CS111, Lecture 4
Unix V6 Filesystem, Continued

Optional reading:

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

Announcements

Sections start this week! Check the course website for your section assignment. Bring a laptop with you if you have one.

- Sections rely on material through each Wed. lecture - the work you do in section will pay dividends when you work on the assignment!
- Checkoff sheet to track participation - section credit is awarded based on your sincere participation for the full section period
- if you have any section accommodation needs (e.g. illness) or need to attend a makeup, or have other section-logistics-related questions, please contact your section TA

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	→	Case study: Unix V6 Filesystem	→	Filesystem System calls and file descriptors	→	Crash recovery
Lecture 2		**Lecture 3 / Today**		**Lecture 5**		**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

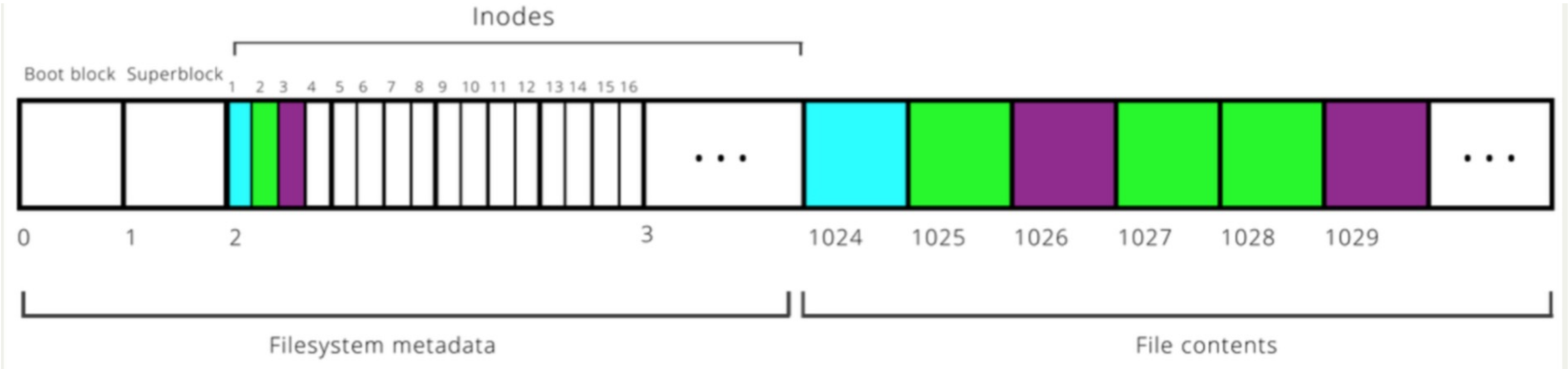
Plan For Today

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

Unix V6 Filesystem

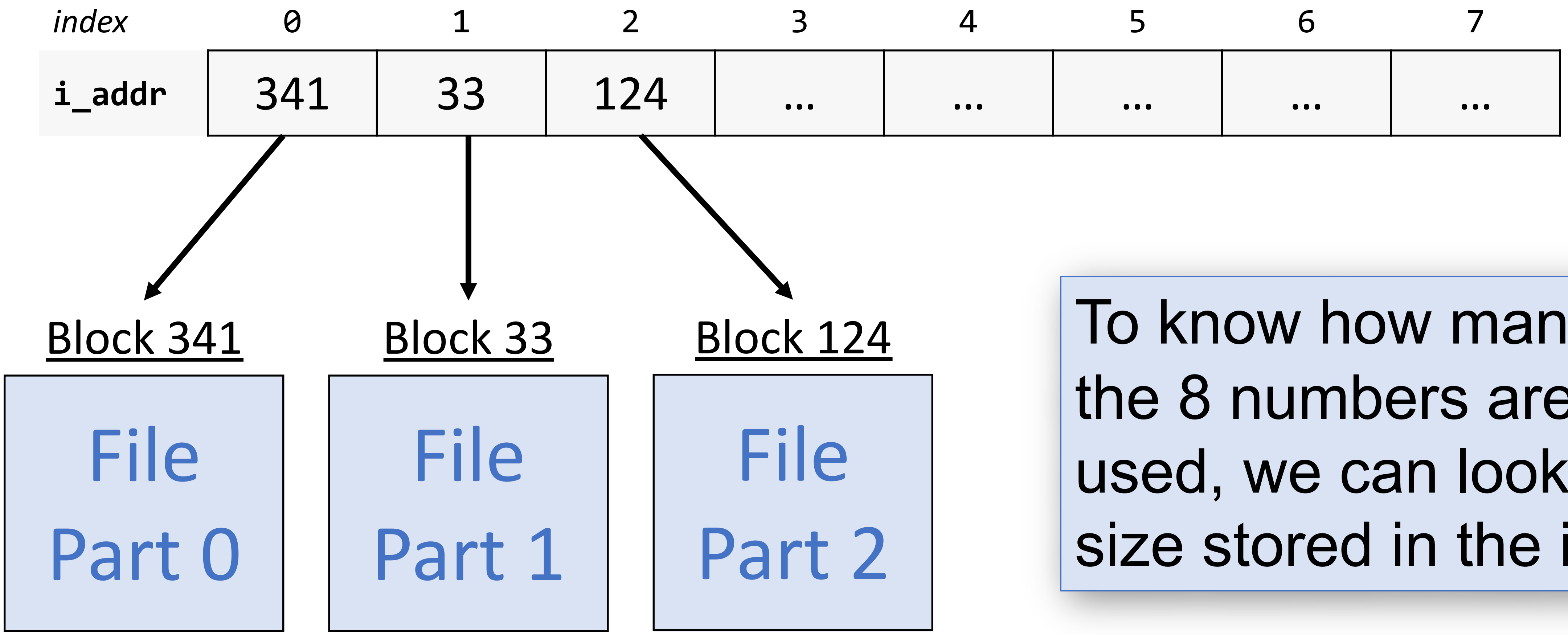
Every file has an associated inode. An inode has space for up to 8 block numbers for file payload data, and this block number space is used differently depending on whether the file is "small mode" or "large mode"

```
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.

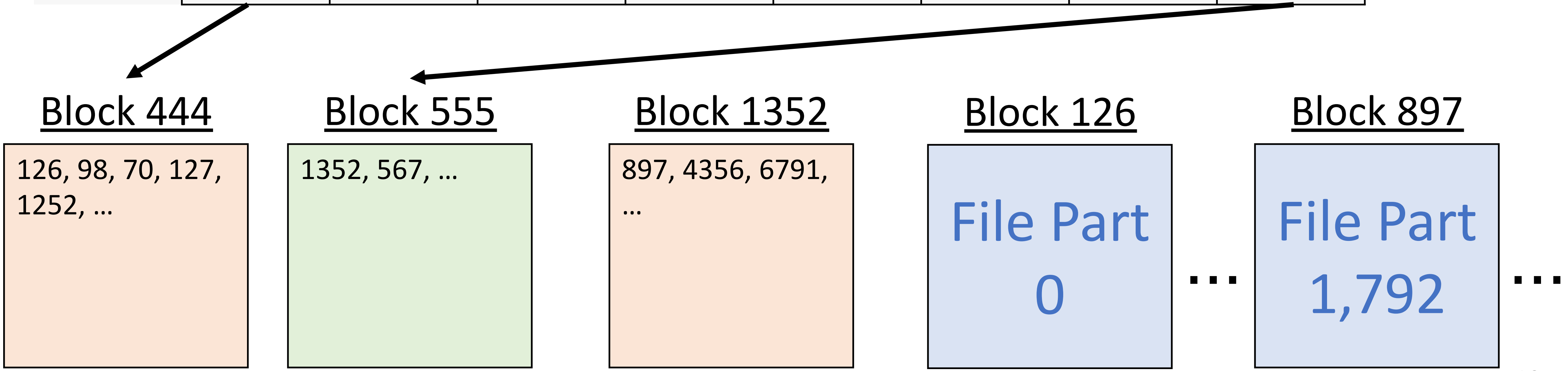


To know how many of the 8 numbers are used, we can look at the size stored in the inode.

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).

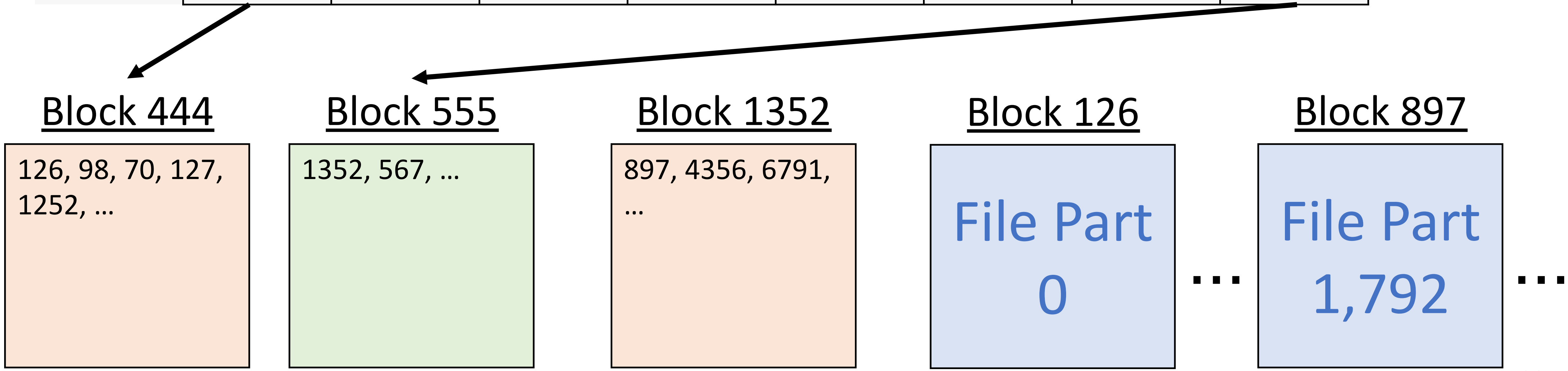
index	0	1	2	3	4	5	6	7
i_addr	444	22	34	792	168	976	2467	555



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.

index	0	1	2	3	4	5	6	7
`i_addr`	444	22	34	792	168	976	2467	555



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/cs111
or text CS111 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?

<u>Inode 16:</u>

- ”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?

Inode 16:
- "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?

Inode 16:
- ”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 a singly-indirect block, and read all block numbers. Repeat for remaining singly-indirect block numbers in block 30.

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	prez.pptx
…	

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

```
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**

Filename Lookup Practice #1

What is the inode number for the file with path /local/files/story.txt?

Inode #	1	…	3	…	12	…	14	…	16
Inode contents	Type: dir Mode: small Size: 80 i_addr = [24, …]	…	Type: file Mode: small Size: 1536 i_addr = [128, 222, 124, …]	…	Type: dir Mode: small Size: 544 i_addr = [32, 41, …]	…	Type: dir Mode: small Size: 64 i_addr = [62, …]	…	Type: file Mode: large Size: 4608 i_addr = [125, …]

Block #	…	2	…	24	…	32	…	41	…	62	…	128	…
Block contents	…	*Start of inode table*	…	. 1 .. 1 local 12 other 10 remote 9	…	. 12 .. 1 file1.txt 4 docs 15 ... ("files" not here)	…	apps 21 files 14	…	. 14 .. 12 story.txt 3 todo.txt 16	…	Once upon a time…	…

Filename Lookup Practice #1

What is the inode number for the file with path /local/files/story.txt?

Inode #	1	…	3	…	12	…	14	…	16
Inode contents	Type: dir Mode: small Size: 80 i_addr = [24, …]	…	Type: file Mode: small Size: 1536 i_addr = [128, 222, 124, …]	…	Type: dir Mode: small Size: 544 i_addr = [32, 41, …]	…	Type: dir Mode: small Size: 64 i_addr = [62, …]	…	Type: file Mode: large Size: 4608 i_addr = [125, …]

Block #	…	2	…	24	…	32	…	41	…	62	…	128	…
Block contents	…	*Start of inode table*	…	. 1 .. 1 local 12 other 10 remote 9	…	. 12 .. 1 file1.txt 4 docs 15 ... ("files" not here)	…	apps 21 files 14	…	. 14 .. 12 story.txt 3 todo.txt 16	…	Once upon a time…	…

Filename Lookup Practice #1

What is the inode number for the file with path /local/files/story.txt?

Inode #	1	…	3	…	12	…	14	…	16

Inode contents	`Type: dir`
	`Mode: small`
	`Size: 80`
	`i_addr = [24, …]`
		…	`Type: file`
			`Mode: small`
			`Size: 1536`
			`i_addr = [128, 222, 124, …]`
				…	`Type: dir`
					`Mode: small`
					`Size: 544`
					`i_addr = [32, 41, …]`
						…	`Type: dir`
							`Mode: small`
							`Size: 64`
							`i_addr = [62, …]`
								…	`Type: file`
									`Mode: large`
									`Size: 4608`
									`i_addr = [125, …]`

Block #	…	2	…	24	…	32	…	41	…	62	…	128	…
Block contents	…	*Start of inode table*	…	`. 1` `.. 1` `local 12` `other 10` `remote 9`	…	`. 12` `.. 1` `file1.txt 4` `docs 15` `...` `("files" not here)`	…	`apps 21` `files 14`	…	`. 14` `.. 12` `story.txt 3` `todo.txt 16`	…	Once upon a time…	…

Filename Lookup Practice #1

*What is the inode number for the file with path /**local**/files/story.txt?*

Inode #	1	…	3	…	12	…	14	…	16

Inode contents	`Type: dir`
	`Mode: small`
	`Size: 80`
	`i_addr = [24, …]`
		…	`Type: file`
			`Mode: small`
			`Size: 1536`
			`i_addr = [128, 222, 124, …]`
				…	`Type: dir`
					`Mode: small`
					`Size: 544`
					`i_addr = [32, 41, …]`
						…	`Type: dir`
							`Mode: small`
							`Size: 64`
							`i_addr = [62, …]`
								…	`Type: file`
									`Mode: large`
									`Size: 4608`
									`i_addr = [125, …]`

Block #	…	2	…	24	…	32	…	41	…	62	…	128	…
Block contents	…	*Start of inode table*	…	`. 1` `.. 1` `local 12` `other 10` `remote 9`	…	`. 12` `.. 1` `file1.txt 4` `docs 15` `...` `("files" not here)`	…	`apps 21` `files 14`	…	`. 14` `.. 12` `story.txt 3` `todo.txt 16`	…	Once upon a time…	…

Filename Lookup Practice #1

*What is the inode number for the file with path /**local**/files/story.txt?*

Inode #	1	…	3	…	12	…	14	…	16
Inode contents	Type: dir Mode: small Size: 80 i_addr = [24, …]	…	Type: file Mode: small Size: 1536 i_addr = [128, 222, 124, …]	…	Type: dir Mode: small Size: 544 i_addr = [32, 41, …]	…	Type: dir Mode: small Size: 64 i_addr = [62, …]	…	Type: file Mode: large Size: 4608 i_addr = [125, …]

Block #	…	2	…	24	…	32	…	41	…	62	…	128	
Block contents	…	*Start of inode table*	…	. 1 .. 1 local 12 other 10 remote 9	…	. 12 .. 1 file1.txt 4 docs 15 ... ("files" not here)	…	apps 21 files 14	…	. 14 .. 12 story.txt 3 todo.txt 16	…	Once upon a time…	…

Filename Lookup Practice #1

*What is the inode number for the file with path /**local**/files/story.txt?*

Inode #	1	…	3	…	12	…	14	…	16

Inode contents	`Type: dir`
	`Mode: small`
	`Size: 80`
	`i_addr = [24, …]`
		…	`Type: file`
			`Mode: small`
			`Size: 1536`
			`i_addr = [128, 222, 124, …]`
				…	`Type: dir`
					`Mode: small`
					`Size: 544`
					`i_addr = [32, 41, …]`
						…	`Type: dir`
							`Mode: small`
							`Size: 64`
							`i_addr = [62, …]`
								…	`Type: file`
									`Mode: large`
									`Size: 4608`
									`i_addr = [125, …]`

Block #	…	2	…	24	…	32	…	41	…	62	…	128	
Block contents	…	*Start of inode table*	…	`. 1` `.. 1` `local 12` `other 10` `remote 9`	…	`. 12` `.. 1` `file1.txt 4` `docs 15` `...` `("files" not here)`	…	`apps 21` `files 14`	…	`. 14` `.. 12` `story.txt 3` `todo.txt 16`	…	Once upon a time…	…

Filename Lookup Practice #1

*What is the inode number for the file with path /local/**files**/story.txt?*

Inode #	1	…	3	…	12	…	14	…	16
Inode contents	Type: dir Mode: small Size: 80 i_addr = [24, …]	…	Type: file Mode: small Size: 1536 i_addr = [128, 222, 124, …]	…	Type: dir Mode: small Size: 544 i_addr = [32, 41, …]	…	Type: dir Mode: small Size: 64 i_addr = [62, …]	…	Type: file Mode: large Size: 4608 i_addr = [125, …]

Block #	…	2	…	24	…	32	…	41	…	62	…	128	…
Block contents	…	*Start of inode table*	…	. 1 .. 1 local 12 other 10 remote 9	…	. 12 .. 1 file1.txt 4 docs 15 ... ("files" not here)	…	apps 21 files 14	…	. 14 .. 12 story.txt 3 todo.txt 16	…	Once upon a time…	…

Filename Lookup Practice #1

*What is the inode number for the file with path /local/**files**/story.txt?*

Inode #	1	…	3	…	12	…	14	…	16

Inode contents	`Type: dir`
	`Mode: small`
	`Size: 80`
	`i_addr = [24, …]`
		…	`Type: file`
			`Mode: small`
			`Size: 1536`
			`i_addr = [128, 222, 124, …]`
				…	`Type: dir`
					`Mode: small`
					`Size: 544`
					`i_addr = [32, 41, …]`
						…	`Type: dir`
							`Mode: small`
							`Size: 64`
							`i_addr = [62, …]`
								…	`Type: file`
									`Mode: large`
									`Size: 4608`
									`i_addr = [125, …]`

Block #	…	2	…	24	…	32	…	41	…	62	…	128	…
Block contents	…	*Start of inode table*	…	`. 1` `.. 1` `local 12` `other 10` `remote 9`	…	`. 12` `.. 1` `file1.txt 4` `docs 15` `...` `("files" not here)`	…	`apps 21` `files 14`	…	`. 14` `.. 12` `story.txt 3` `todo.txt 16`	…	Once upon a time…	…

Filename Lookup Practice #1

*What is the inode number for the file with path /local/**files**/story.txt?*

Inode #	1	…	3	…	12	…	14	…	16
Inode contents	Type: dir Mode: small Size: 80 i_addr = [24, …]	…	Type: file Mode: small Size: 1536 i_addr = [128, 222, 124, …]	…	Type: dir Mode: small Size: 544 i_addr = [32, 41, …]	…	Type: dir Mode: small Size: 64 i_addr = [62, …]	…	Type: file Mode: large Size: 4608 i_addr = [125, …]

Block #	…	2	…	24	…	32	…	41	…	62	…	128	
Block contents	…	*Start of inode table*	…	. 1 .. 1 local 12 other 10 remote 9	…	. 12 .. 1 file1.txt 4 docs 15 ... ("files" not here)	…	apps 21 files 14	…	. 14 .. 12 story.txt 3 todo.txt 16	…	Once upon a time…	…

Filename Lookup Practice #2

What is the inode number for the file with path /usr/note.txt?

Inode #	1	…	3	…	12	…	14	…	16
Inode contents	Type: dir Mode: small Size: 96 i_addr = [56, …]	…	Type: dir Mode: large Size: 131584 i_addr = [122, 545, …]	…	Type: dir Mode: small Size: 544 i_addr = [32, 41, …]	…	Type: dir Mode: small Size: 64 i_addr = [62, …]	…	Type: file Mode: large Size: 4608 i_addr = [876, …]

Block #	…	2	…	56	…	67	…	122	…	421	…	545	
Block contents	…	*Start of inode table*	…	. 1 .. 1 bin 13 tmp 10 other 9 usr 3	…	. 3 .. 1 apps 21 files 14 … ("note.txt" not here)	…	67,421,872, 999,135,346 ,…	…	icon.png 30 doc.pdf 15 note.txt 16 …	…	565	…

Filename Lookup Practice #2

What is the inode number for the file with path /usr/note.txt?

Inode #	1	…	3	…	12	…	14	…	16

Inode contents	`Type: dir`
	`Mode: small`
	`Size: 96`
	`i_addr = [56, …]`
		…	`Type: dir`
			`Mode: large`
			`Size: 131584`
			`i_addr = [122, 545, …]`
				…	`Type: dir`
					`Mode: small`
					`Size: 544`
					`i_addr = [32, 41, …]`
						…	`Type: dir`
							`Mode: small`
							`Size: 64`
							`i_addr = [62, …]`
								…	`Type: file`
									`Mode: large`
									`Size: 4608`
									`i_addr = [876, …]`

Block #	…	2	…	56	…	67	…	122	…	421	…	545	…
Block contents	…	*Start of inode table*	…	`. 1` `.. 1` `bin 13` `tmp 10` `other 9` `usr 3`	…	`. 3` `.. 1` `apps 21` `files 14` `…` `("note.txt" not here)`	…	`67,421,872,` `999,135,346` `,…`	…	`icon.png 30` `doc.pdf 15` `note.txt 16` `…`	…	`565`	…

Filename Lookup Practice #2

What is the inode number for the file with path /usr/note.txt?

Inode #	1	…	3	…	12	…	14	…	16
Inode contents	Type: dir Mode: small Size: 96 i_addr = [56, …]	…	Type: dir Mode: large Size: 131584 i_addr = [122, 545, …]	…	Type: dir Mode: small Size: 544 i_addr = [32, 41, …]	…	Type: dir Mode: small Size: 64 i_addr = [62, …]	…	Type: file Mode: large Size: 4608 i_addr = [876, …]

Block #	…	2	…	56	…	67	…	122	…	421	…	545	
Block contents	…	*Start of inode table*	…	. 1 .. 1 bin 13 tmp 10 other 9 usr 3	…	. 3 .. 1 apps 21 files 14 … ("note.txt" not here)	…	67,421,872, 999,135,346 ,…	…	icon.png 30 doc.pdf 15 note.txt 16 …	…	565	…

Filename Lookup Practice #2

*What is the inode number for the file with path /**usr**/note.txt?*

Inode #	1	…	3	…	12	…	14	…	16

Inode contents	`Type: dir`
	`Mode: small`
	`Size: 96`
	`i_addr = [56, …]`
		…	`Type: dir`
			`Mode: large`
			`Size: 131584`
			`i_addr = [122, 545, …]`
				…	`Type: dir`
					`Mode: small`
					`Size: 544`
					`i_addr = [32, 41, …]`
						…	`Type: dir`
							`Mode: small`
							`Size: 64`
							`i_addr = [62, …]`
								…	`Type: file`
									`Mode: large`
									`Size: 4608`
									`i_addr = [876, …]`

Block #	…	2	…	56	…	67	…	122	…	421	…	545	…
Block contents	…	*Start of inode table*	…	`. 1` `.. 1` `bin 13` `tmp 10` `other 9` `usr 3`	…	`. 3` `.. 1` `apps 21` `files 14` `…` `("note.txt" not here)`	…	`67,421,872,` `999,135,346` `,…`	…	`icon.png 30` `doc.pdf 15` `note.txt 16` `…`	…	`565`	…

Filename Lookup Practice #2

*What is the inode number for the file with path /**usr**/note.txt?*

Inode #	1	…	3	…	12	…	14	…	16
Inode contents	Type: dir Mode: small Size: 96 i_addr = [56, …]	…	Type: dir Mode: large Size: 131584 i_addr = [122, 545, …]	…	Type: dir Mode: small Size: 544 i_addr = [32, 41, …]	…	Type: dir Mode: small Size: 64 i_addr = [62, …]	…	Type: file Mode: large Size: 4608 i_addr = [876, …]

Block #	…	2	…	56	…	67	…	122	…	421	…	545	
Block contents	…	*Start of inode table*	…	. 1 .. 1 bin 13 tmp 10 other 9 usr 3	…	. 3 .. 1 apps 21 files 14 … ("note.txt" not here)	…	67,421,872, 999,135,346 ,…	…	icon.png 30 doc.pdf 15 note.txt 16 …	…	565	…

Filename Lookup Practice #2

*What is the inode number for the file with path /**usr**/note.txt?*

Inode #	1	…	3	…	12	…	14	…	16
Inode contents	Type: dir Mode: small Size: 96 i_addr = [56, …]	…	Type: dir Mode: large Size: 131584 i_addr = [122, 545, …]	…	Type: dir Mode: small Size: 544 i_addr = [32, 41, …]	…	Type: dir Mode: small Size: 64 i_addr = [62, …]	…	Type: file Mode: large Size: 4608 i_addr = [876, …]

Block #	…	2	…	56	…	67	…	122	…	421	…	545	
Block contents	…	*Start of inode table*	…	. 1 .. 1 bin 13 tmp 10 other 9 usr 3	…	. 3 .. 1 apps 21 files 14 … ("note.txt" not here)	…	67,421,872, 999,135,346 ,…	…	icon.png 30 doc.pdf 15 note.txt 16 …	…	565	…

Filename Lookup Practice #2

*What is the inode number for the file with path /**usr**/note.txt?*

Inode #	1	…	3	…	12	…	14	…	16
Inode contents	Type: dir Mode: small Size: 96 i_addr = [56, …]	…	Type: dir Mode: large Size: 131584 i_addr = [122, 545, …]	…	Type: dir Mode: small Size: 544 i_addr = [32, 41, …]	…	Type: dir Mode: small Size: 64 i_addr = [62, …]	…	Type: file Mode: large Size: 4608 i_addr = [876, …]

Block #	…	2	…	56	…	67	…	122	…	421	…	545	…
Block contents	…	*Start of inode table*	…	. 1 .. 1 bin 13 tmp 10 other 9 usr 3	…	. 3 .. 1 apps 21 files 14 … ("note.txt" not here)	…	67,421,872, 999,135,346 ,…	…	icon.png 30 doc.pdf 15 note.txt 16 …	…	565	…

Filename Lookup Practice #2

*What is the inode number for the file with path /**usr**/note.txt?*

Inode #	1	…	3	…	12	…	14	…	16
Inode contents	Type: dir Mode: small Size: 96 i_addr = [56, …]	…	Type: dir Mode: large Size: 131584 i_addr = [122, 545, …]	…	Type: dir Mode: small Size: 544 i_addr = [32, 41, …]	…	Type: dir Mode: small Size: 64 i_addr = [62, …]	…	Type: file Mode: large Size: 4608 i_addr = [876, …]

Block #	…	2	…	56	…	67	…	122	…	421	…	545	
Block contents	…	*Start of inode table*	…	. 1 .. 1 bin 13 tmp 10 other 9 usr 3	…	. 3 .. 1 apps 21 files 14 … ("note.txt" not here)	…	67,421,872, 999,135,346 ,…	…	icon.png 30 doc.pdf 15 note.txt 16 …	…	565	…

Unix V6 Filesystem Summary

We built layers on top of the low-level **readSector** and **writeSector** to implement a higher-level filesystem. We encountered several design ideas:

- **Modularity** –subdivision of a larger system into a collection of smaller subsystems, which themselves may be further subdivided
- **Layering** –the organization of several modules that interact in some hierarchical manner where each layer typically only opens its interface to the module above it
- **Name resolution** – system resolves human-friendly names (paths) to machine-friendly names (inumbers). Names let us refer to system resources.
- **Virtualization** – making one thing look like another (e.g. disk is just an array of sectors)

Unix V6 Filesystem

The Unix V6 Filesystem is one example of a “multi-level index” filesystem design.

- What are the benefits / drawbacks of the Unix V6 Filesystem design?

<u>Advantages</u>

- Can access all block numbers for a file
- Still supports easy sequential access
- Easy to grow files

Unix V6 Filesystem

The Unix V6 Filesystem is one example of a "multi-level index" filesystem design.

- What are the benefits / drawbacks of the Unix V6 Filesystem design?

Disadvantages

- More steps and disk reads to get block data for large files
- More disk space taken up by metadata
- Upper limit on file size (though if larger than disk, doesn't matter)
- Size change requires restructuring the inode

Multi-level Indexes

There are many alternative designs that could be used – some alterations you could propose might be:

- What if the block size was different?
- What if inodes stored a different number of block numbers?
- What if the file size scheme (small / large) worked differently?

Example: 4.3 BSD Unix filesystem (evolutionary descendent of V6)

- 4Kb block size
- Inodes store 14 block numbers
- First 12 block numbers always direct, 13^{th} always singly indirect, 14^{th} always doubly indirect (no small vs. large schemes)

Other Filesystem Design Ideas

Larger block size? Improves efficiency of I/O and inodes but worsens internal fragmentation. Generally: challenges with both large and small files coexisting.

One idea: multiple block sizes

- Large blocks are 4KB, *fragments* are 512 bytes (8 fragments fit in a block)
- The last block in a file can be a fragment (0-7 fragments)
- One large block can hold fragments from multiple files
- Get the time efficiency benefit of larger blocks, but the internal fragmentation benefit of smaller blocks (small files can use fragments)

Filesystem Techniques Today

- Filesystem design is a hard problem! Tradeoffs, challenges with large and small files.
- Even larger block sizes (16KB large blocks, 2KB fragments) – disk space cheap, internal fragmentation doesn’t matter as much
- Reallocate files as blocks grow – initially allocate blocks one at a time, but when a file reaches a certain size, reallocate blocks looking for large contiguous clusters
- ext4 is a popular current Linux filesystem – you may notice similarities!
- NTFS (replacement for FAT) is the current Windows filesystem
- APFS (“Apple Filesystem”) is the filesystem for Apple devices

Assignment 1

Implement core functions to read from a Unix v6 filesystem disk!

- **inode_iget** -> fetch a specific inode
- **inode_indexlookup** -> fetch a specific payload block number
- **file_getblock** -> fetch a specified payload block
- **directory_findname** -> fetch directory entry with the given name
- **pathname_lookup** -> fetch inumber for the file with the given path

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?