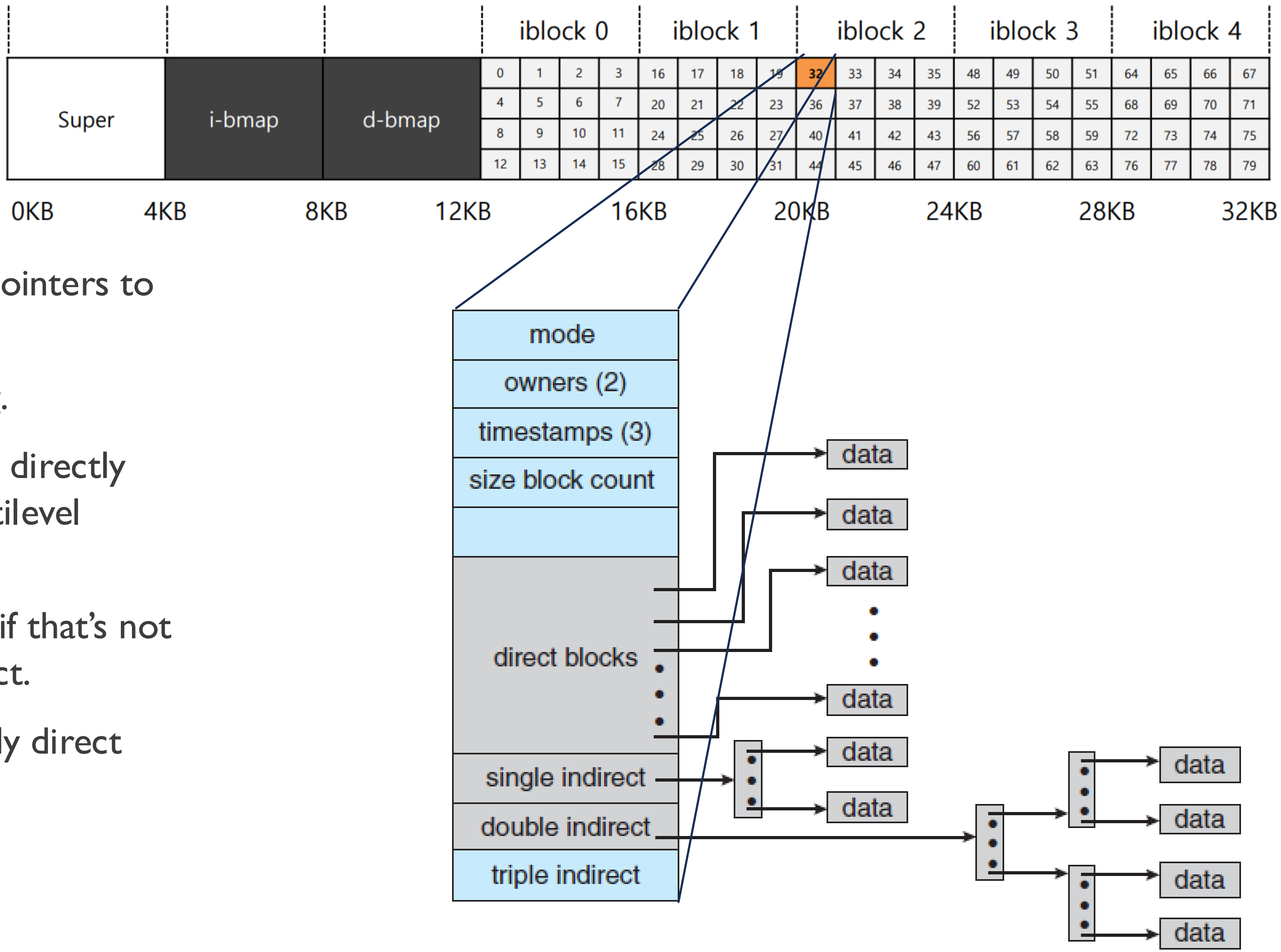
CSCI 509

OPERATING SYSTEMS

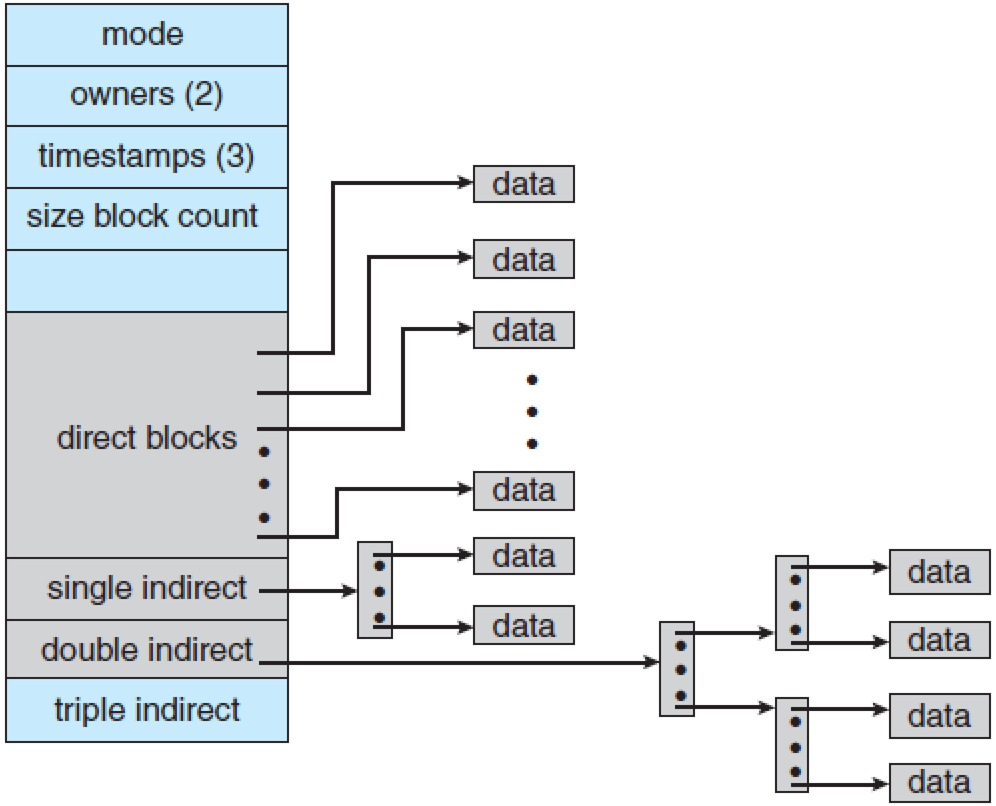
LINUX INODE

- Contain both metadata and pointers to blocks used.
- Uses various type of indexing.
- First blocks can be addressed directly others could have utilize multilevel indexing.
- Start with using direct block, if that’s not enough for the file use indirect.
- Most files are small and usually direct blocks would suffice.



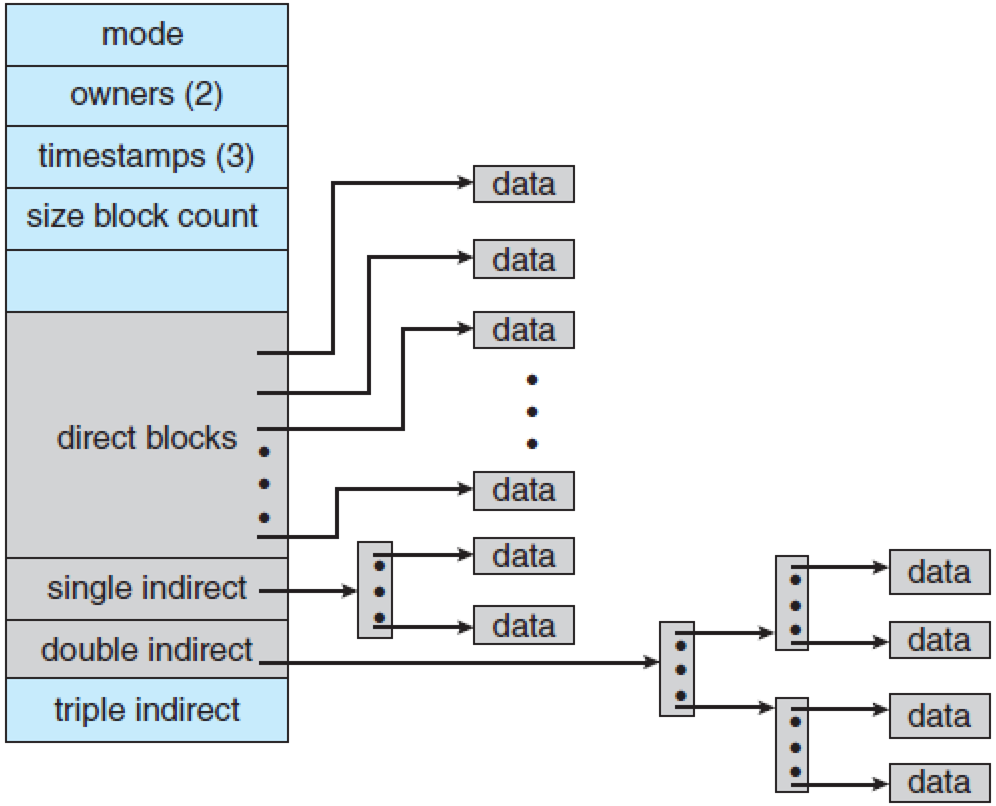
WORKSHEET

- In an ext2 file system an inode consists of only 15 block pointers.
- The first 12 block pointers are direct block pointers.
- The 13th pointer is an indirect pointer.
- The 14th pointer is a double indirect pointer.
- The 15th pointer is a triple indirect pointer.
- Block size of 4KB
- 32-bit addressing for the blocks
- Which of these pointers will be utilized when the inode represents a file of size 64 KB?
- Which of these pointers will remain unutilized?



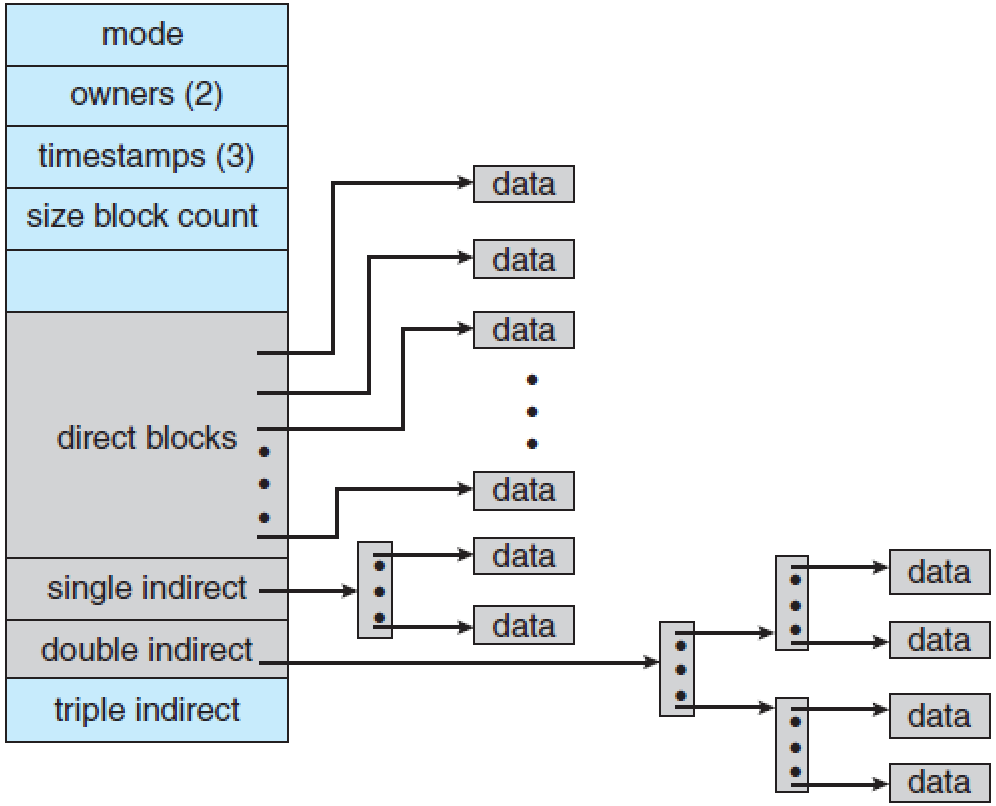
WORKSHEET

- How many blocks do we need for the file?
- How much "size on disk" does each direct block pointer support?
- How much "size on disk" does a single indirect pointer can support?
    - How many direct pointers can a block on disk hold?



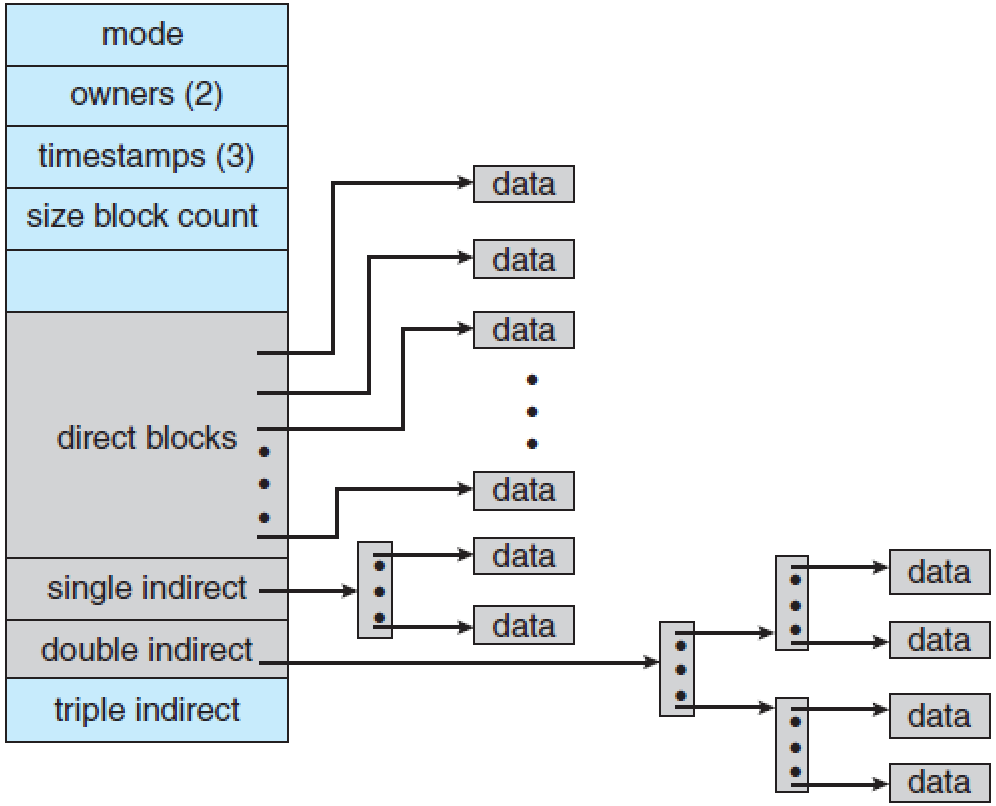
INODE EXERCISE

- How many blocks do we need for the file?
- 64KB and 4KB per block ➔ 16 blocks.



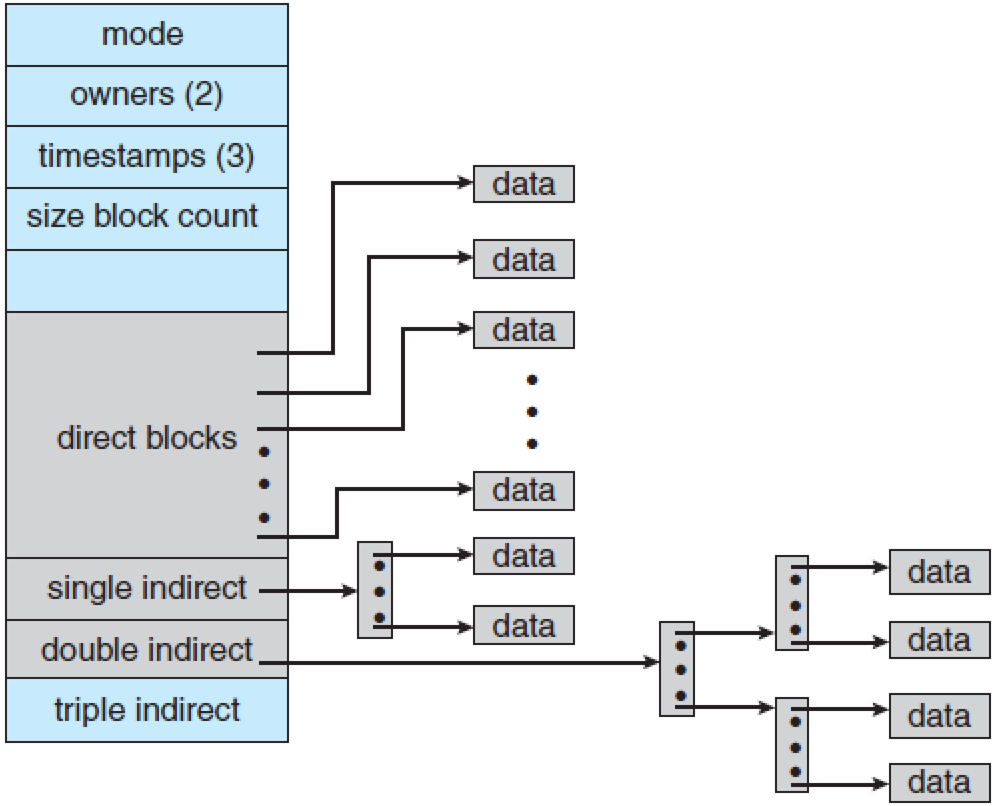
INODE EXERCISE

- How many blocks do we need for the file?
- 64KB and 4KB per block ➔ 16 blocks.
- We only have 12 direct pointers, we need at least single indirect for the extra 4.



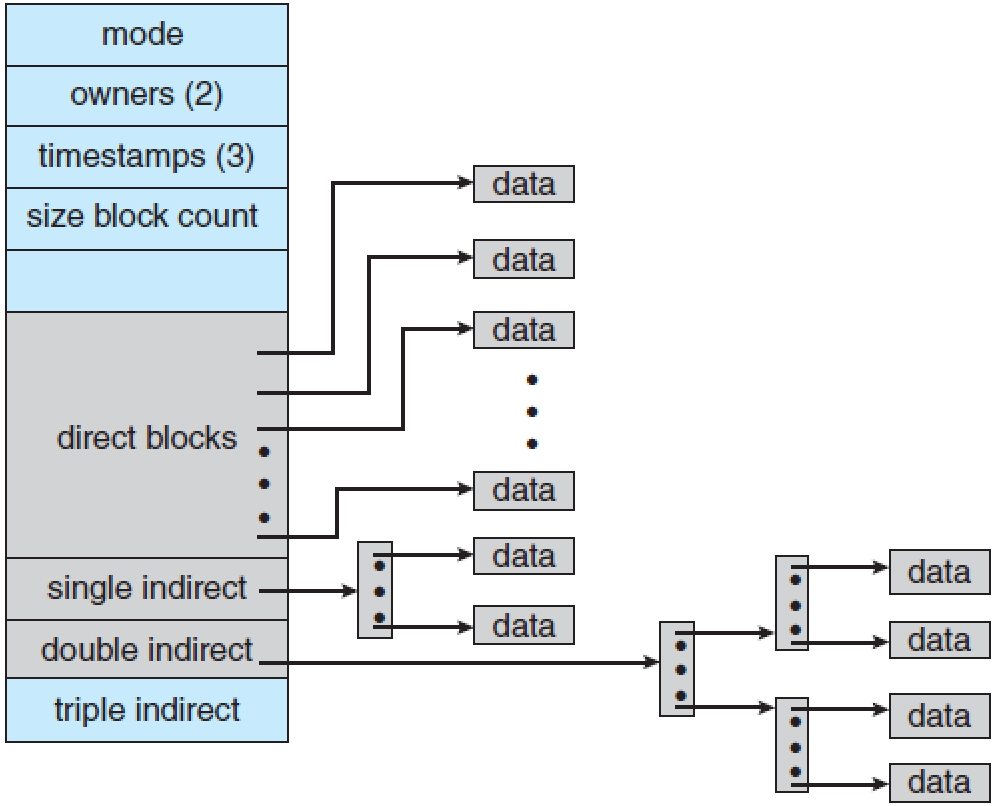
INODE EXERCISE

- How many blocks do we need for the file?
- 64KB and 4KB per block ➔ 16 blocks.
- We only have 12 direct pointers, we need at least single indirect for the extra 4.
- How many does single indirect supports?



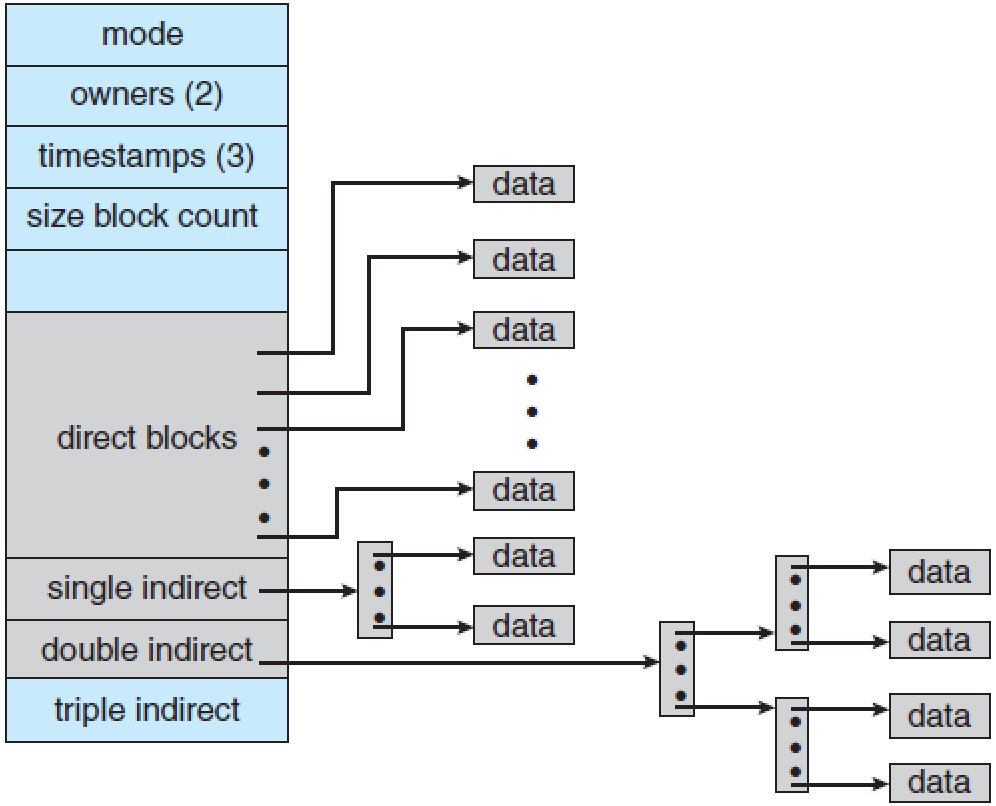
INODE EXERCISE

- How many blocks do we need for the file?
- 64KB and 4KB per block ➔ 16 blocks.
- We only have 12 direct pointers, we need at least single indirect for the extra 4.
- How many does single indirect supports?
- 32-bit machine ➔ each direct pointer needs 4 bytes to be stored.



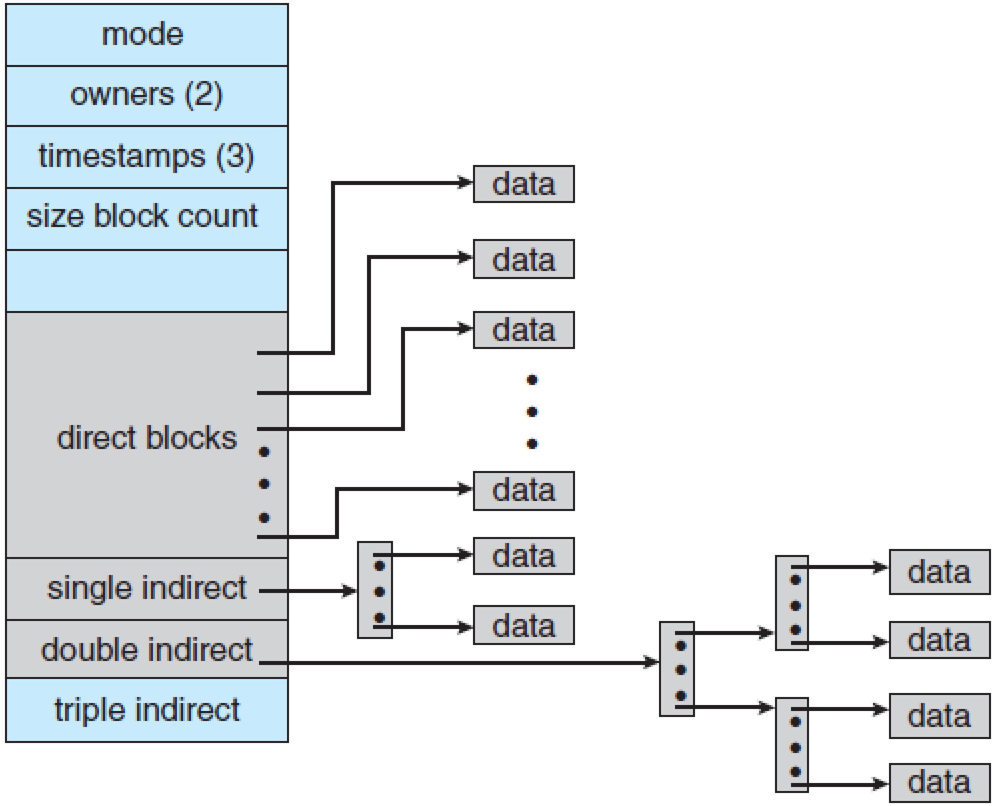
INODE EXERCISE

- How many blocks do we need for the file?
- 64KB and 4KB per block ➔ 16 blocks.
- We only have 12 direct pointers, we need at least single indirect for the extra 4.
- How many does single indirect supports?
- 32-bit machine ➔ each direct pointer needs 4 bytes to be stored.
- With a block size of 4KB, one block can store 1024 block addresses or 1024 "direct pointer" …



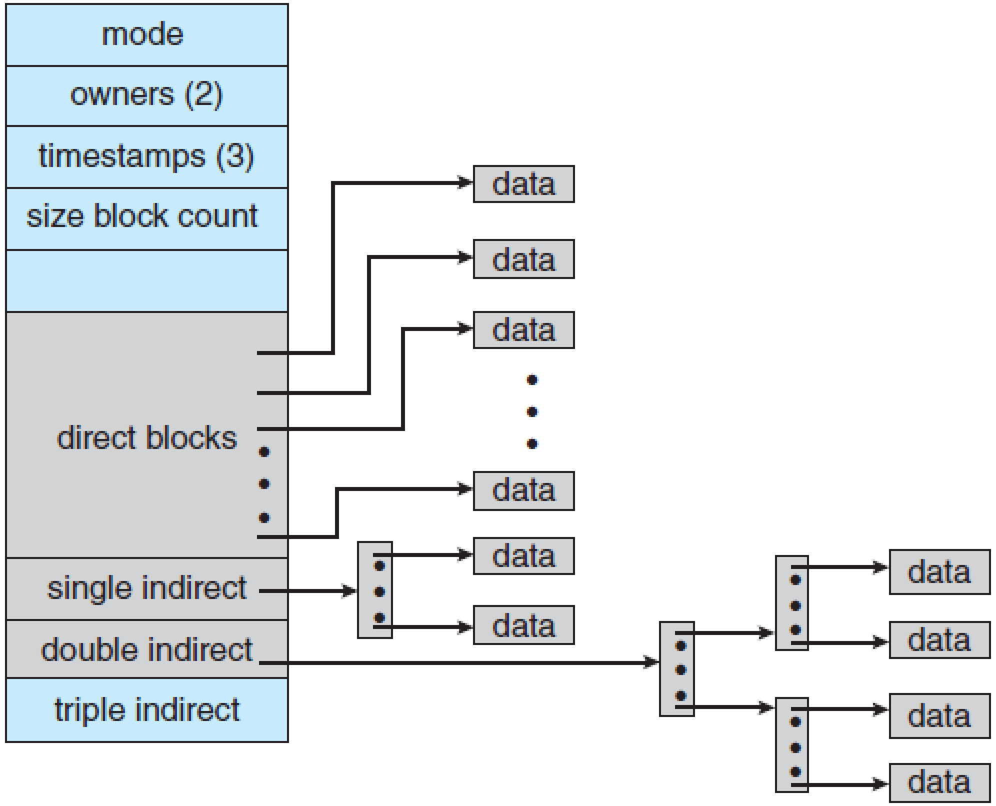
INODE EXERCISE

- How many blocks do we need for the file?
- 64KB and 4KB per block ➔ 16 blocks.
- We only have 12 direct pointers, we need at least single indirect for the extra 4.
- How many does single indirect supports?
- 32-bit machine ➔ each direct pointer needs 4 bytes to be stored.
- With a block size of 4KB, one block can store 1024 block addresses or 1024 "direct pointer" …
- We only need 4 more, single indirect provides 1024 more so single indirect should suffice.

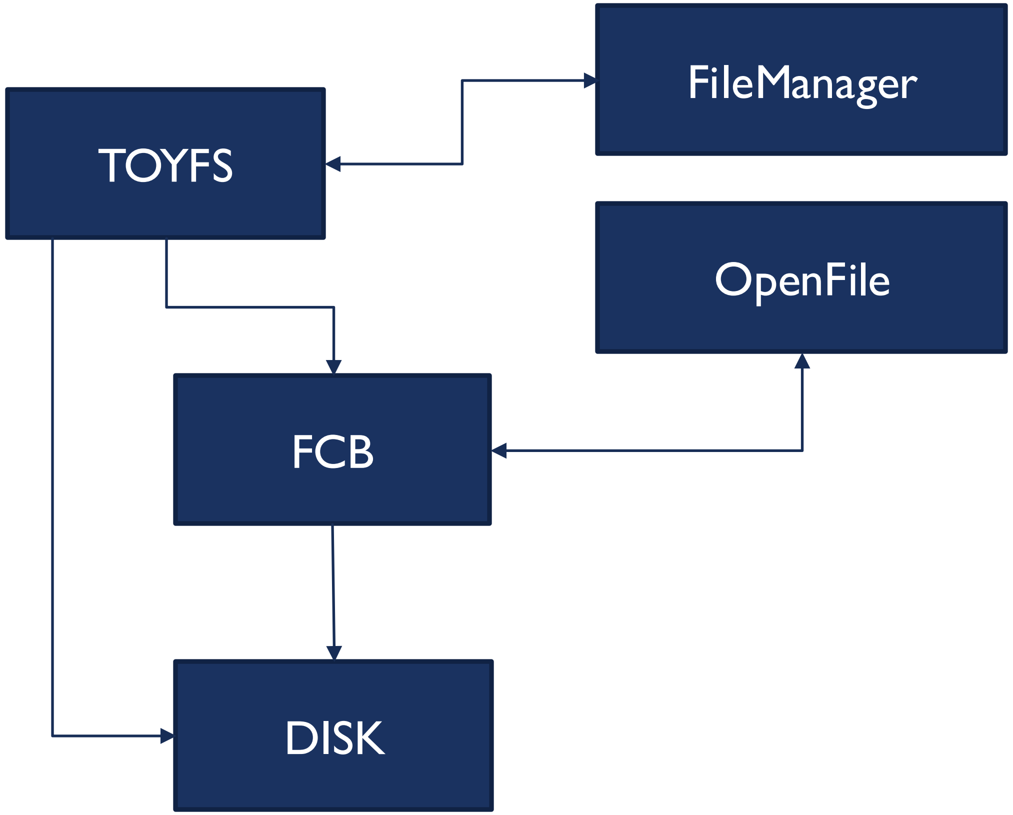


INODE EXERCISE

- How many blocks do we need for the file?
- 64KB and 4KB per block ➔ 16 blocks.
- We only have 12 direct pointers, we need at least single indirect for the extra 4.
- How many does single indirect supports?
- 32-bit machine ➔ each direct pointer needs 4 bytes to be stored.
- With a block size of 4KB, one block can store 1024 block addresses or 1024 "direct pointer" …
- We only need 4 more, single indirect provides 1024 more so single indirect should suffice.
- Double and triple indirect are never used.



TOYFS



There are other classes as well

FILEMANAGER

- Manages OpenFile and Pipe Objects
- Methods include
  - GetAnOpenFile
  - Open, Close
  - GetAPipe
  - …
- Includes data structures
  - OpenFileTable
  - openFileFreeList
  - …

FCB

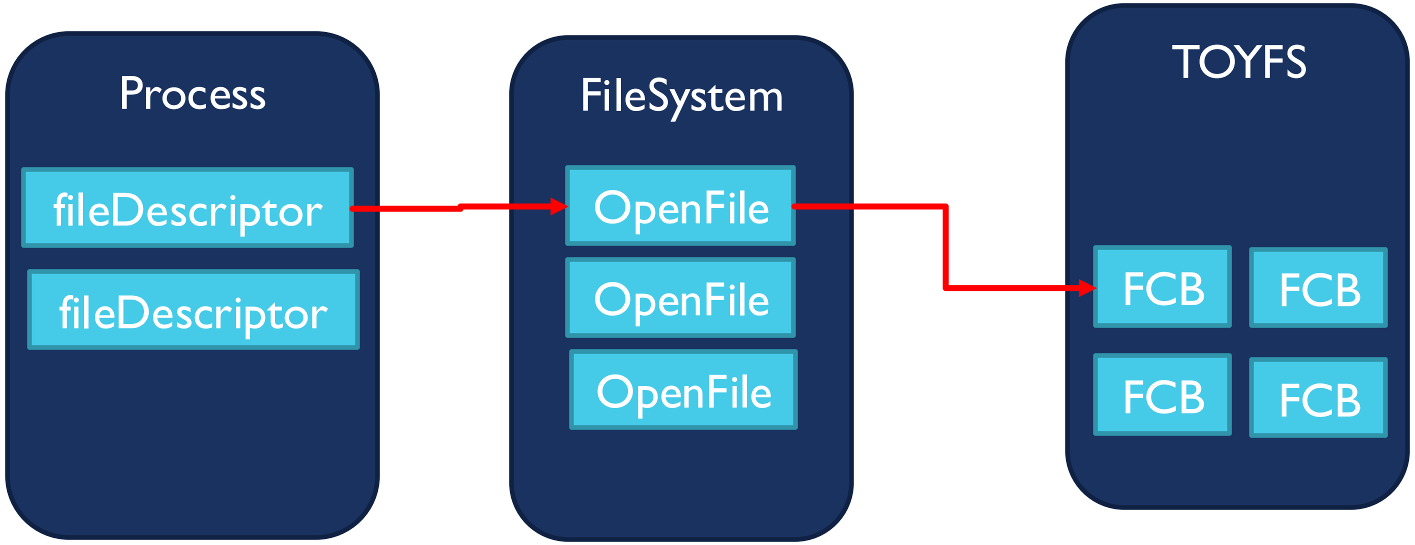
- Records all data associated with a single ToyFS file
- Has an InodeData object that can be used to get file info, like the actual sectors on disk.
- Has methods to read and write to disk

TOYFS

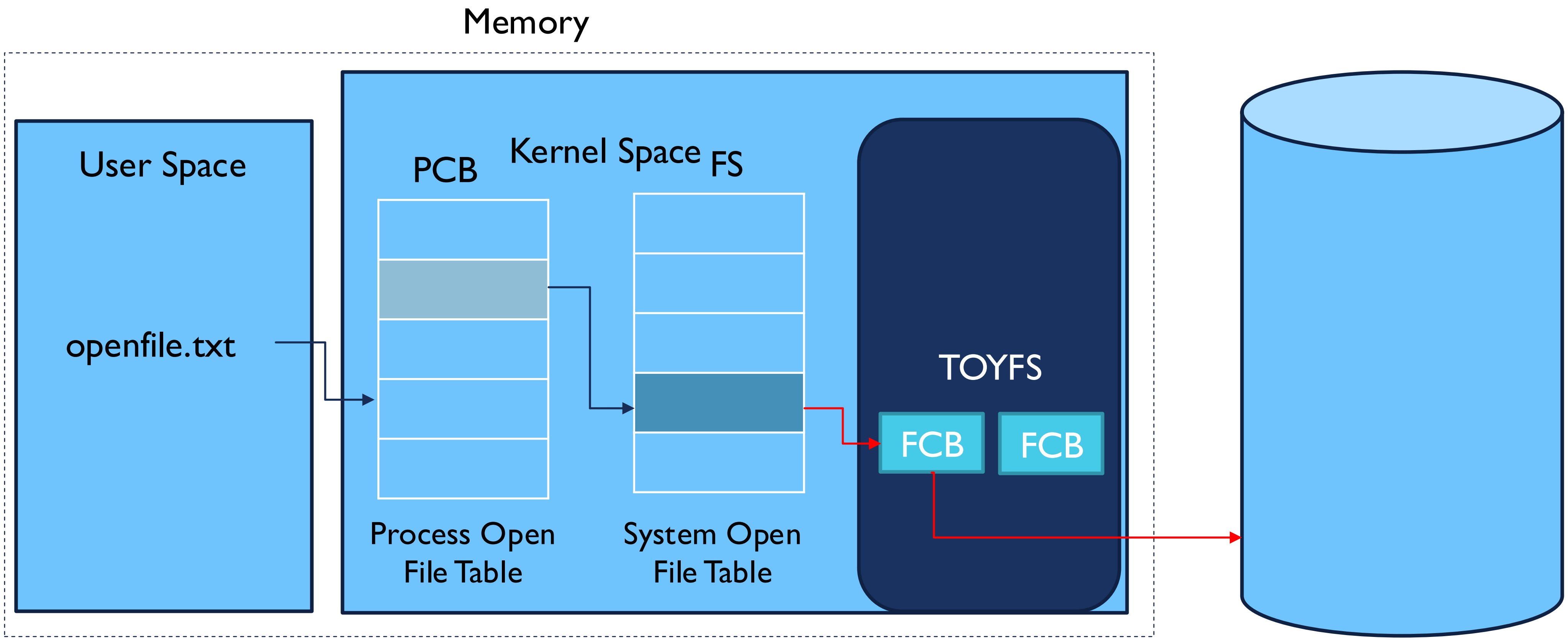
- Contains the superblock (in memory)
  - This include data and inode bitmaps
- Has a handle to root directory (root always in inode 1 on disk)
- FCB Table, + List of FREE FCB
- Can allocate/free inodes or data blocks
- Methods to look up inode give a file name or an FCB

FileSystem has limited OpenFile objects. It maintain a free list of OpenFile objects

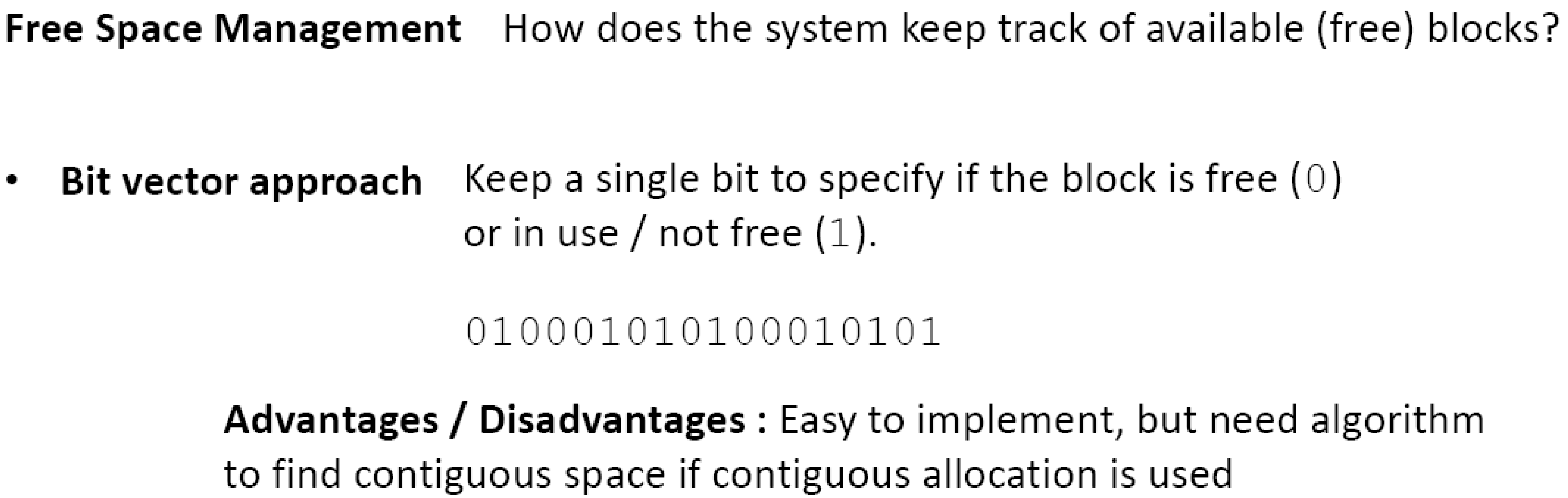
TOYFS has limited FCBs. It maintain a free list of FCBs



IN MEMORY FILE SYSTEM IMPLEMENTATION



TRACKING FREE DISK SPACE



Free Space Management How does the system keep track of available (free) blocks?

- **Bit vector approach** Keep a single bit to specify if the block is free (`0`) or in use / not free (`1`).

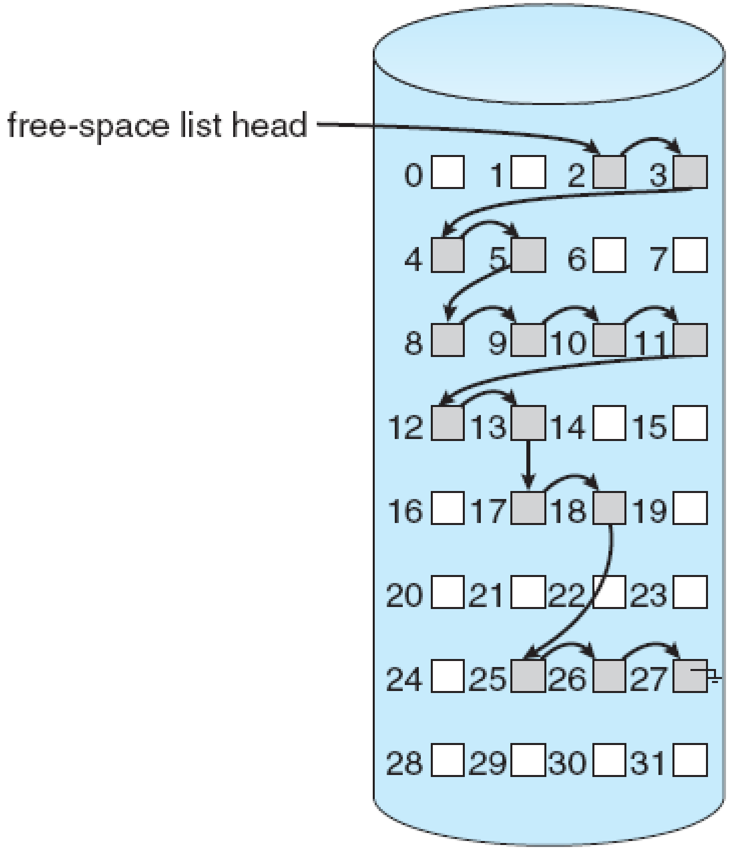
```
010001010100010101
```

Advantages / Disadvantages : Easy to implement, but need algorithm to find contiguous space if contiguous allocation is used

You need to search for the '0's …

TRACKING FREE DISK SPACE

- Linked list of available block.
- Disadvantage: Complex to implement
- Advantage: Can easily allocate contiguous space.



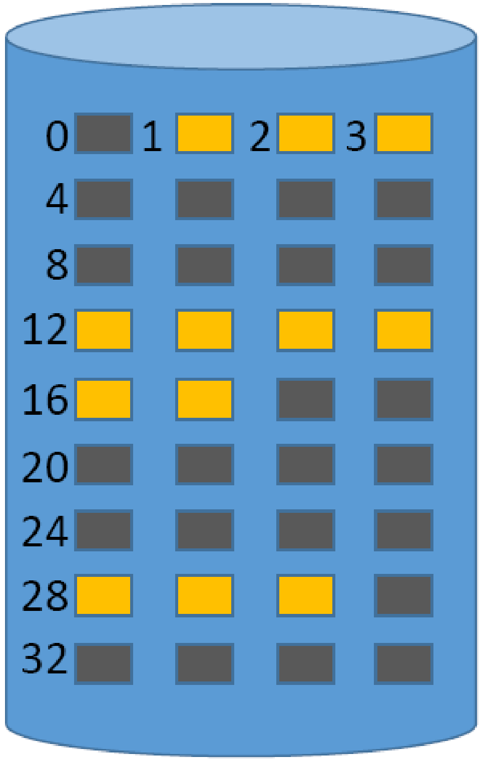
FREE SPACE MANAGEMENT

- Counting
  - Because space is frequently contiguously used and freed, with contiguous-allocation allocation, extents, or clustering
  - Keep address of first free block and count of following free blocks
  - Free space list then has entries containing addresses and counts

directory

File	Start	Length
myFile	1	3
aPic	12	6
song	28	3

Very similar to contiguous allocation … we're just keeping track of free blocks instead of file blocks.

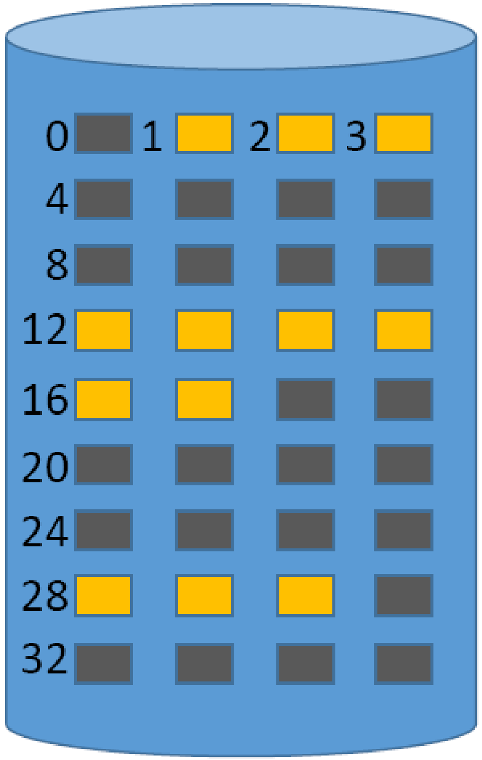


FREE SPACE MANAGEMENT

- Counting
  - Because space is frequently contiguously used and freed, with contiguous-allocation allocation, extents, or clustering
  - Keep address of first free block and count of following free blocks
  - Free space list then has entries containing addresses and counts

Free Blocks List

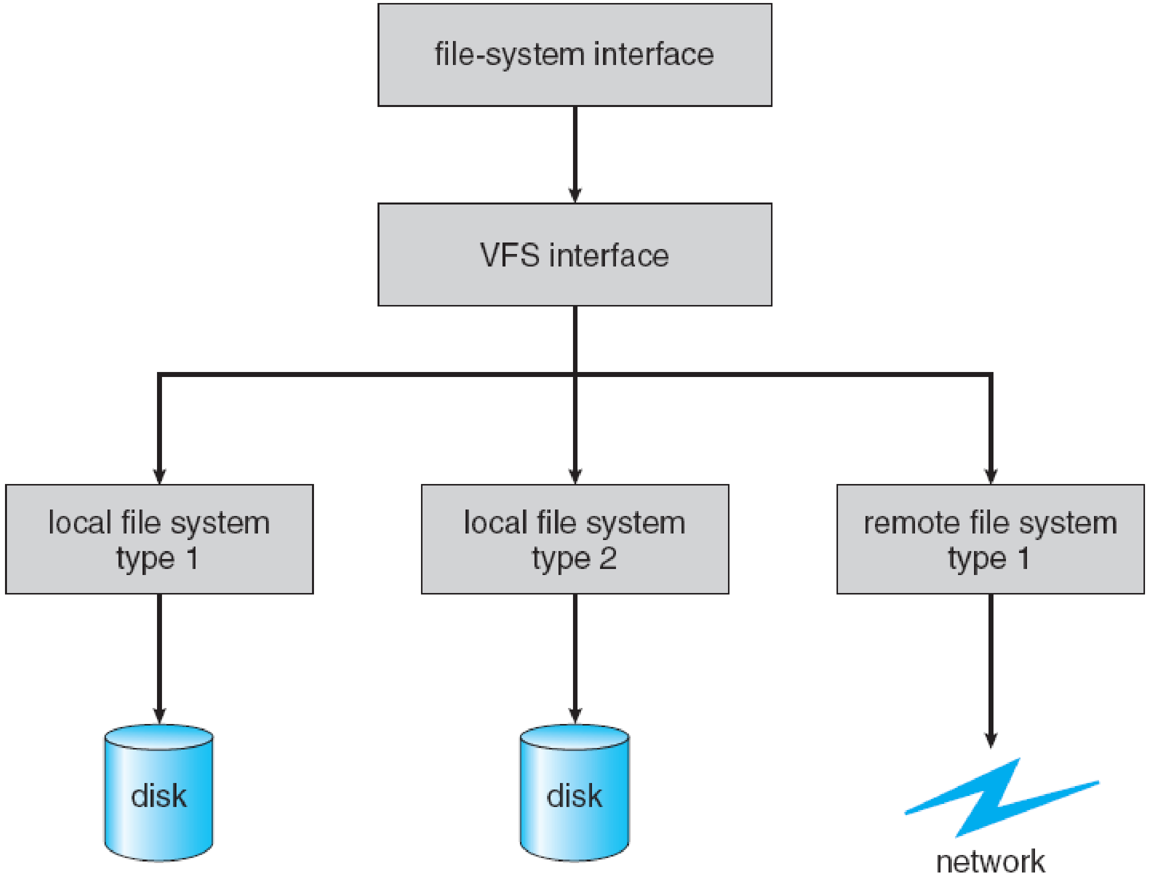
Block	Start	Length
	1	3
	12	6
	28	3



Very similar to contiguous allocation … we're just keeping track of free blocks instead of file blocks.

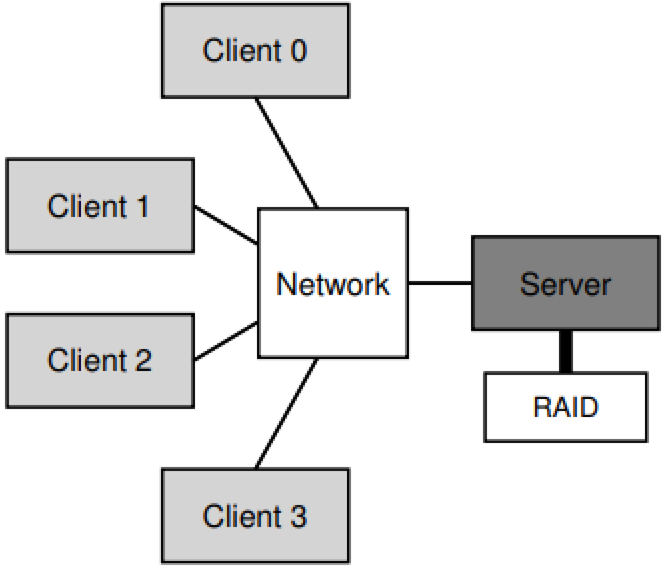
VIRTUAL FILE SYSTEM

- Operating systems utilize a virtual file system interface.
- This allows for programs to use the same system calls (such as open(), read(), write(), close(), mkdir(), etc.) regardless of what file system they are accessing.



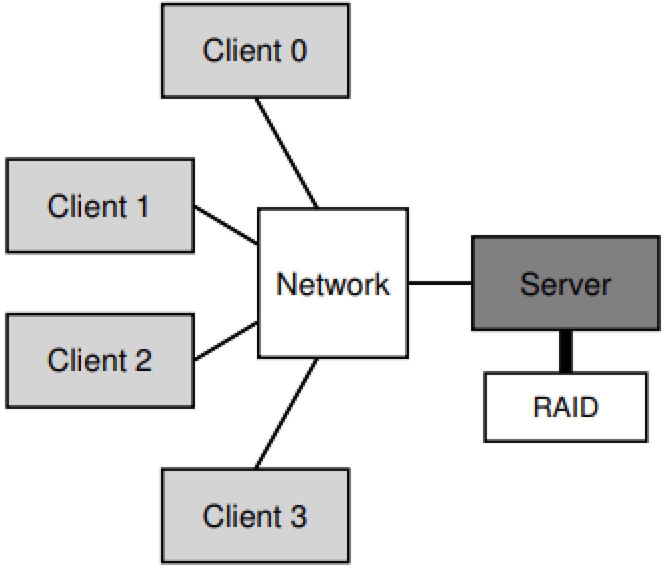
NETWORK FILE SYSTEM

- Advantages?



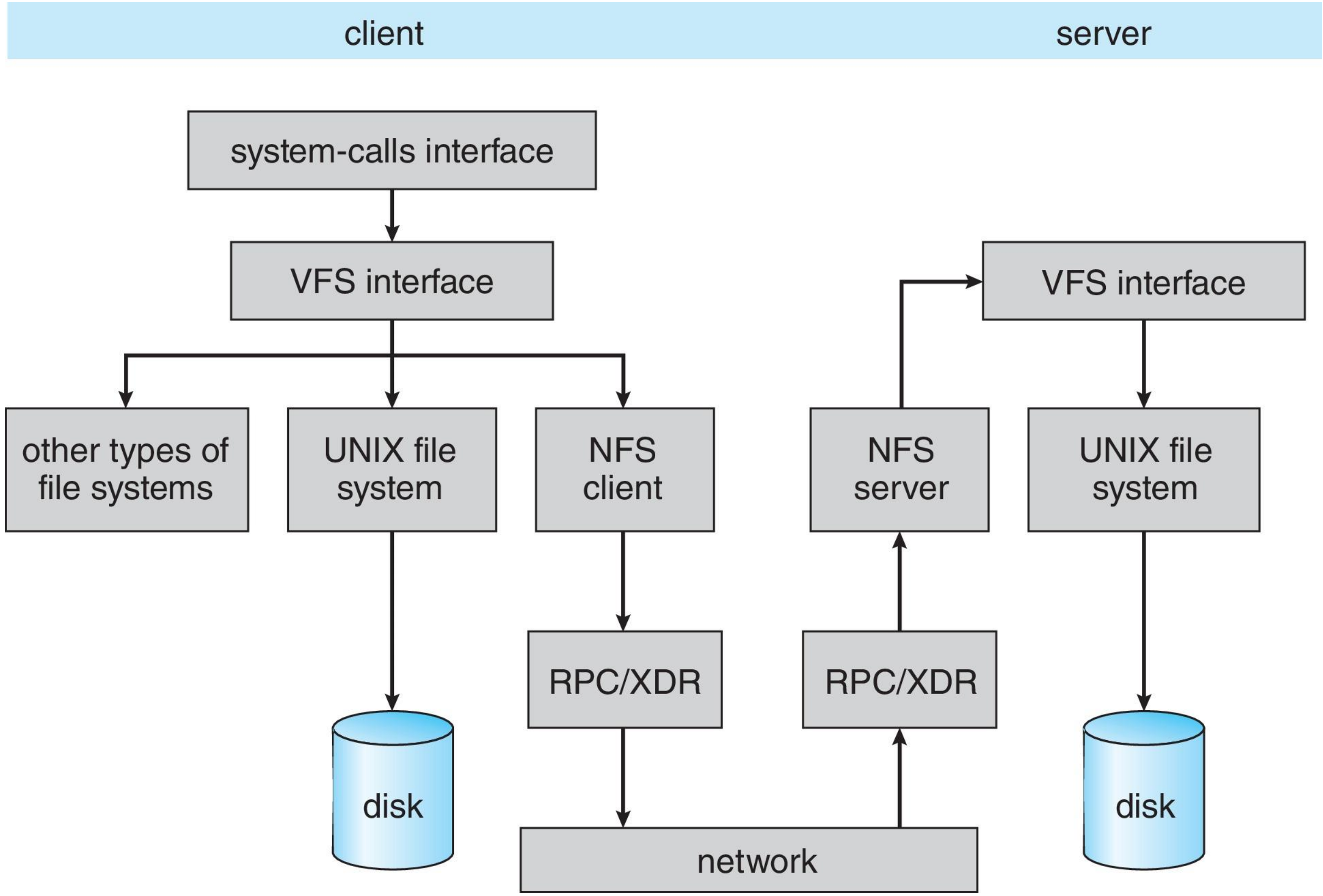
NETWORK FILE SYSTEM

- Sharing
  - Sharing data
  - Sharing free space
- Centralized Administration
  - Backing Up
  - Restoration
- Security



NETWORK FILE SYSTEM

- Once setup, the network file system is accessed like any local file system.
- Users/Programs utilize the virtual file system where they access remote or local file.

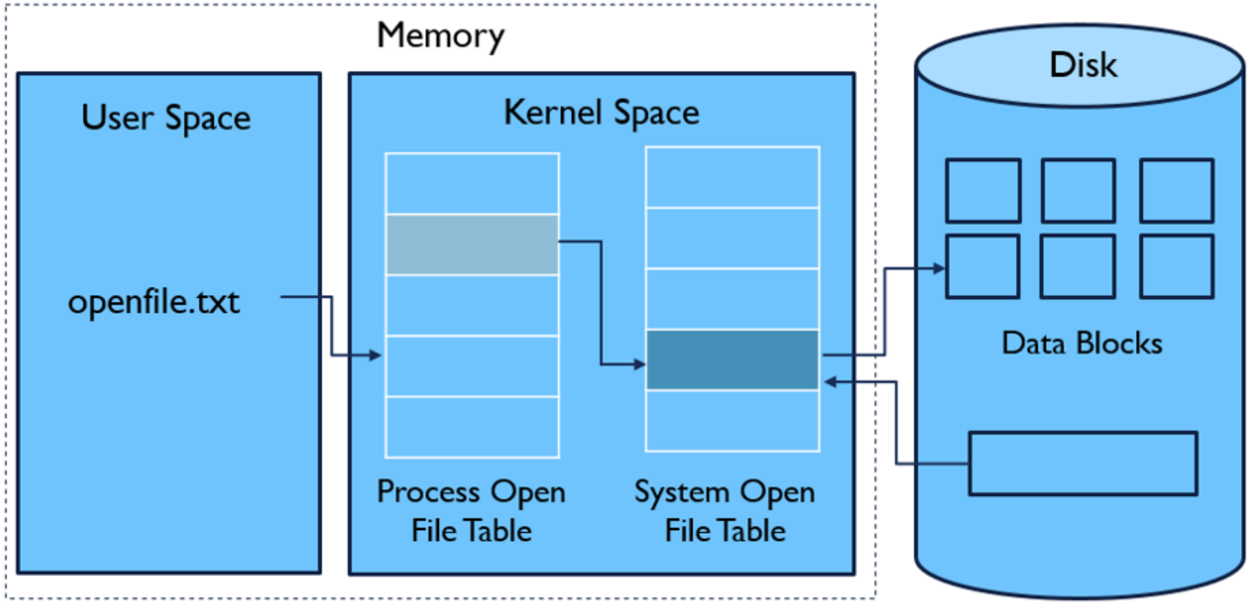


NETWORK FILE SYSTEM

- NFS was developed by Sun microsystem.
- Latest version: NFSv4, we will look at NFSv2 which was what made NFS popular.
- Design was centered around *Fast Crash Recovery*.
- To achieve this, the NFS used "stateless" design.

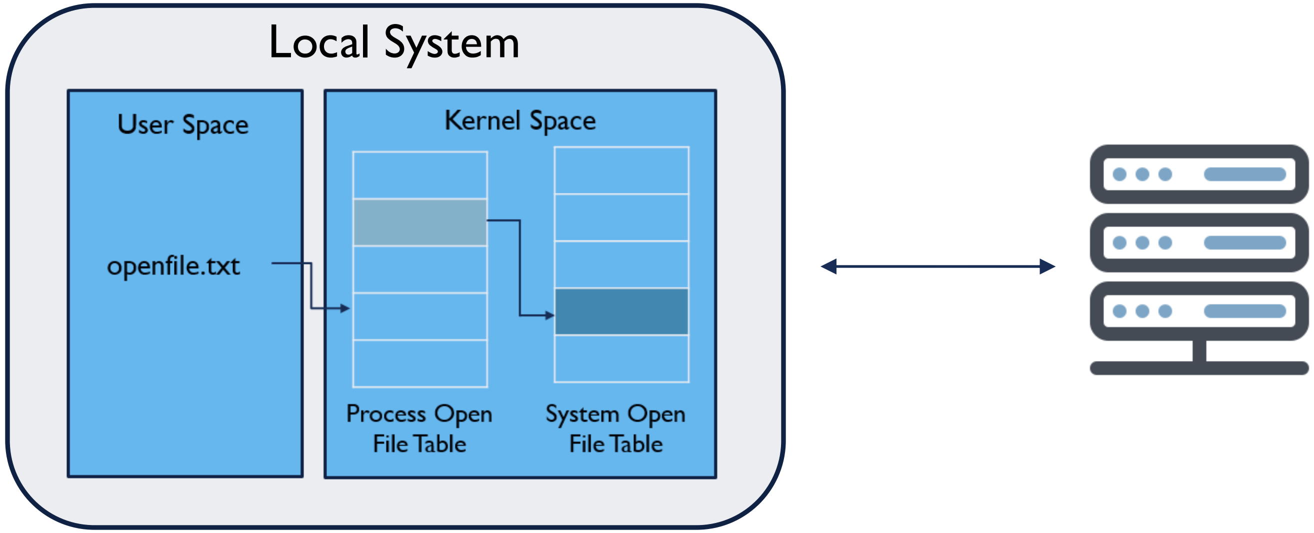
STATELESS FILE SYSTEM

- "Stateful" file systems, like local file systems keep track of almost all ongoing operations.
- Open file table, file pointers, users accessing the file.
- If a server crash occurs, all this information is lost (which files are open …)
- If a client crash occurs, it also creates problems.
- Need to implement recovery algorithm and perform recovery with each connecting client.



HOW TO IMPLEMENT A STATELESS FILE SYSTEM?

- From the client perspective, the NFS is a stateful protocol: Virtual File System.
- The system however, only issues stand-alone commands to the NFS server.



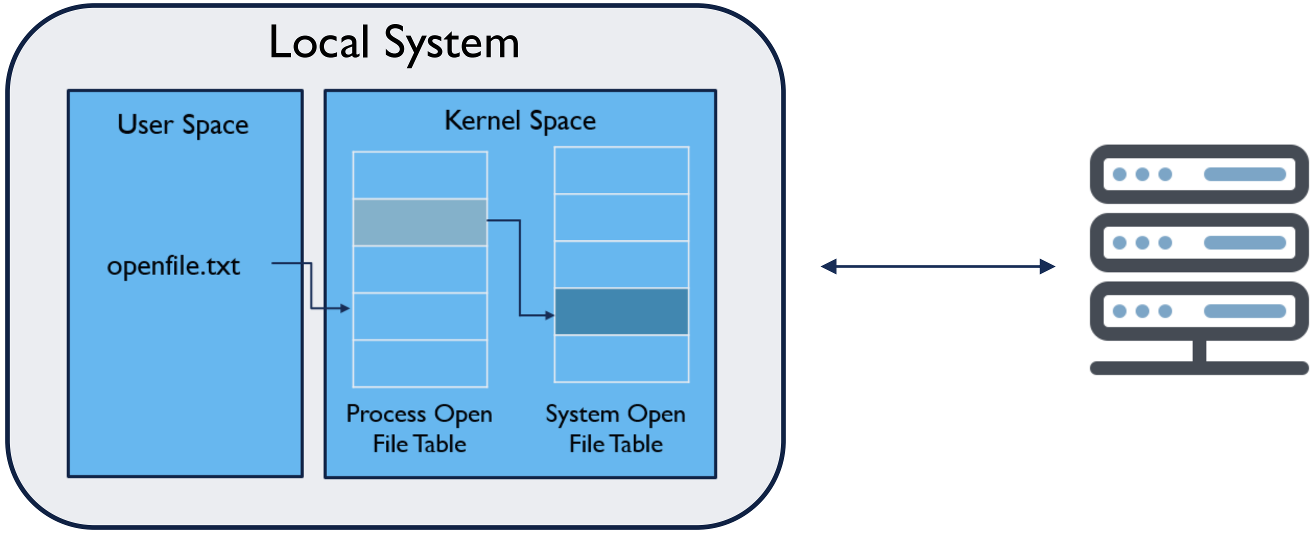
Q: What access method should the NFS 2.0 utilize?

A B

A: Direct
B: Sequential

HOW TO IMPLEMENT A STATELESS FILE SYSTEM?

- From the client perspective, the NFS is a stateful protocol: Virtual File System.
- The system however, only issues stand-alone commands to the NFS server.
- Example commands:



Q: What access method should the NFS 2.0 utilize?

A B

A: Direct
B: Sequential

The server can't keep a record of any state, including file pointers for "read next".

NFS 4

- The client and server establish a long-lived session
- This session maintains the client's state information, such as file locks and open file handles
- Allows the client to recover its state after a network interruption or server failure without having to re-establish its entire connection.
- With the advancement of computing power, memory capacity and network bandwidth, the overhead of reestablishing a connection became minimal compared to the benefits.

FILE SYSTEM RECOVERY

- How can the file system recover from crashes?

FILE SYSTEM RECOVERY

- How can the file system recover from crashes?
- Example: Power failure could occur while writing to disk …
  - Directory structure could become inaccurate
  - Files could be partially written
  - File meta data could be out of date
  - …

FILE SYSTEM RECOVERY

- **Consistency checking** – compares data in directory structure with data blocks on disk, and tries to fix inconsistencies
  - Can be slow and sometimes fails

FILE SYSTEM RECOVERY

- **Consistency checking** – compares data in directory structure with data blocks on disk, and tries to fix inconsistencies
    - Can be slow and sometimes fails
- Use system programs to **back up** data from disk to another storage device (magnetic tape, other magnetic disk, optical)

LOG STRUCTURED FILE SYSTEMS

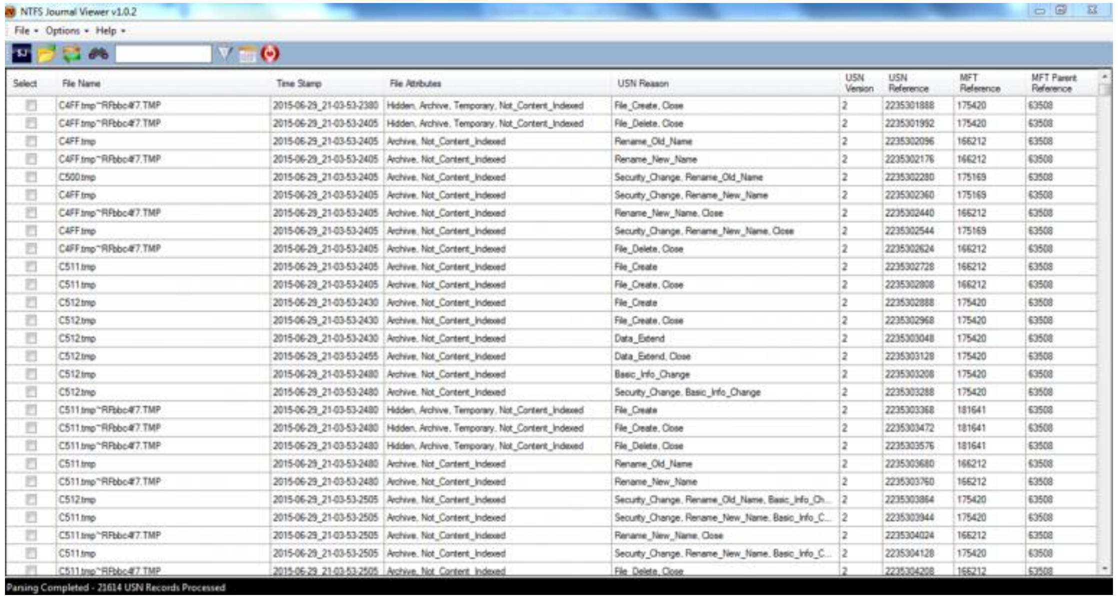
- **Log structured** (or **journaled**) file systems record each metadata update to the file system as a **transaction**

LOG STRUCTURED FILE SYSTEMS

- **Log structured** (or **journaled**) file systems record each metadata update to the file system as a **transaction**
- All transactions are written to a log
  - A transaction is considered committed once it is written to the log (sequentially)

LOG STRUCTURED FILE SYSTEMS

- **Log structured** (or **journaled**) file systems record each metadata update to the file system as a **transaction**
- All transactions are written to a log
  - A transaction is considered committed once it is written to the log (sequentially)
- Allows faster recovery from crash, removes chance of inconsistency of metadata



Select	File Name	Time Stamp	File Attributes	USN Reason	USN Version	USN Reference	MFT Reference	MFT Parent Reference
□	C4FF.tmp~RFbbc4f7.TMP	2015-06-29_21-03-53-2380	Hidden, Archive, Temporary, Not_Content_Indexed	File_Create, Close	2	2235301888	175420	63508
□	C4FF.tmp~RFbbc4f7.TMP	2015-06-29_21-03-53-2405	Hidden, Archive, Temporary, Not_Content_Indexed	File_Delete, Close	2	2235301992	175420	63508
□	C4FF.tmp	2015-06-29_21-03-53-2405	Archive, Not_Content_Indexed	Rename_Old_Name	2	2235302096	166212	63508
□	C4FF.tmp~RFbbc4f7.TMP	2015-06-29_21-03-53-2405	Archive, Not_Content_Indexed	Rename_New_Name	2	2235302176	166212	63508
□	C500.tmp	2015-06-29_21-03-53-2405	Archive, Not_Content_Indexed	Security_Change, Rename_Old_Name	2	2235302280	175169	63508
□	C4FF.tmp	2015-06-29_21-03-53-2405	Archive, Not_Content_Indexed	Security_Change, Rename_New_Name	2	2235302360	175169	63508
□	C4FF.tmp~RFbbc4f7.TMP	2015-06-29_21-03-53-2405	Archive, Not_Content_Indexed	Rename_New_Name, Close	2	2235302440	166212	63508
□	C4FF.tmp	2015-06-29_21-03-53-2405	Archive, Not_Content_Indexed	Security_Change, Rename_New_Name, Close	2	2235302544	175169	63508
□	C4FF.tmp~RFbbc4f7.TMP	2015-06-29_21-03-53-2405	Archive, Not_Content_Indexed	File_Delete, Close	2	2235302624	166212	63508
□	C511.tmp	2015-06-29_21-03-53-2405	Archive, Not_Content_Indexed	File_Create	2	2235302728	166212	63508
□	C511.tmp	2015-06-29_21-03-53-2405	Archive, Not_Content_Indexed	File_Create, Close	2	2235302808	166212	63508
□	C512.tmp	2015-06-29_21-03-53-2430	Archive, Not_Content_Indexed	File_Create	2	2235302888	175420	63508
□	C512.tmp	2015-06-29_21-03-53-2430	Archive, Not_Content_Indexed	File_Create, Close	2	2235302968	175420	63508
□	C512.tmp	2015-06-29_21-03-53-2430	Archive, Not_Content_Indexed	Data_Extend	2	2235303048	175420	63508
□	C512.tmp	2015-06-29_21-03-53-2455	Archive, Not_Content_Indexed	Data_Extend, Close	2	2235303128	175420	63508
□	C512.tmp	2015-06-29_21-03-53-2480	Archive, Not_Content_Indexed	Basic_Info_Change	2	2235303208	175420	63508
□	C512.tmp	2015-06-29_21-03-53-2480	Archive, Not_Content_Indexed	Security_Change, Basic_Info_Change	2	2235303288	175420	63508
□	C511.tmp~RFbbc4f7.TMP	2015-06-29_21-03-53-2480	Hidden, Archive, Temporary, Not_Content_Indexed	File_Create	2	2235303368	181641	63508
□	C511.tmp~RFbbc4f7.TMP	2015-06-29_21-03-53-2480	Hidden, Archive, Temporary, Not_Content_Indexed	File_Create, Close	2	2235303472	181641	63508
□	C511.tmp~RFbbc4f7.TMP	2015-06-29_21-03-53-2480	Hidden, Archive, Temporary, Not_Content_Indexed	File_Delete, Close	2	2235303576	181641	63508
□	C511.tmp	2015-06-29_21-03-53-2480	Archive, Not_Content_Indexed	Rename_Old_Name	2	2235303680	166212	63508
□	C511.tmp~RFbbc4f7.TMP	2015-06-29_21-03-53-2480	Archive, Not_Content_Indexed	Rename_New_Name	2	2235303760	166212	63508
□	C512.tmp	2015-06-29_21-03-53-2505	Archive, Not_Content_Indexed	Security_Change, Rename_Old_Name, Basic_Info_Ch...	2	2235303864	175420	63508
□	C511.tmp	2015-06-29_21-03-53-2505	Archive, Not_Content_Indexed	Security_Change, Rename_New_Name, Basic_Info_C...	2	2235303944	175420	63508
□	C511.tmp~RFbbc4f7.TMP	2015-06-29_21-03-53-2505	Archive, Not_Content_Indexed	Rename_New_Name, Close	2	2235304024	166212	63508
□	C511.tmp	2015-06-29_21-03-53-2505	Archive, Not_Content_Indexed	Security_Change, Rename_New_Name, Basic_Info_C...	2	2235304128	175420	63508
□	C511.tmp~RFbbc4f7.TMP	2015-06-29_21-03-53-2505	Archive, Not_Content_Indexed	File_Delete, Close	2	2235304208	166212	63508

Parsing Completed - 21614 USN Records Processed

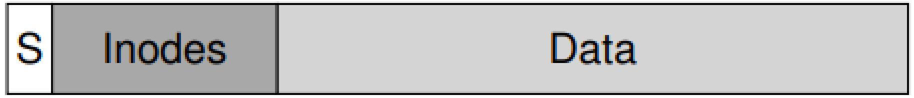
NTFS Logfile

UNIX FILE SYSTEM

- Initially used “FS” or “File System”, first UNIX file system.
- Has gone through significant changes and iterations
- BFS (Berkley File System)
- ext or Extended File System
- Latest is ext4

OLD UNIX FILE SYSTEM

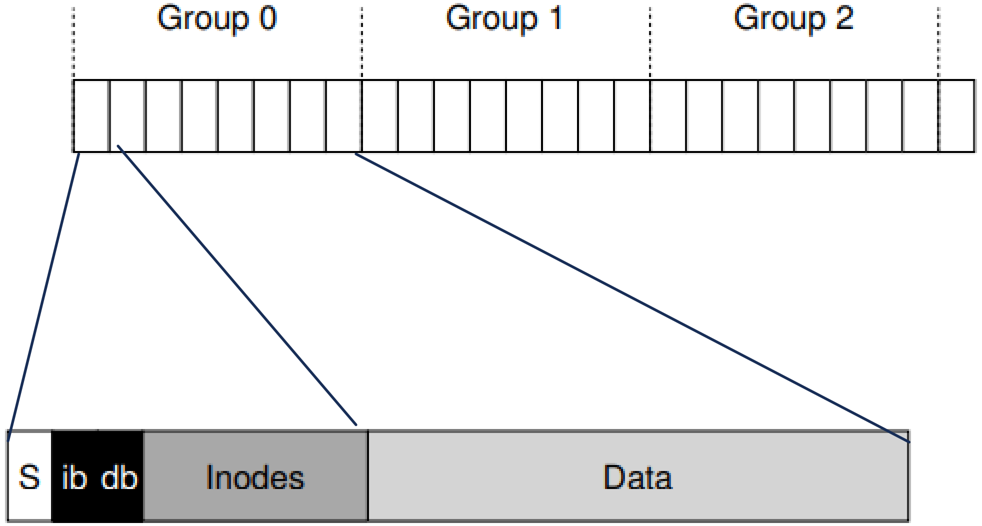
- Ken Thompson wrote the first file system.
- Very simple.
- Very poor performance:
  - Not disk aware, treating disk like random memory.
  - High fragmentation.
  - inodes can be allocated very far from their data.
  - Block size too small.
  - Overtime, performance was 2% of actual disk I/O bandwidth.



BERKLEY FAST FILE SYSTEM

Disk aware:

- Include structure information for each group: file system.
- Directory search, meta data access and modification same group as file access
- Overall, greatly increased performance.


Group 0
Group 1
Group 2
S
ib
db
Inodes
Data