# CS111, Lecture 4
## Unix V6 Filesystem, Continued

Optional reading:

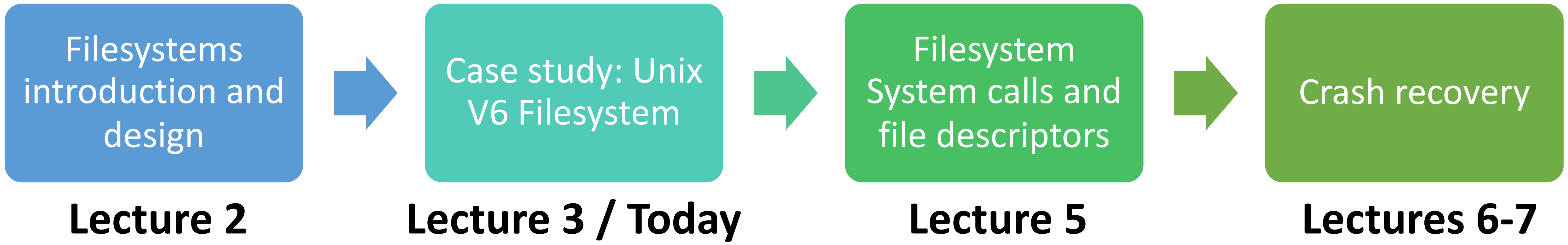
Operating Systems: Principles and Practice (2nd Edition): Sections 13.1-13.2

😷 masks strongly recommended

**Topic 1: Filesystems** - How can we design filesystems to manage files on disk, and what are the tradeoffs inherent in designing them? How can we interact with the filesystem in our programs?

# CS111 Topic 1: Filesystems

**Key Question:** *How can we design filesystems to manage files on disk, and what are the tradeoffs inherent in designing them? How can we interact with the filesystem in our programs?*


Filesystems introduction and design
Lecture 2
Case study: Unix V6 Filesystem
Lecture 3 / Today
Filesystem System calls and file descriptors
Lecture 5
Crash recovery
Lectures 6-7

**assign1:** implement portions of the Unix v6 filesystem!

## Learning Goals

- Explore the design of the Unix V6 filesystem
- Understand the design of the Unix v6 filesystem in how it represents directories
- Practice with the full process of going from file path to file data

## Plan For Today

- **Recap**: the Unix V6 Filesystem so far
- **Practice:** doubly-indirect addressing
- Directories and filename lookup
- **Practice:** filename lookup

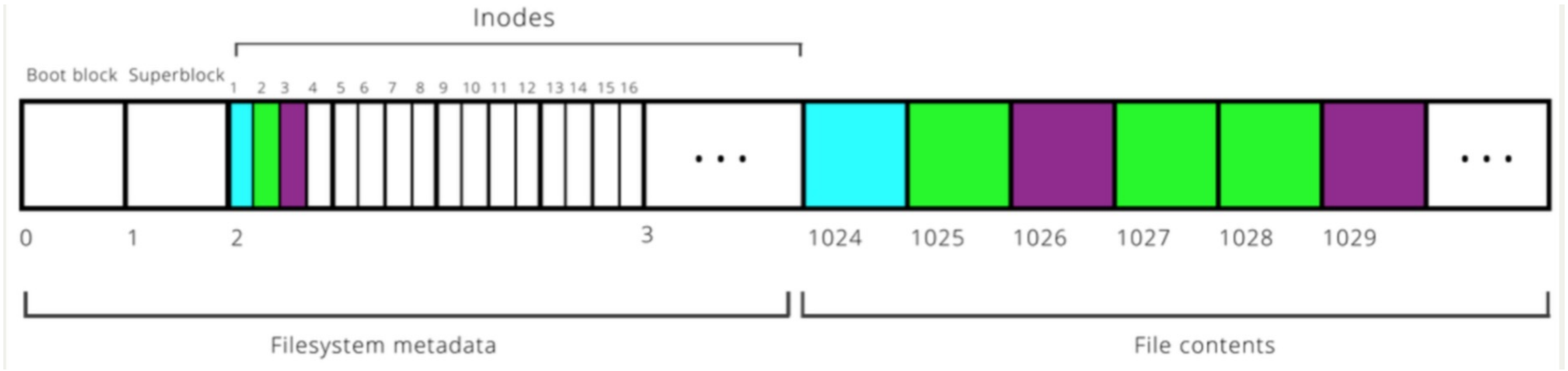
# Plan For Today

- **<u>Recap</u>: the Unix V6 Filesystem so far**
- **<u>Practice:</u>** doubly-indirect addressing
- Directories and filename lookup
- **<u>Practice:</u>** filename lookup

# Unix V6 Filesystem

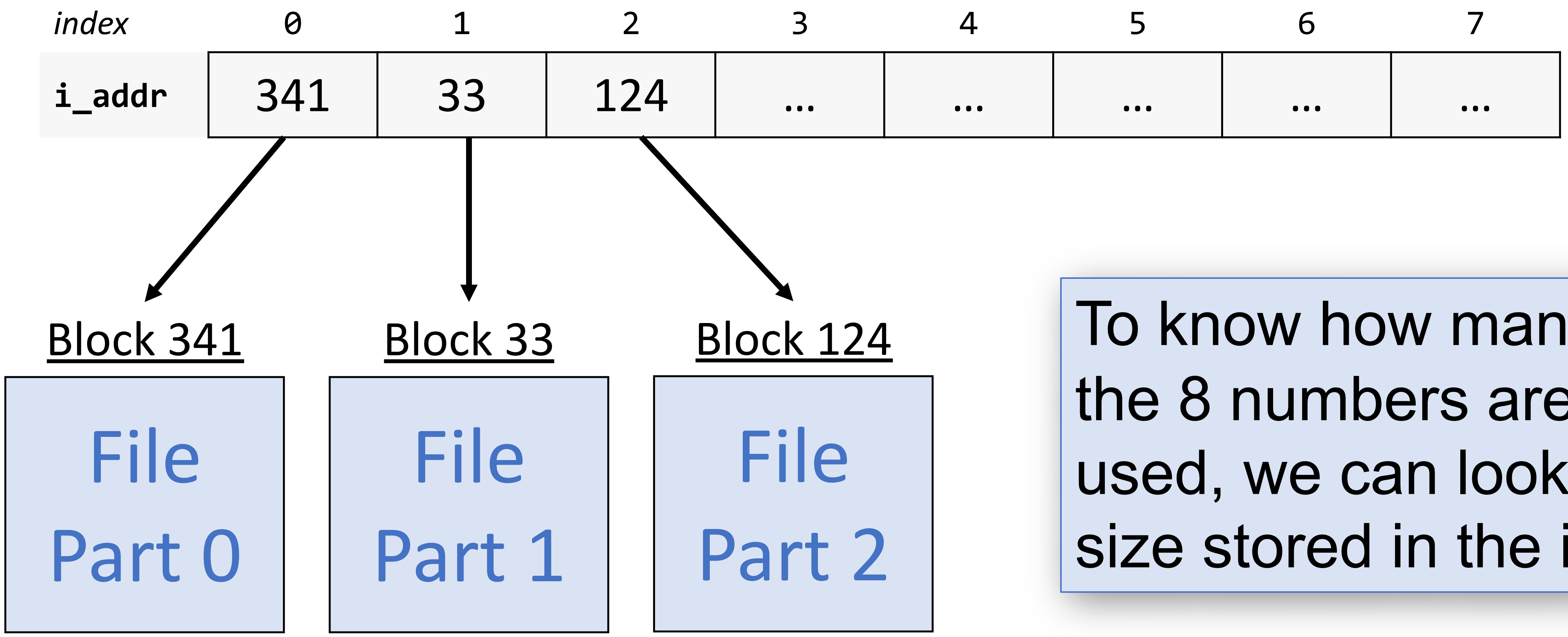
Every file has an associated inode.  An inode has space for up to 8 block numbers for file payload data, and this block number space is used differently depending on whether the file is "small mode" or "large mode"

```
if ((inode.i_mode & ILARG) != 0) { // file is "large mode"
```



# Small File Scheme

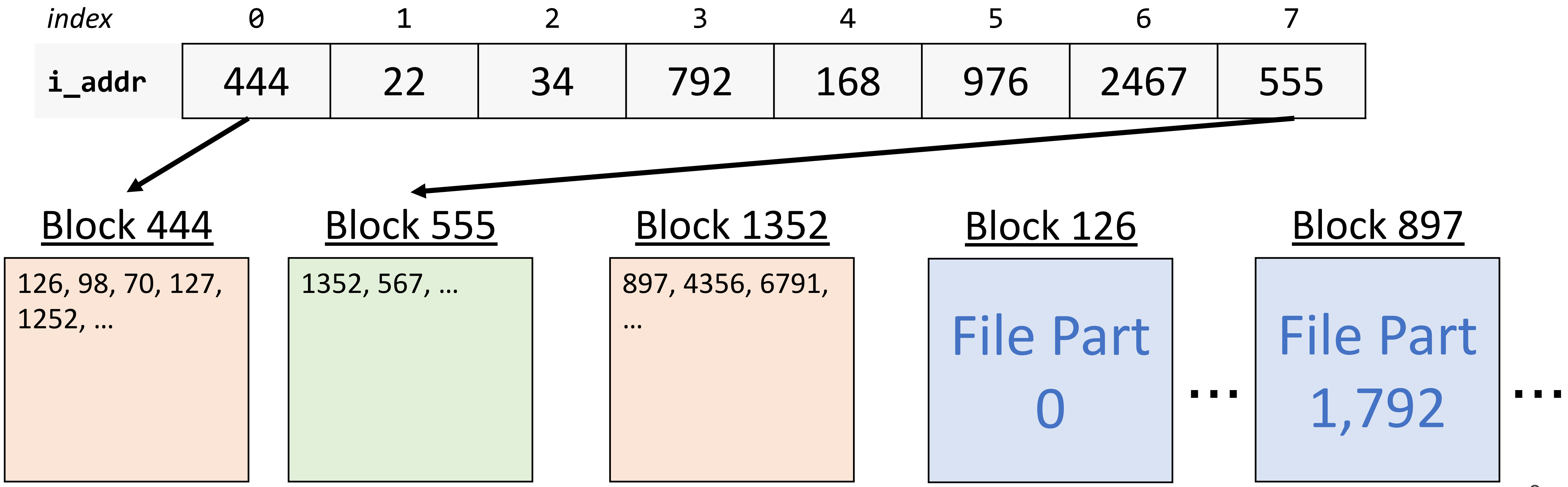
If the file is small, **i_addr** stores *direct block numbers*: numbers of blocks that contain payload data.



To know how many of the 8 numbers are used, we can look at the size stored in the inode.

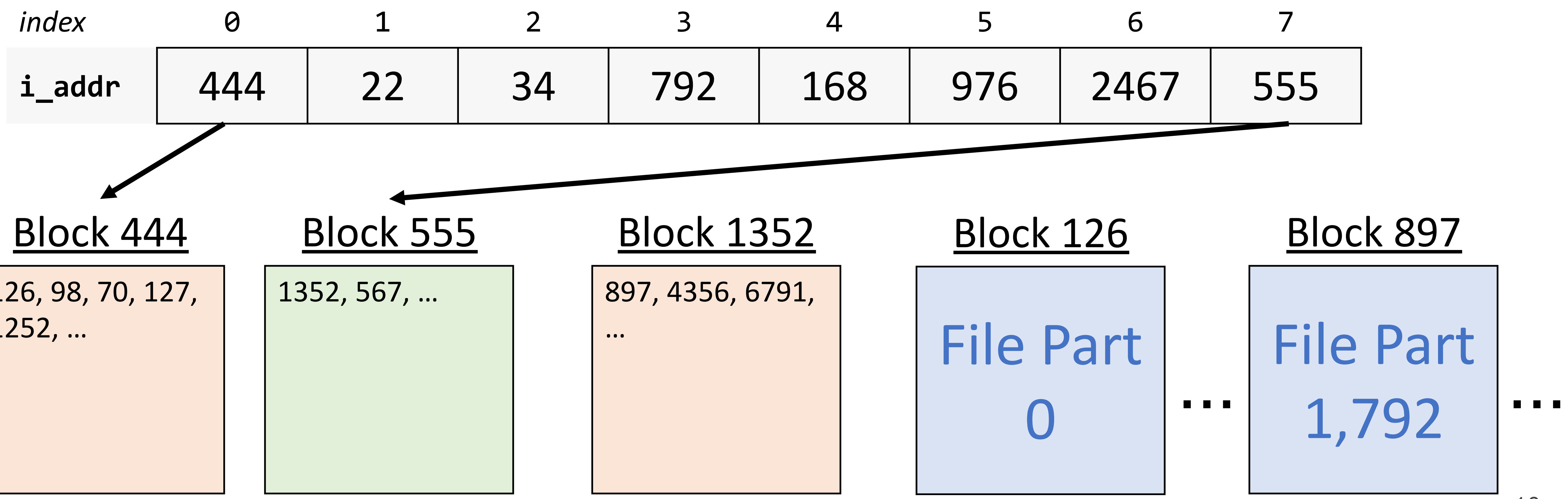
# Large File Scheme

If the file is large, the first 7 entries in **i_addr** are *singly-indirect block numbers* (block numbers of blocks that contain direct block numbers). The 8th entry (if needed) is a *doubly-indirect block number* (the number of a block that contains singly-indirect block numbers).



# Large File Scheme

**Another way to think about it:** a file can be represented using at most 7 + 256 = 263 singly-indirect blocks. The first seven are stored in the inode. The remaining 256 are stored in a block whose block number is stored in the inode.



## Large File Scheme

An inode for a large file stores 7 singly-indirect block numbers and 1 doubly-indirect block number. What is the largest file size this supports? Each block number is 2 bytes big.

*(7+256)* singly-indirect block numbers total x

*256* block numbers per singly-indirect block x

*512* bytes per block

= ~34MB

## Large File Scheme

An inode for a large file stores 7 singly-indirect block numbers and 1 doubly-indirect block number. What is the largest file size this supports? Each block number is 2 bytes big.

OR:

*(7 * 256 * 512) + (256 * 256 * 512) ~ 34MB*

*(singly indirect) + (doubly indirect)*

Better! still not sufficient for today's standards, but perhaps in 1975. Moreover, since block numbers are 2 bytes, we can number at most $2^{16}$ - 1 = 65,535 blocks, meaning the entire filesystem can be at most 65,535 * 512 ~ 32MB.

## Inodes

- Files only use the block numbers they need (depending on their size)
- Note: doubly-indirect is useful (and there are many other possible designs!), but it means even more steps to access data.

# Plan For Today

- **Recap**: the Unix V6 Filesystem so far
- **Practice: doubly-indirect addressing**
- Directories and filename lookup
- **Practice:** filename lookup

## Doubly-Indirect Addressing

What is the smallest file size (in bytes) that would require using the doubly-indirect block to store its data?

**Respond on PollEv:** pollev.com/cs111fall23 or text CS111FALL23 to 22333 once to join.

## What is the smallest file size (in bytes) that would require using the doubly-indirect block to store its data?

Nobody has responded yet.

Hang tight! Responses are coming in.

# Doubly-Indirect Addressing

What is the smallest file size (in bytes) that would require using the doubly-indirect block to store its data?

**Files up to (7 * 256 * 512) bytes are representable using just the 7 singly-indirect blocks. Files of (7 * 256 * 512) + 1 or more bytes would need the doubly-indirect block as well.**

# Doubly-Indirect Addressing

Assume we have a the following inode. How do we find the block containing the start of its payload data? How about the remainder of its payload data?

Inode 16:

- ”large mode”
- size = 18,855,234
- i_addr = [26,35,32,50,58,22,59,30]

**Step 1:** Go to block 26 and read block numbers. For the first number, 80, go to block 80 and read the beginning of the file (the first 512 bytes). Then go to block 41 for the next 512 bytes, etc.

| *Block #* | … | 2 | … | 26 | … | 30 | … | 80 | … | 87 | … | 89 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Block contents* | … | Inode table start | … | 80,41,82,85, 103, 24,45,… | … | 87,114,47,48, 122,99,111, 543,… | … | It was the best of times, it was the worst of times… | … | 89,448,234,99, … | … | “My father,” exclaimed Lucie, “you are ill!” |

# Doubly-Indirect Addressing

Assume we have a the following inode. How do we find the block containing the start of its payload data? How about the remainder of its payload data?

<u>Inode 16:</u>
- ”large mode”
- size = 18,855,234
- i_addr = [26,35,32,50,58,22,59,30]

**Step 2:** After 256 blocks, go to block 35, repeat the process. Do this a total of 7 times, for blocks 26, 35, 32, 50, 58, 22, and 59, reading 1792 blocks.

| *Block #* | … | 2 | | … | 26 | … | 30 | … | 80 | … | 87 | … | 89 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Block contents* | … | Inode table start | 🟦 | … | 80,41,82,85, 103, 24,45,… | … | 87,114,47,48, 122,99,111, 543,… | … | It was the best of times, it was the worst of times… | … | 89,448,234,99, … | … | “My father,” exclaimed Lucie, “you are ill!” |

# Doubly-Indirect Addressing

Assume we have a the following inode. How do we find the block containing the start of its payload data? How about the remainder of its payload data?

<u>Inode 16:</u>
- ”large mode”
- size = 18,855,234
- i_addr = [26,35,32,50,58,22,59,30]

**Step 3:** Go to block 30, which is a doubly-indirect block. From there, go to block 87, which is an indirect block. From there, go to block 89, which is the 1793rd (256*7 + 1) block.

| *Block #* | … | 2 | | … | 26 | … | 30 | … | 80 | … | 87 | … | 89 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Block contents* | … | Inode table start | 🟦 | … | 80,41,82,85, 103, 24,45,… | … | 87,114,47,48, 122,99,111, 543,… | … | It was the best of times, it was the worst of times… | … | 89,448,234,99, … | … | “My father,” exclaimed Lucie, “you are ill!” |

# Plan For Today

- **Recap**: the Unix V6 Filesystem so far
- **Practice:** doubly-indirect addressing
- **Directories and filename lookup**
- **Practice:** filename lookup

Now we understand how files are *stored*. But how do we *find* them?

## The Directory Hierarchy

Filesystems usually support directories ("folders")

- A directory can contain files and more directories
- A directory is a file container.  It needs to store information about what files/folders are contained within it.
- On Unix/Linux, all files live within the root directory, "/"
- We can specify the location of a file via the path to it from the root directory (“absolute path”):

```
/classes/cs111/index.html
```

**Common filesystem task:** given a filepath, get the file's contents.

# Directories

**Key idea:** Unix V6 directories are what map filenames to inode numbers in the filesystem. Filenames are *not* stored in inodes; they are stored in directories. Thefore, file lookup must happen via directories.

A Unix V6 directory contains an unsorted list of 16 byte “directory entries”. Each entry contains the name and inode number of one thing in that directory.

```
struct direntv6 {
  uint16_t d_inumber;
  char     d_name[14];
};
```

| | |
|---|---|
| 23 | myfile.txt |
| 54 | song.mp3 |
| 1245 | otherFolder |
| … | |

# Directories

Unix V6 directories contain lists of 16 byte “directory entries”.  Each entry contains the name and inode number of one thing in that directory.

- The first 14 bytes are the name (not necessarily null-terminated!)
- The last two bytes are the inumber

```
struct direntv6 {
  uint16_t d_inumber;
  char     d_name[14];
};
```

| | |
|---|---|
| 23 | myfile.txt |
| 54 | song.mp3 |
| 1245 | prez.pptx |
| … | |

## How can we use this directory representation to translate from a filepath to its inode number?

## The Lookup Process

```
/classes/cs111/index.html
```

Start at the root directory

## The Lookup Process

```
/classes/cs111/index.html
```

In the root directory, find the entry named "classes".

## The Lookup Process

```
/classes/cs111/index.html
```

In the "classes" directory, find the entry named "cs111".

## The Lookup Process

```
/classes/cs111/index.html
```

In the "cs111" directory, find the entry named "index.html". Then read its contents.

# Directories

How can we store directories on disk?

- Directories store directory entries – could be many entries
- Directories also have associated metadata (size, permissions, creation date, …)

**Key idea:** let's model a directory as a *file*. We'll pretend it's a "file" whose contents are its directory entries! Each directory will have an inode, too.

**Key benefit:** we can leverage all the existing logic for how files and inodes work, no need for extra work or complexity!

- Inodes can store a field telling us whether something is a directory or file.
- Directories can be "small mode" or "large mode", just like files

## The Lookup Process

The root directory ("/") is set to have inumber 1. That way we always know where to go to start traversing. (0 is reserved to mean "NULL" or "no inode").

## The Lookup Process

```
/classes/cs111/index.html
```

Go to inode with inumber 1 (root directory).

# The Lookup Process

```
/classes/cs111/index.html
```

In its payload data, look for the entry "classes" and get its inumber. Go to that inode.

## The Lookup Process

```
/classes/cs111/index.html
```

In its payload data, look for the entry "cs111" and get its inumber. Go to that inode.

## The Lookup Process

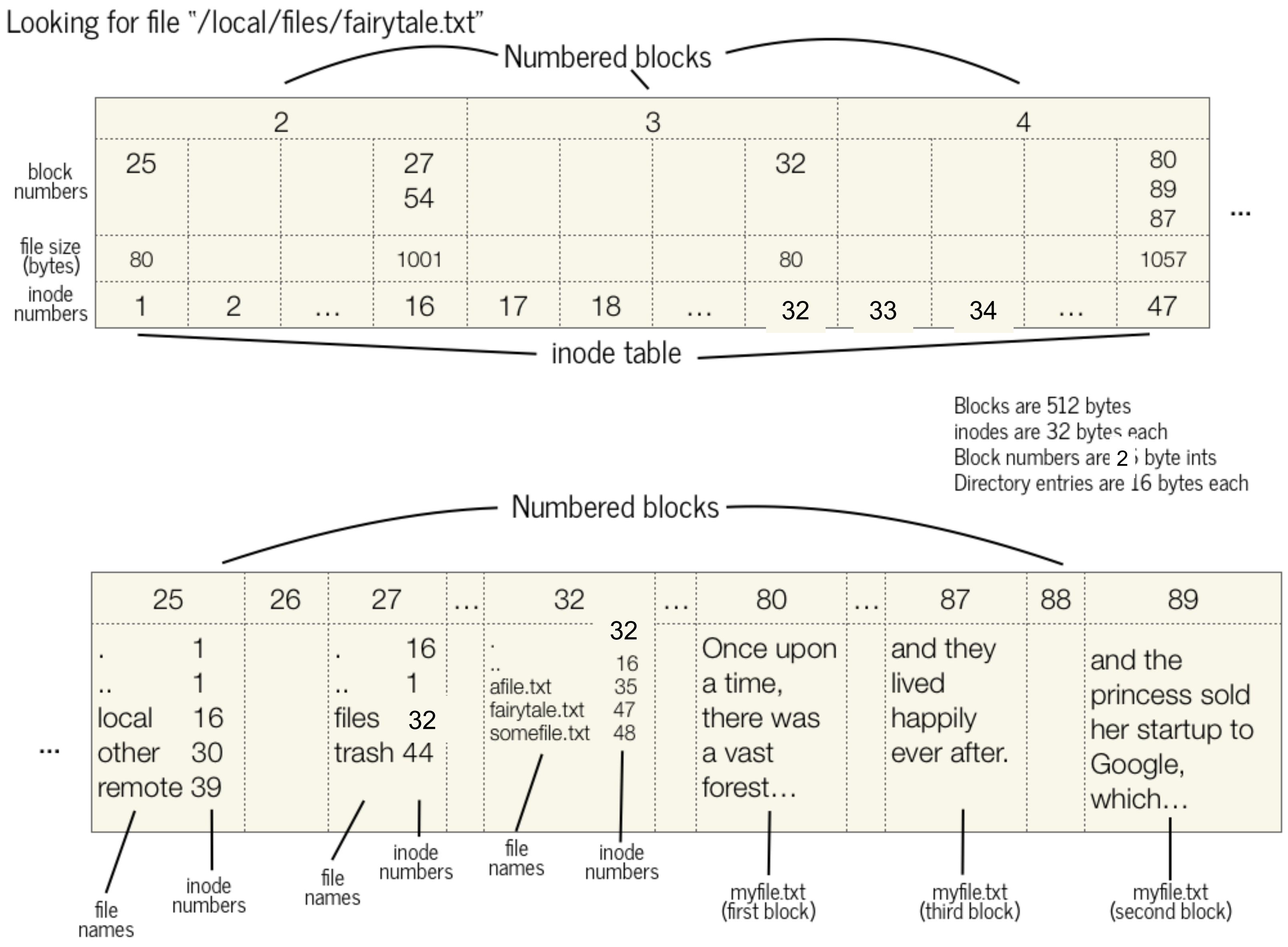
```
/classes/cs111/index.html
```

In its payload data, look for the entry "index.html" and get its inumber. Go to that inode and read in its payload data.

# Plan For Today

- Recap: the Unix V6 Filesystem so far
- Practice: doubly-indirect addressing
- Directories and filename lookup
- **Practice: filename lookup**

# Ex.: Finding "/local/files/fairytale.txt" (small file)



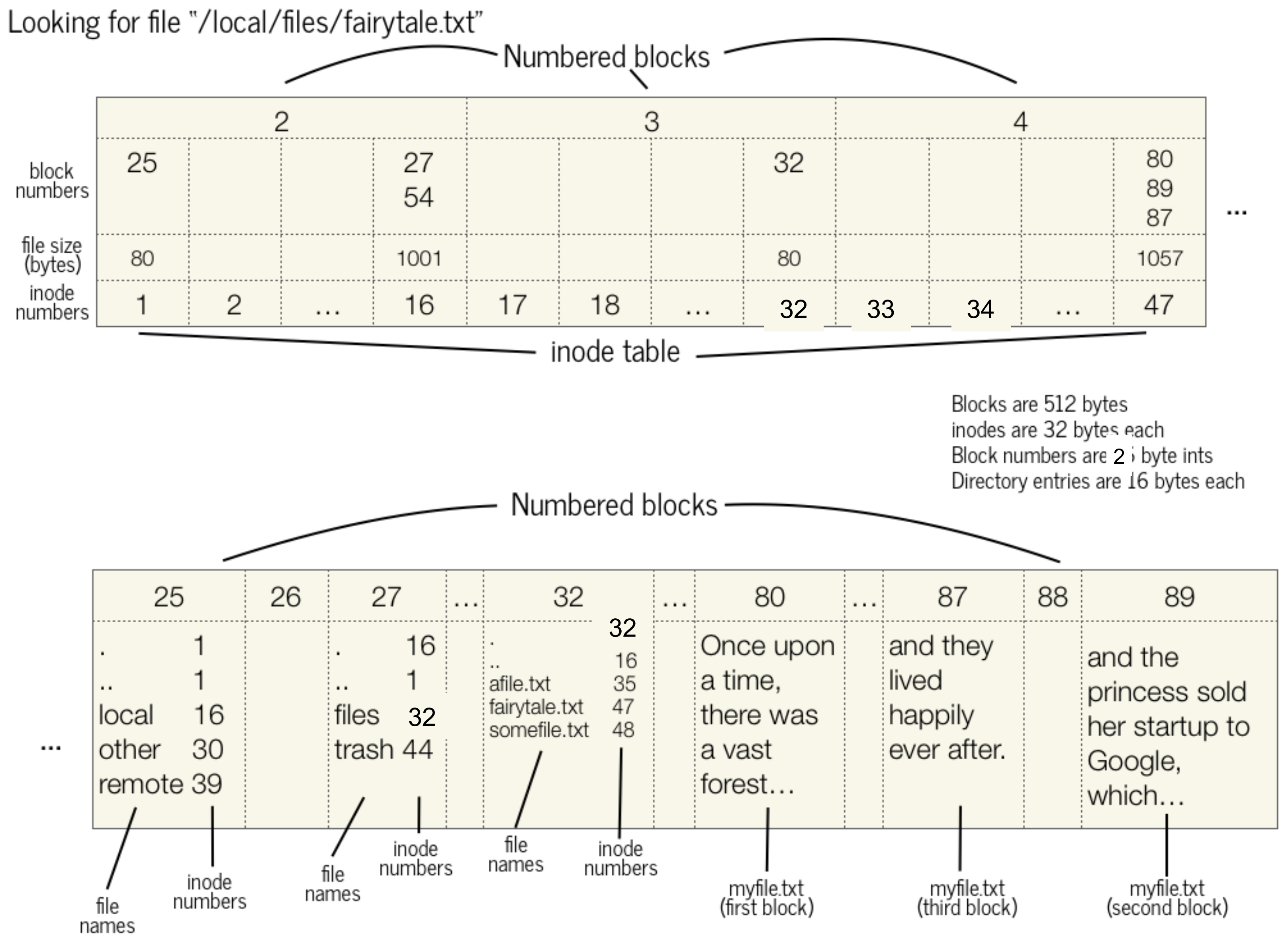
Blocks are 512 bytes
inodes are 32 bytes each
Block numbers are 2 byte ints
Directory entries are 16 bytes each

1. go to inode 1. It's small. We need to look in block 25 for the list of its entries.

2. Look in block 25 for "local" -> inode 16.

3. Go to inode 16. It's small. We need to look in blocks 27/54 for the list of its entries.

# Ex.: Finding "/local/files/fairytale.txt" (small file)

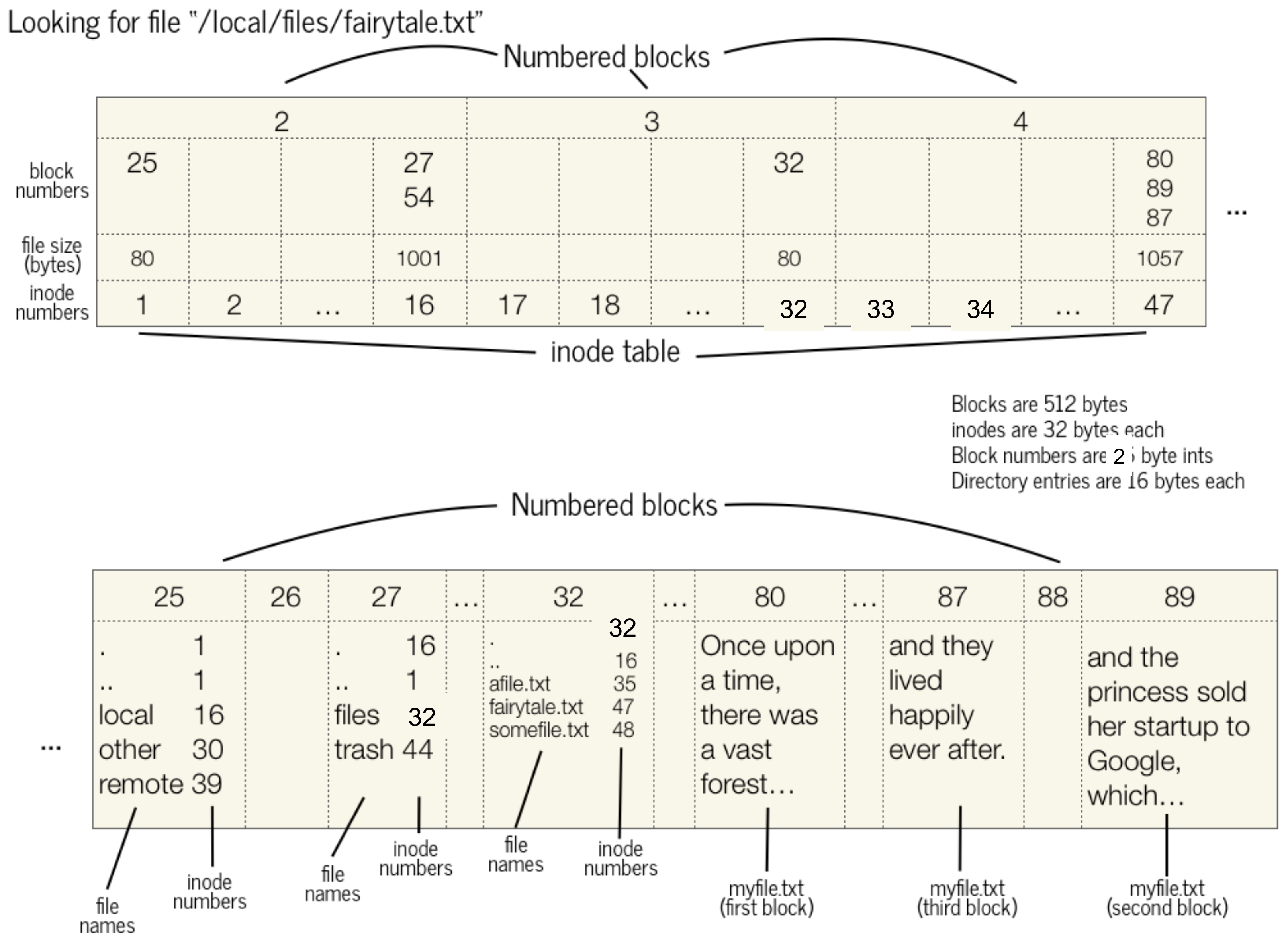


4. Look in block 27 for "files" (and then 54 if necessary) -> inode 32.

5. Go to inode 32. It's small. We need to look in block 32 for the list of its entries.

6. Look in block 32 for "fairytale.txt" -> inode 47.

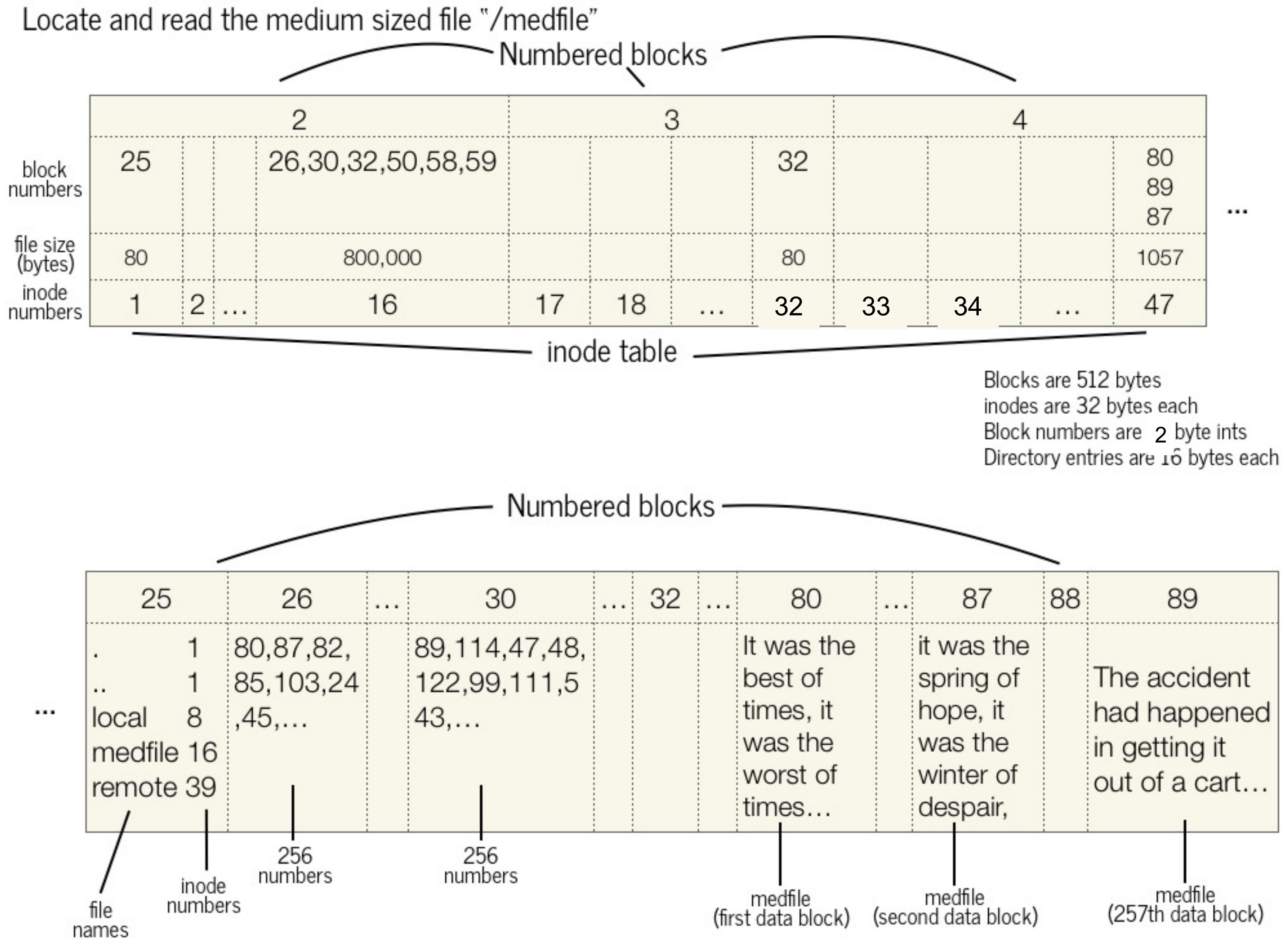
# Ex.: Finding "/local/files/fairytale.txt" (small file)



Blocks are 512 bytes
inodes are 32 bytes each
Block numbers are 2 byte ints
Directory entries are 16 bytes each

7. go to inode 47. It's small. We need to look in blocks 80,89,87 in order for its 1,057 bytes of payload data.

# Ex.: Finding "/medfile" (large file)



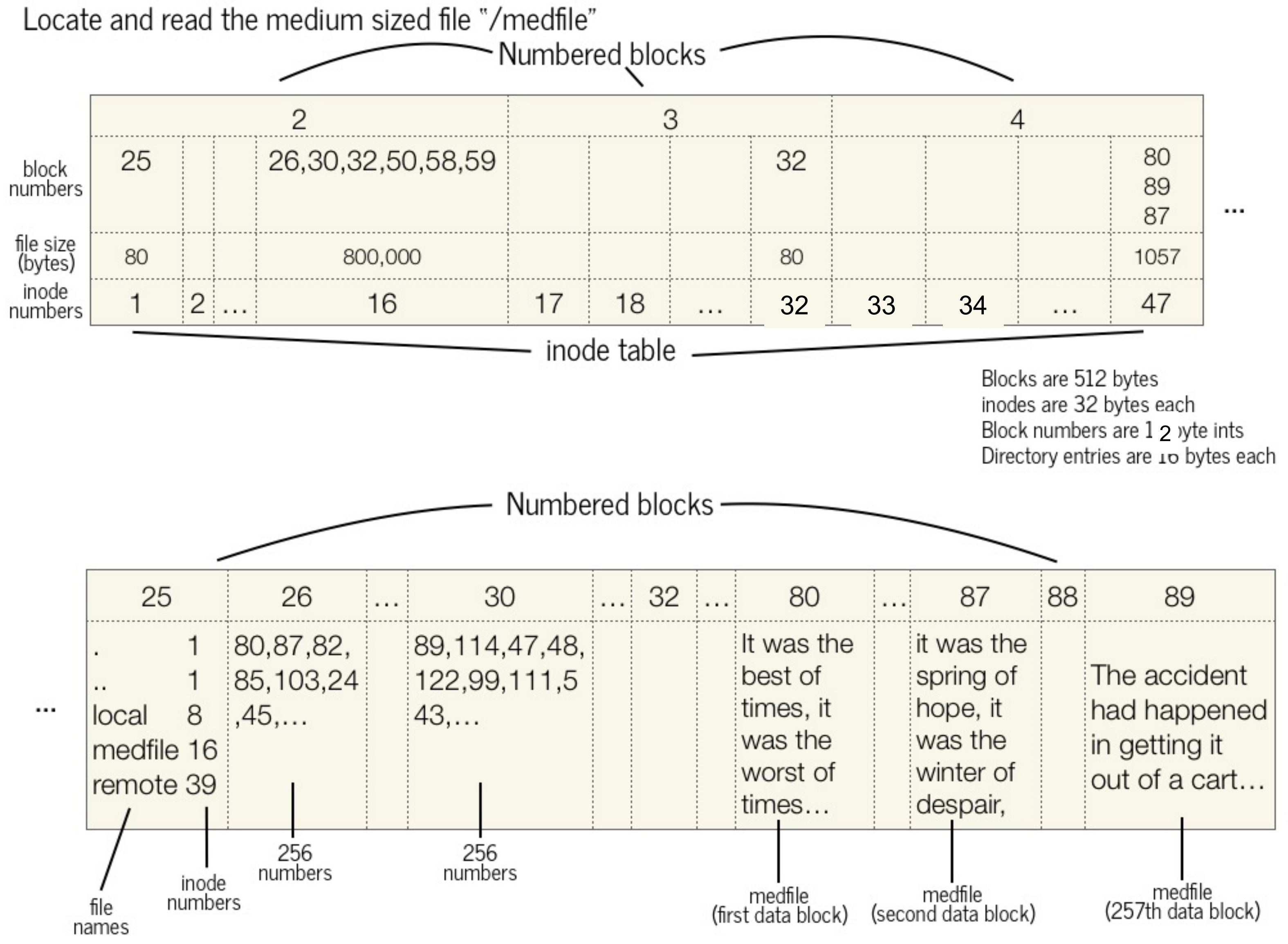
Locate and read the medium sized file "/medfile"
Numbered blocks
2
3
4
block numbers
25
26,30,32,50,58,59
32
80 89 87
file size (bytes)
80
800,000
80
1057
inode numbers
1
2 …
16
17
18
…
32
33
34
…
47
…
inode table
Blocks are 512 bytes
inodes are 32 bytes each
Block numbers are 2 byte ints
Directory entries are 16 bytes each
Numbered blocks
25
26
…
30
…
32
…
80
…
87
88
89
. 1
.. 1
local 8
medfile 16
remote 39
80,87,82, 85,103,24 ,45,…
89,114,47,48, 122,99,111,5 43,…
It was the best of times, it was the worst of times…
it was the spring of hope, it was the winter of despair,
The accident had happened in getting it out of a cart…
…
file names
inode numbers
256 numbers
256 numbers
medfile (first data block)
medfile (second data block)
medfile (257th data block)

1. go to inode 1. It's small. We need to look in block 25 for the list of its entries.

2. Look in block 25 for "medfile" -> inode 16.

3. Go to inode 16. It's large. We need to look in block 26 for the first 256 payload block numbers.

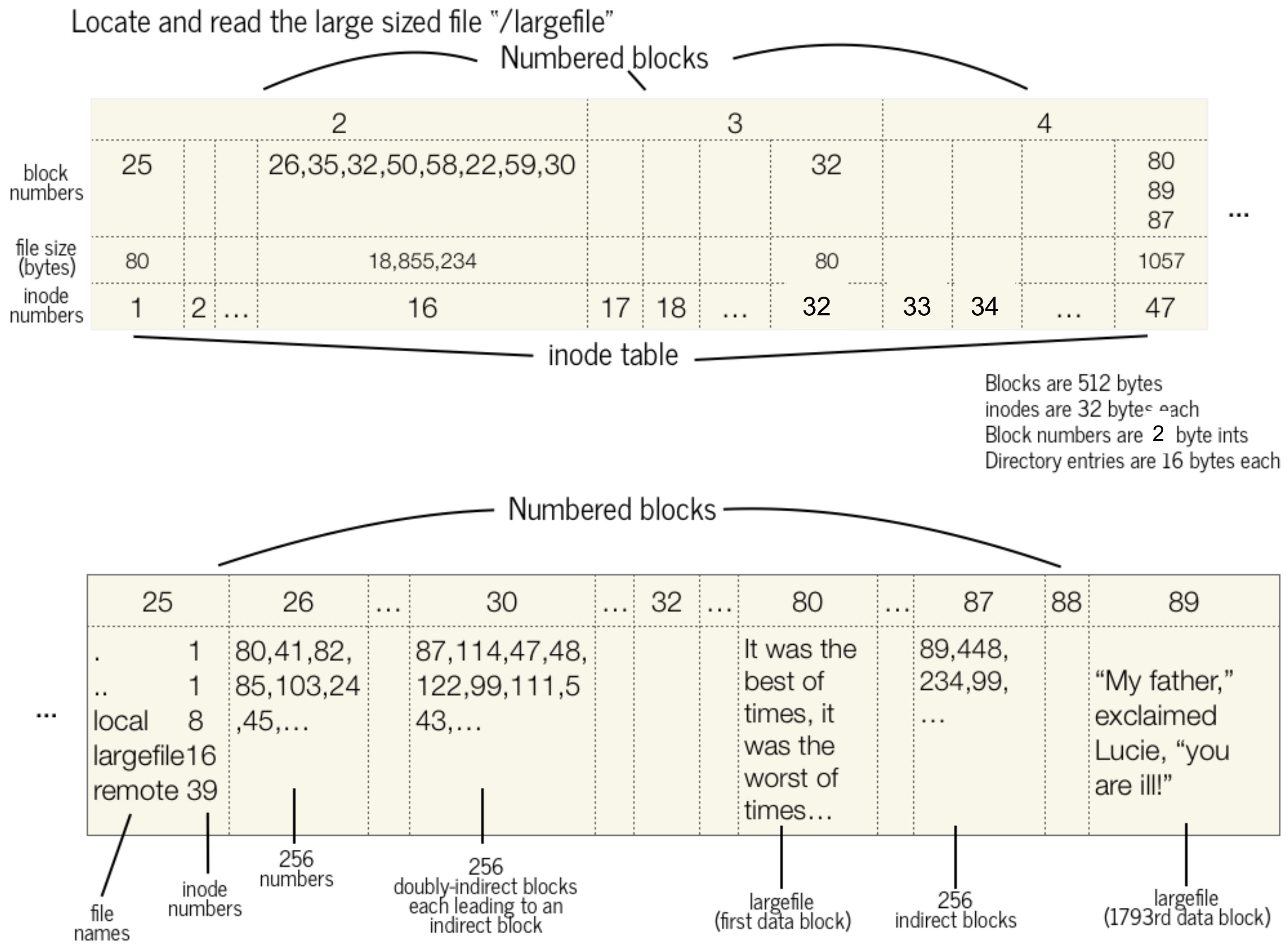
# Ex.: Finding "/medfile" (large file)



4. Read through numbers in block 26. First, go to block 80 for the first 512 payload bytes. Then, go to block 87 for the second 512 payload bytes.

5. After doing this 256 times, go to block 30 and repeat. Then continue with all remaining singly-indirect blocks in the inode.

# Ex.: Finding "/largefile" (large file)



Locate and read the large sized file "/largefile"

Numbered blocks

| | 2 | | | | 3 | | | | 4 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| block numbers | 25 | | | 26,35,32,50,58,22,59,30 | | | | 32 | | | | 80 89 87 | ... |
| file size (bytes) | 80 | | | 18,855,234 | | | | 80 | | | | 1057 | |
| inode numbers | 1 | 2 | ... | 16 | 17 | 18 | ... | 32 | 33 | 34 | ... | 47 | |

inode table

Blocks are 512 bytes
inodes are 32 bytes each
Block numbers are 2 byte ints
Directory entries are 16 bytes each

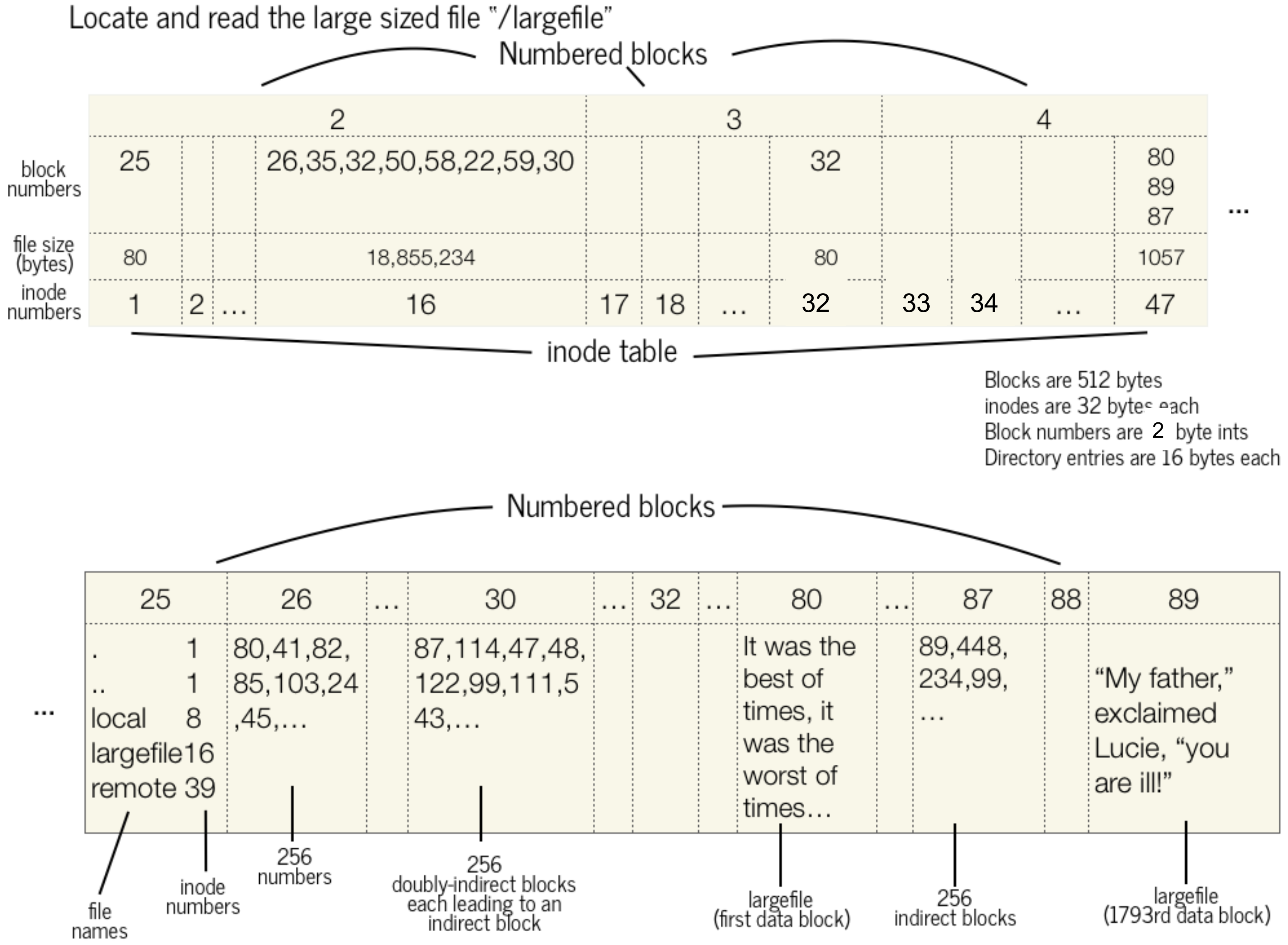
Numbered blocks

| | 25 | 26 | ... | 30 | ... | 32 | ... | 80 | ... | 87 | 88 | 89 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ... | . 1<br>.. 1<br>local 8<br>largefile 16<br>remote 39 | 80,41,82, 85,103,24 ,45,... | | 87,114,47,48, 122,99,111,5 43,... | | | | It was the best of times, it was the worst of times... | | 89,448, 234,99, ... | | "My father," exclaimed Lucie, "you are ill!" |

file names
inode numbers
256 numbers
256 doubly-indirect blocks each leading to an indirect block
largefile (first data block)
256 indirect blocks
largefile (1793rd data block)

**Question:** What is the number of the block that stores the first 512 bytes of **largefile**?

# Ex.: Finding "/largefile" (large file)

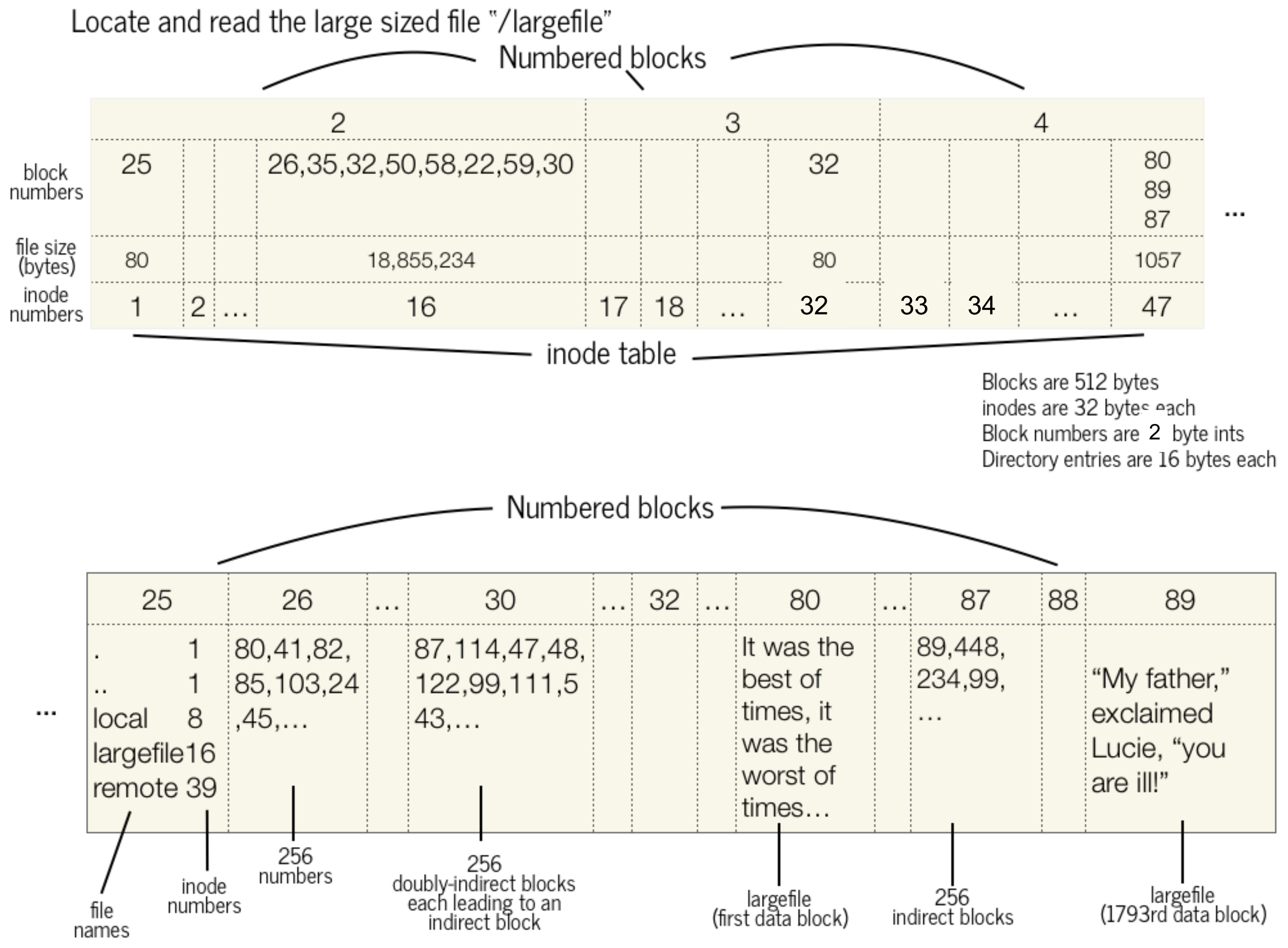


1. go to inode 1. It's small. We need to look in block 25 for the list of its entries.

2. Look in block 25 for "largefile" -> inode 16.

3. Go to inode 16. It's large. For the first seven block numbers, go to those blocks and read their 256 block numbers to get payload blocks.

## Ex.: Finding "/largefile" (large file)



4. For the eighth block, go to block 30. For each block number, go to that block number and read in its block numbers to get payload blocks.

**First payload block number = 80**.

# Recap

- **Recap**: the Unix V6 Filesystem so far
- **Practice:** doubly-indirect addressing
- Directories and filename lookup
- **Practice:** filename lookup

**Lecture 4 takeaway:** The Unix V6 Filesystem represents directories as files, with payloads containing directory entries. Lookup begins at the root directory for absolute paths.

**Next time:** how do we interact with the filesystem in our programs?