THE UNIVERSITY *of* EDINBURGH
informatics

Operating Systems (INFR10079)

2022/2023 Semester 2

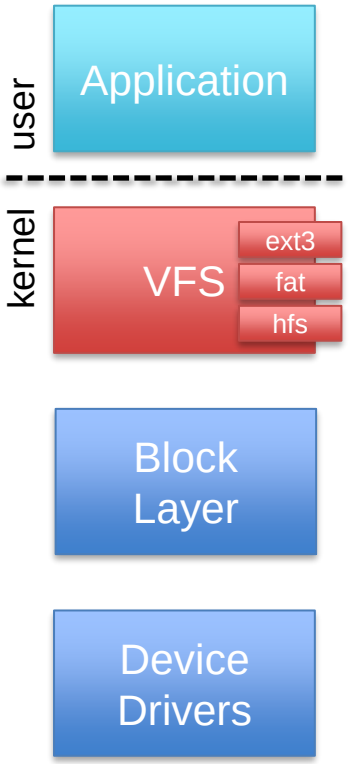
File System
(Format and Data Structures)

abarbala@inf.ed.ac.uk

Chapter 13, Chapter 14

The Original UNIX File System

- **Dennis Ritchie and Ken Thompson**, Bell Labs, 1969
- "UNIX rose from the ashes of a multi-organizational effort in the early 1960s to develop a dependable timesharing operating system" – **Multics**
  - Designed for a "workgroup" sharing a single system
  - Did its job well, prepared the foundations for today's OSes
- Today's Operating Systems **support multiple file system formats**
  - ext3, ext4, ntfs, fat, hfs, hfs+, etc.
  - **Virtual File System (VFS)**

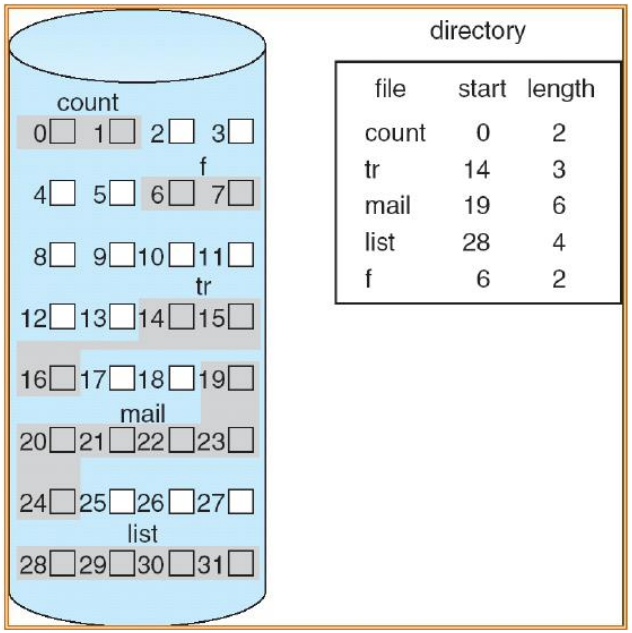


File System **Data Structures** (high-level)

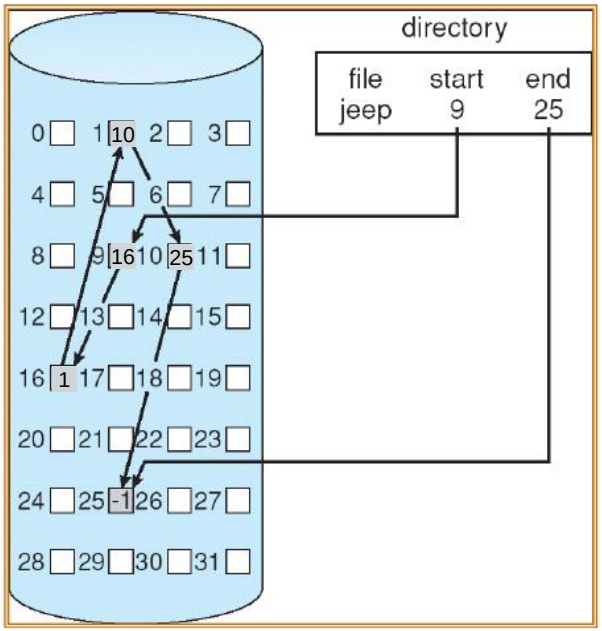
- In **Storage**
  - Boot control block
    - contain information needed by the system to boot an operating system
  - A volume control block
    - contains volume details, such as the number of blocks in the volume, the size of the blocks, a free-block count and free-block pointers, and a free-FCB count and FCB pointers
  - A directory structure
    - is used to organize the files
  - A per-file FCB
    - contains details about the file
    - It has a unique identifier number to allow association with a directory entry
- In **Memory**
  - Mount table
    - contains information about each mounted volume
  - Directory-structure cache
    - holds the directory information of recently accessed directories
  - The system-wide open-file table
    - contains a copy of the FCB of each open file, and other information
  - The per-process open-file table
    - contains pointers to the appropriate entries in the system-wide open-file table for all files the process has opened
  - Buffers
    - hold file-system blocks when they are being read/written to a FS

Disk Allocation Strategies

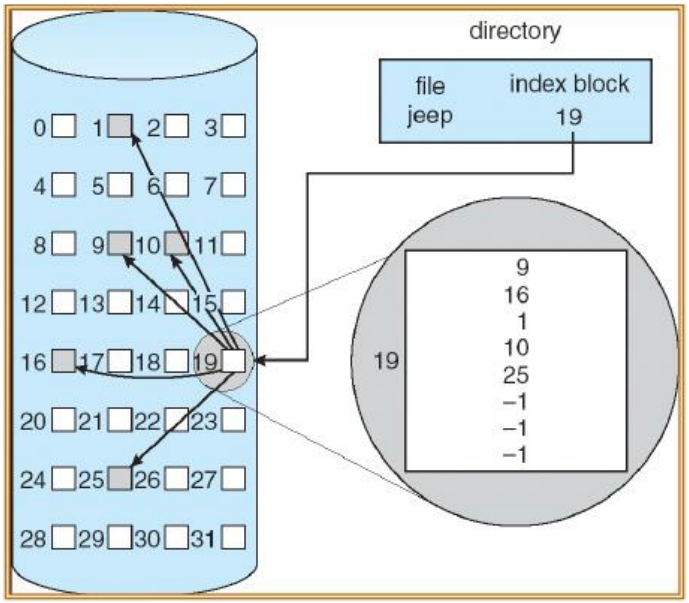
Contiguous Allocation



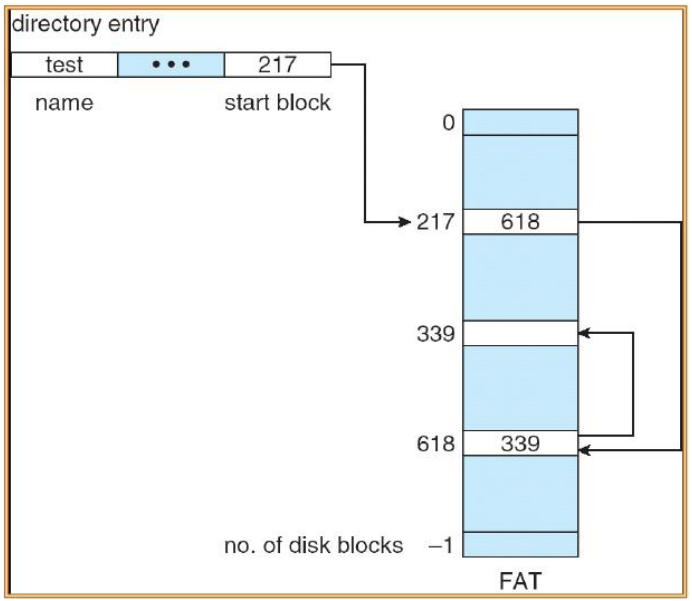
Linked Allocation



Indexed Allocation

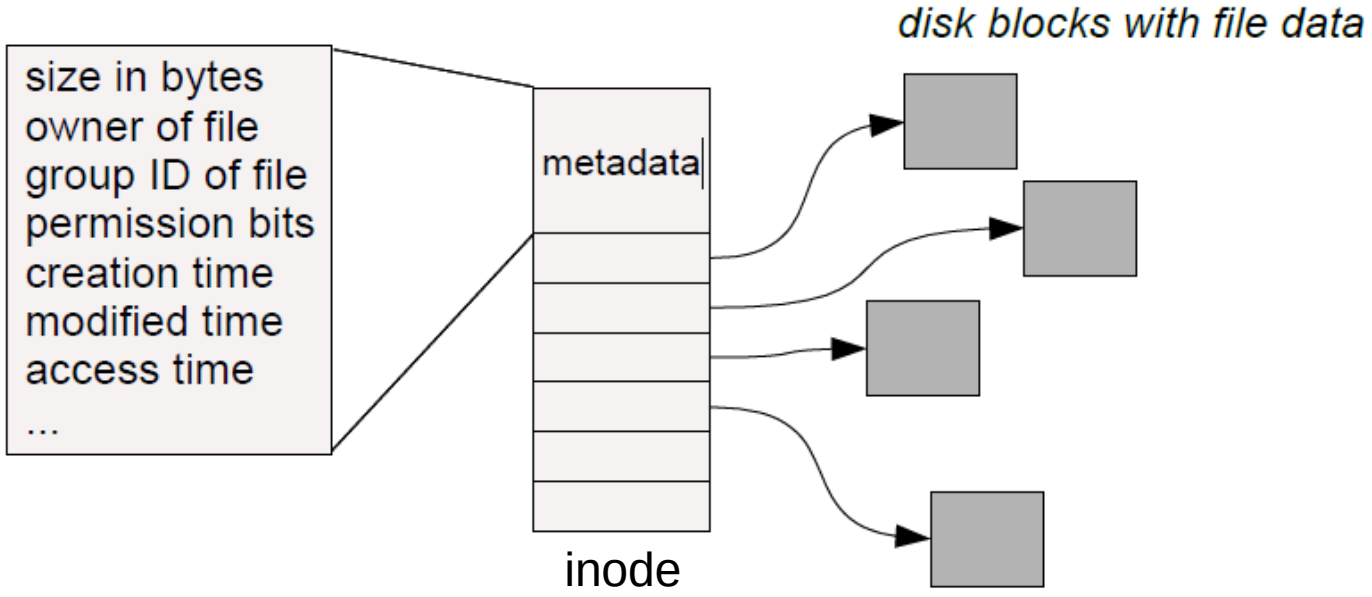


FAT



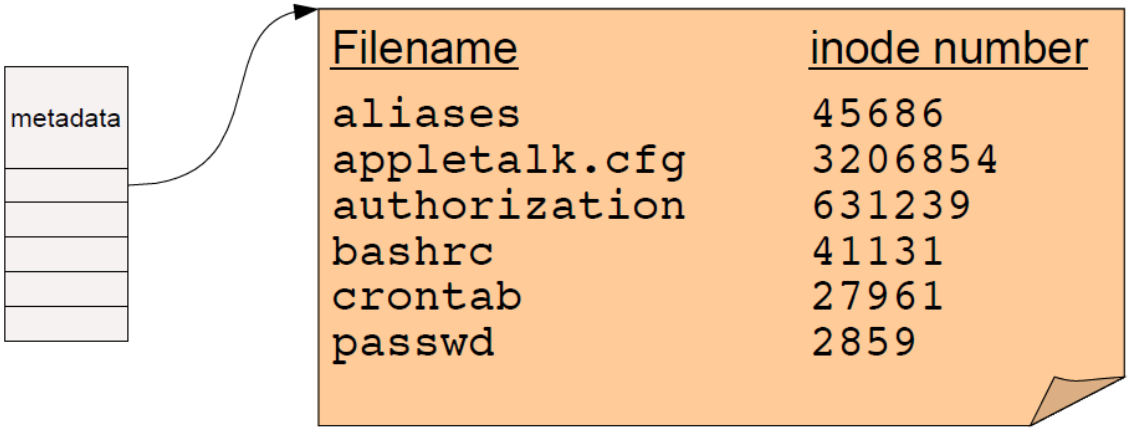
Inode File System: Indexed Allocation

- Every file and directory is **represented by an inode**
  - Inode == **index node**
  - Inode **is a number**
- **Inode** contains two kinds of information
  - **Metadata** describing file's owner, access rights, etc.
  - **Location** of the file's blocks on disk



Inode File System: Directories

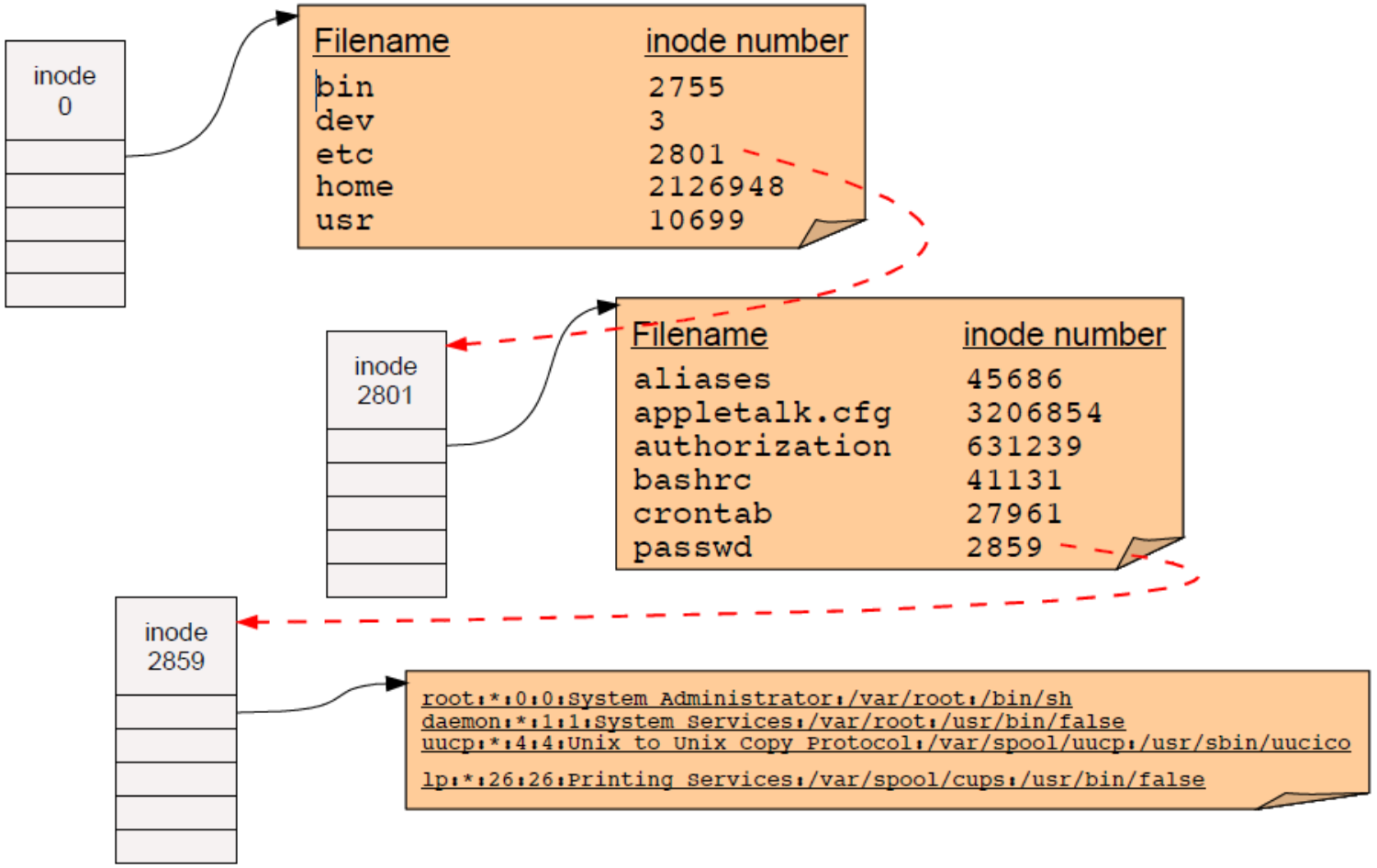
- A **directory** is a flat file of fixed-size entries
- Each entry consists of an inode number and a file name



  - These are the contents of the directory "file data" itself
    - **NOT the directory's inode**
  - Filenames (in UNIX) are not stored in the inode
- **Special** inodes
  - Root inode
  - Inode containing all bad blocks

Inode File System: Pathname Resolution

- To look up a pathname “/etc/passwd”, start at root directory and walk down chain of inodes...



Inode File System: More About Directories

- Directories map filenames to inode numbers
- **Multiple pointers** to the **same inode** in different directories
  - Or even the same directory with different filenames
  - Avoid saving the same file multiple times

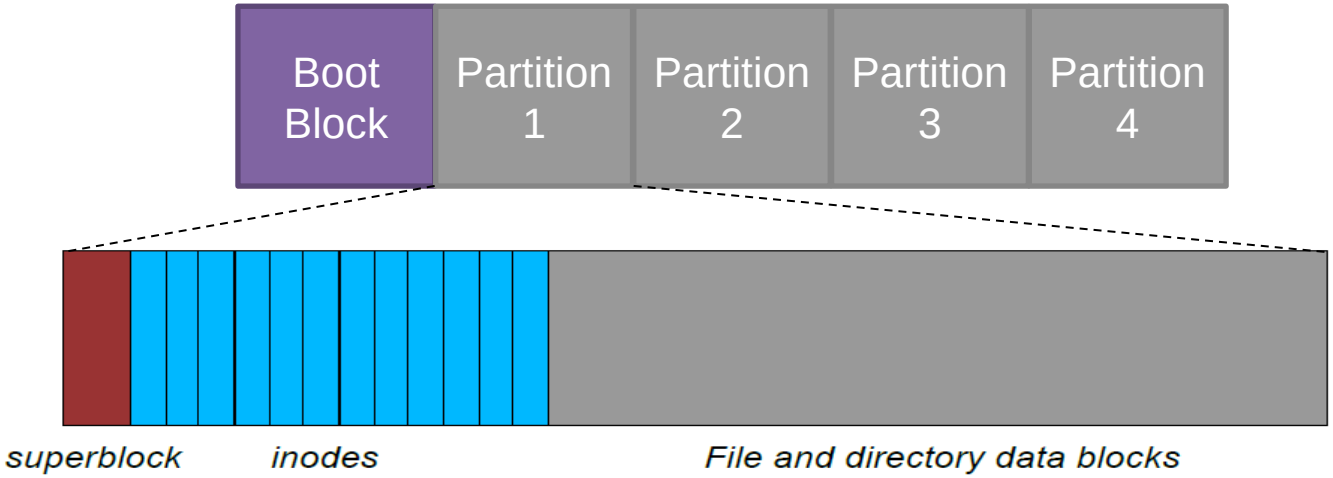
- In UNIX this is called a "hard link" and can be done using "ln"

```
bash$ ls -i /home/foo
287663 /home/foo          (This is the inode number of "foo")
bash$ ln /home/foo /tmp/foo
bash$ ls -i /home/foo /tmp/foo
287663 /home/foo
287663 /tmp/foo
```

  - "/home/foo" and "/tmp/foo" now refer to the same file on disk
    - **Not a copy!** But identical data no matter which filename is used

Inode File Systems: Data and Metadata Layout

- Superblock
  - Specifies boundaries of next areas
  - Contains head of freelists of inodes and file blocks
- inode area
  - Contains descriptors (inodes) for each file on the disk
  - All inodes are the same size
- File contents area
  - Fixed-size blocks



Inode File System: Locating Inodes on Disk

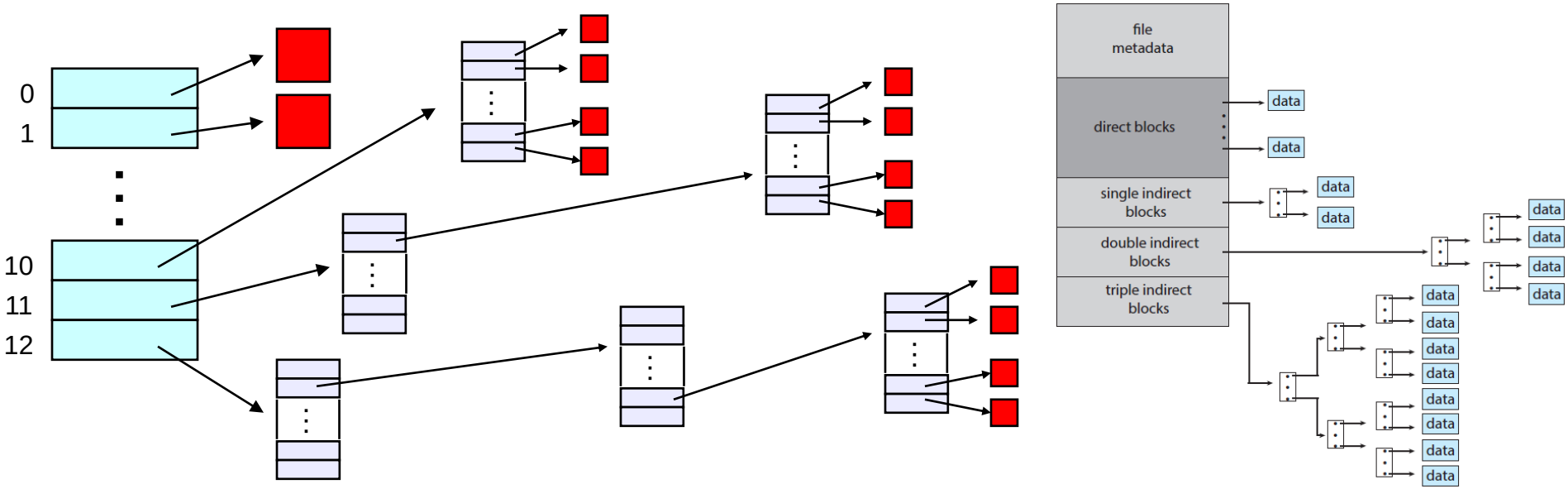
- Directories give the **inode number** of a file
  - How to **find the inode** itself on disk?
- Basic idea: Top part of filesystem contains *all* of the inodes

  - inode number is just the "index" of the inode
- Easy to compute the block address of a given inode
    - block_addr(inode_num) = block_offset_of_first_inode + (inode_num * inode_size)
  - This implies that a filesystem has a fixed number of potential inodes
    - This number is generally set when the filesystem is created
  - The superblock stores important metadata on filesystem layout, list of free blocks, etc.

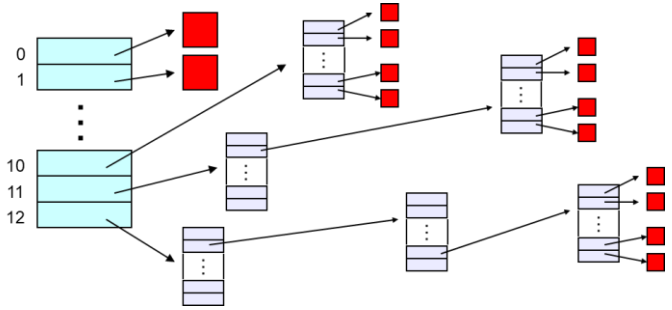
Inode File System: **Block List** in an Inode

- **Points to blocks** in the file contents area
- Able to represent **very small** and **very large** files

- (Example) Each inode contains 13 block pointers
  - First 10 are “direct pointers” (pointers to 512B blocks of file data)
  - Then, single, double, and triple indirect pointers

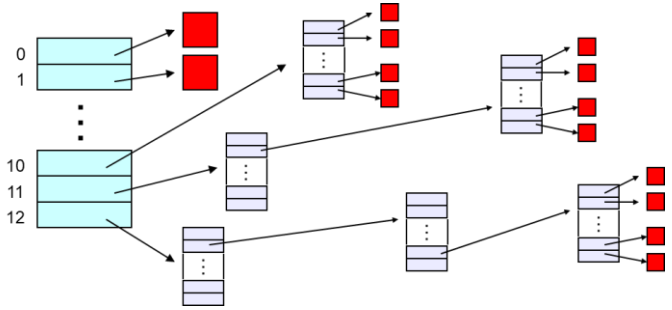


Example: Max File Size #1



- Assume **4B block pointers** and **512B block size**
- Block list occupies 13 x 4B in the inode
- Can get to **10 x 512B = 5120B** file **directly**
  - 10 direct pointers, blocks in the file contents area
- Can get to **(512/4) x 512B = 64kB** with a **single indirect** reference
  - The 11th pointer in the inode
  - Points to a block (512B) in the file contents area
  - This contains (512/4) x 4B pointers to blocks holding file data
- Can get to **(512/4) x (512/4) x 512B = 8MB** with a **double indirect** reference
  - The 12th pointer in the inode
  - Points to a block (512B) in the file contents area
    - that contains (512/4) x 4B pointers to
    - blocks (512B) in the file contents area
      - that contain (512/4) x 4B pointers to
      - blocks (512B) holding file data

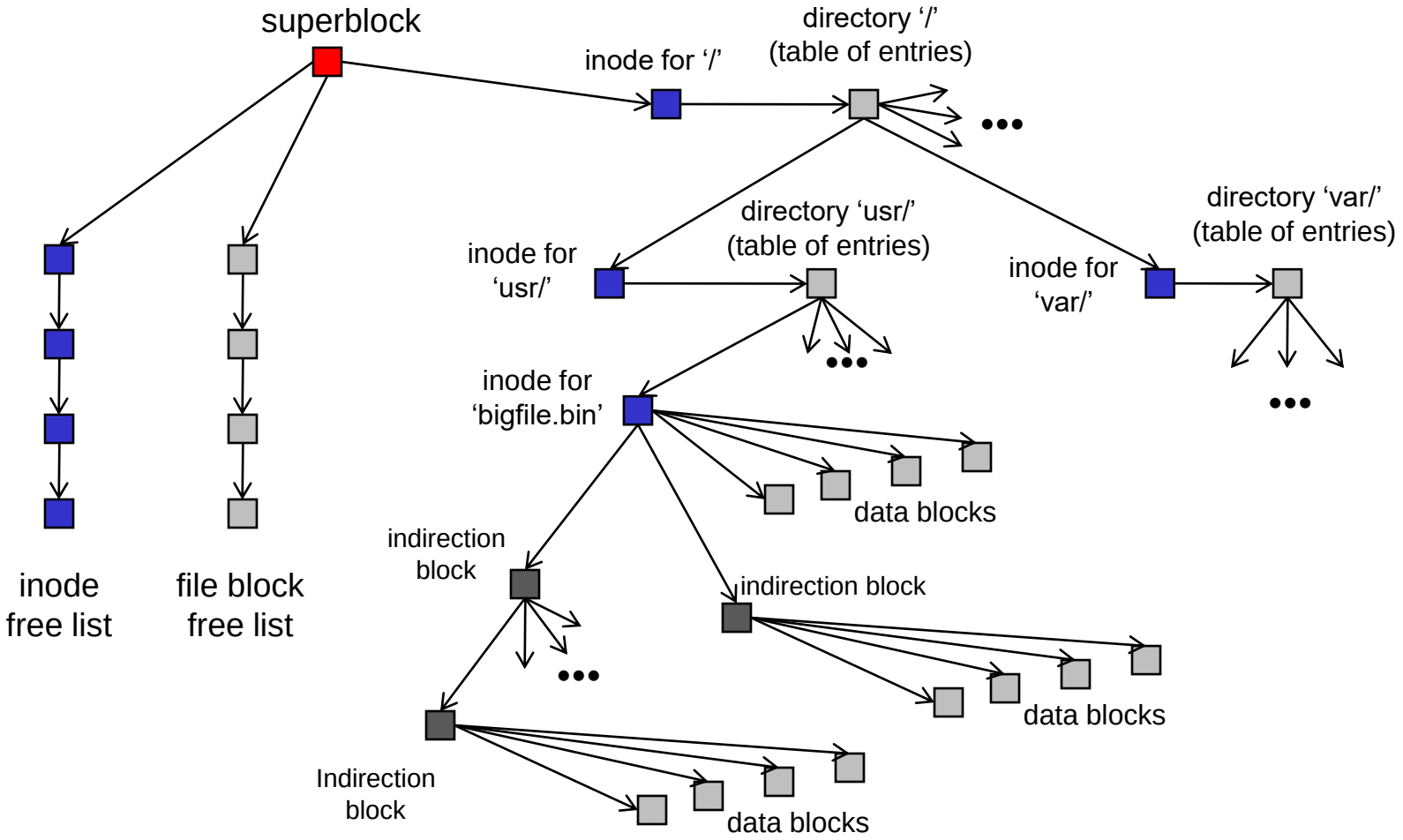
Example: Max File Size #2



- Can get to **(512/4) x (512/4) x (512/4) x 512B = 1GB** with a **triple indirect** reference
  - The 13th pointer in the inode
  - Points to a block (512B) in the file contents area
    - that contains (512/4) x 4B pointers to
    - blocks (512B) in the file contents area
      - that contain (512/4) x 4B pointers to
      - blocks (512B) in the file contents area
        - » that contain (512/4) x 4B pointers to
        - » blocks (512B) holding file data
- **Maximum file size** is
  - 5120B + 64kB + 8MB + 1GB = ~ 1GB

All Together

- The file system is just a huge data structure



File System Layout

- One important goal of a file system is to **lay this data structure on disk**
  - Keep in mind the physical characteristics of the disk
  - Seeks are expensive
  - Characteristics of the workload
    - Locality across files within a directory
    - Sequential access to many files
- Old layouts were inefficient
  - constantly seeking
- Newer file systems are more efficient
- Newer storage devices (SSDs) **changed constraints**