

Limits of triple indirect addressing

- In our example:
 - There can be $2048 \times 2048 \times 2048$ data blocks
 - i.e., $2^{11} \times 2^{11} \times 2^{11} = 2^{33}$
 - Pointers would need to be longer than 32-bits to fully address this storage



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.38

38

What if we increase the size of the pointers to 64-bits (data block is still 8 KB) ?

- What is the maximum size of the file that we can hold?
- 8 KB data block can hold $(8192/8) = 1024$ pointers
- **Single indirect** can add
 - $1024 \times 8 \text{ KB} = 2^{10} \times 2^3 \times 2^{10} = 2^{23}$ (8MB) of additional bytes to the file



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.39

39

What if we increase the size of the pointers to 64-bits (data block is still 8 KB)?

- **Double indirect** addressing allows the file to have an additional size of
 - $1024 \times 1024 \times 8 \text{ KB} = 2^{10} \times 2^{23} = 2^{33}$ (8 GB)
- **Triple indirect** addressing allows the file to have an additional size of
 - $1024 \times 1024 \times 1024 \times 8 \text{ KB} = 2^{10} \times 2^{33} = 2^{43}$ (8 TB)



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

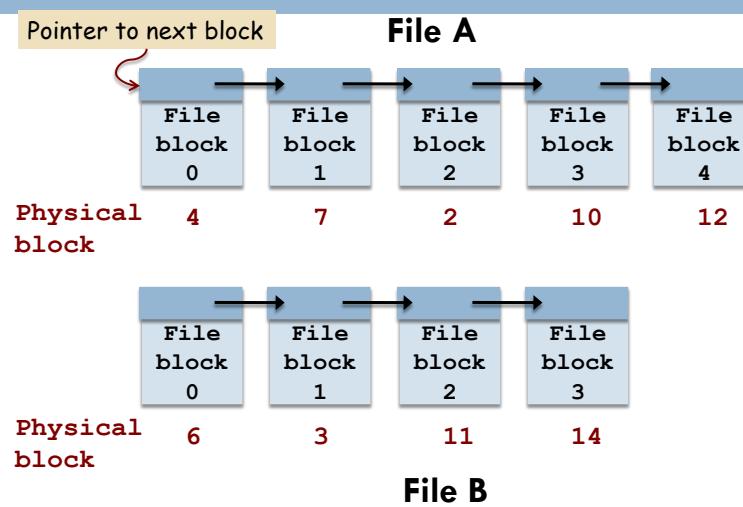
L28.40

40



41

Linked Allocation: Each file is a linked list of disk blocks



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.42

42

Linked List Allocations: Advantages

- **Every** disk block can be used
 - No space is lost in external fragmentation
- Sufficient for directory entry to merely store *disk address of first block*
 - Rest can be found starting there



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.43

43

Linked List Allocation: Disadvantages

- Used effectively only for sequential accesses
 - Extremely **slow random access**
- Space in each block set aside for pointers
 - Each file requires *slightly more space*
- Reliability
 - What if a pointer is lost or damaged?



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.44

44

Linked List Allocations: Reading and writing files is much less efficient

- Amount of data storage in block is no longer a **power of two**
 - Pointer takes up some space
- Peculiar size** is less efficient
 - Programs read/write in blocks that is a power of two



COLORADO STATE UNIVERSITY

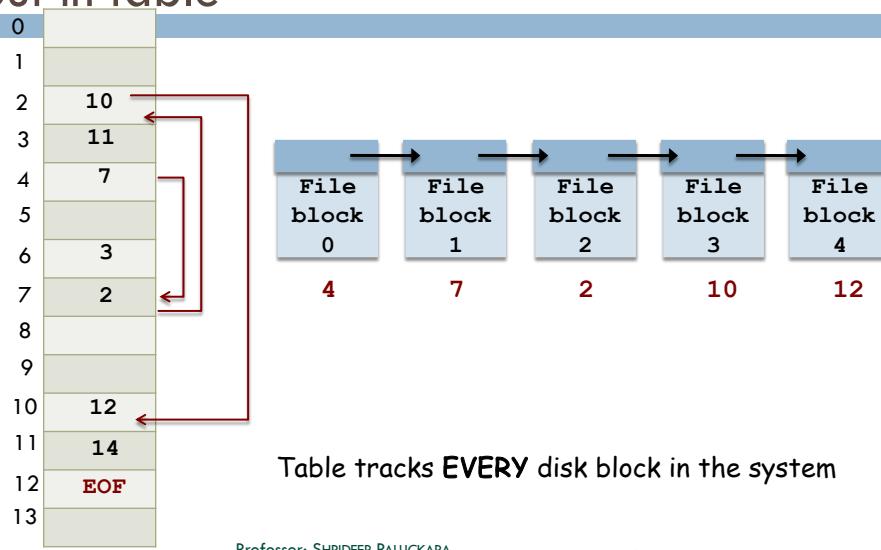
Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.45

45

Linked list allocation: Take pointers from disk block and put in table



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.46

46

Linked list allocation using an index/table

- **Entire** disk block is available for data
- Random access is much easier
 - Chain must still be followed
 - But this chain could be *cached in memory*
- MS-DOS and OS/2 operating systems
 - Use such a file allocation table (FAT)



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.47

47

Linked list allocation using an index: Disadvantages

- Table must be cached **in memory** for efficient access
- A large disk will have a large number of data blocks
 - Table consumes a large amount of physical memory



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.48

48

FAT-32: Linked List Allocations using an index

- The Microsoft File Allocation Table (FAT) file system was first implemented in the late 1970s
 - Was the main file system for MS-DOS and early versions of Microsoft Windows
- FAT-32, which supports volumes with up to 2^{28} blocks and files with up to $2^{32} - 1$ bytes



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.49

49

The FAT file system is named for its file allocation table

- An **array** of 32-bit entries in a **reserved** area of the volume
- Each file in the system corresponds to a linked list of FAT entries
 - Each FAT entry containing a pointer to the next FAT entry of the file (or a special “end of file” value)
- The FAT has **one entry for each block in the volume**, and the file’s blocks are the blocks that correspond to the file’s FAT entries:
 - If FAT entry i is the j^{th} FAT entry of a file, then storage block i is the j^{th} data block of the file



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.50

50

File numbers

- ❑ Directories map file names to **file numbers**
- ❑ In the FAT file system, a file's number is the index of the file's first entry in the FAT



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.51

51

The FAT is also used for free space tracking

- ❑ If data block i is free, then $\text{FAT}[i]$ contains 0
- ❑ Thus, the file system can find a free block by scanning through the FAT to find a zeroed entry



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.52

52

The FAT file system is widely used because it is simple and supported by many operating systems

- Many flash storage USB keys and camera storage cards use FAT
 - Allowing them to be read and written by almost any computer running almost any modern operating system
- Variations of the FAT file system are even used by applications for organizing data within individual files
 - For example, .doc files produced by versions of Microsoft Word from 1997 to 2007 are actually compound documents with many internal pieces
 - The .doc format creates a FAT-like file system within the .doc file to manage the objects in the .doc file



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.53

53

FAT-32: LIMITATIONS

COMPUTER SCIENCE DEPARTMENT



COLORADO STATE UNIVERSITY

54

FAT-32 limitations: No support for hard links

- ❑ FAT represents each file as a linked list of 32-bit entries in the file allocation table
 - ❑ This representation does not include room for any other file metadata
- ❑ Instead, **file metadata is stored with directory entries** with the file's name
 - ❑ This approach demands that each file be accessed via exactly one directory entry, ruling out multiple hard links to a file



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.55

55

FAT-32 Limitations: Volume and File size

- ❑ FAT table entries are 32 bits, but the top four bits are reserved
- ❑ Thus, a FAT **volume** can have at most **2^{28} blocks**
- ❑ With 4 KB blocks, the maximum volume size is limited
 - ❑ E.g., 2^{28} blocks/volume \times 2^{12} bytes/block = 2^{40} bytes/volume = 1 TB
 - ❑ Block sizes up to 256 KB are supported, but they risk wasting large amounts of disk space due to internal fragmentation
- ❑ Similarly, **file sizes are encoded in 32 bits**, so no file can be larger than $2^{32} - 1$ bytes (just under 4 GB)



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.56

56

The contents of this slide-set are based on the following references

- *Kay Robbins & Steve Robbins. Unix Systems Programming, 2nd edition, Prentice Hall ISBN-13: 978-0-13-042411-2. [Chapter 4]*
- *Thomas Anderson and Michael Dahlin. Operating Systems Principles and Practice. 2nd Edition. Recursive Books. ISBN: 978-0985673529. [Chapter 13]*
- *Andrew S Tanenbaum and Herbert Bos. Modern Operating Systems. 4th Edition, 2014. Prentice Hall. ISBN: 013359162X/ 978-0133591620. [Chapter 4]*



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

FILE SYSTEMS

L28.57