Directories and Links

Mendel Rosenblum

Directories and Links

Optional readings: Operating Systems: Principles and Practice:

- Section 13.1-13.2

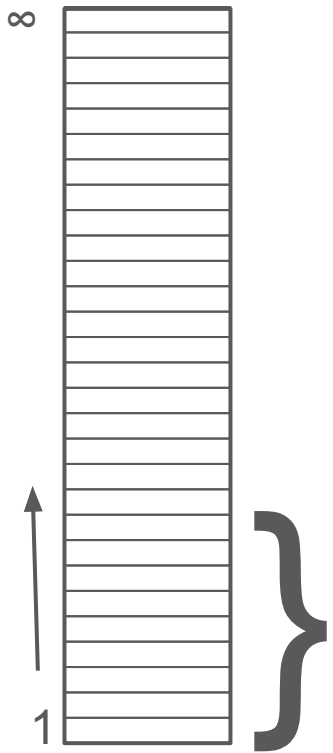
Naming: how do users refer to their files?

- Last week:
  - Find a file's blocks on disk using its inode
- Today:
  - How does OS find file, given name?

First step: inode has to be stored on disk, so it will survive system reboots

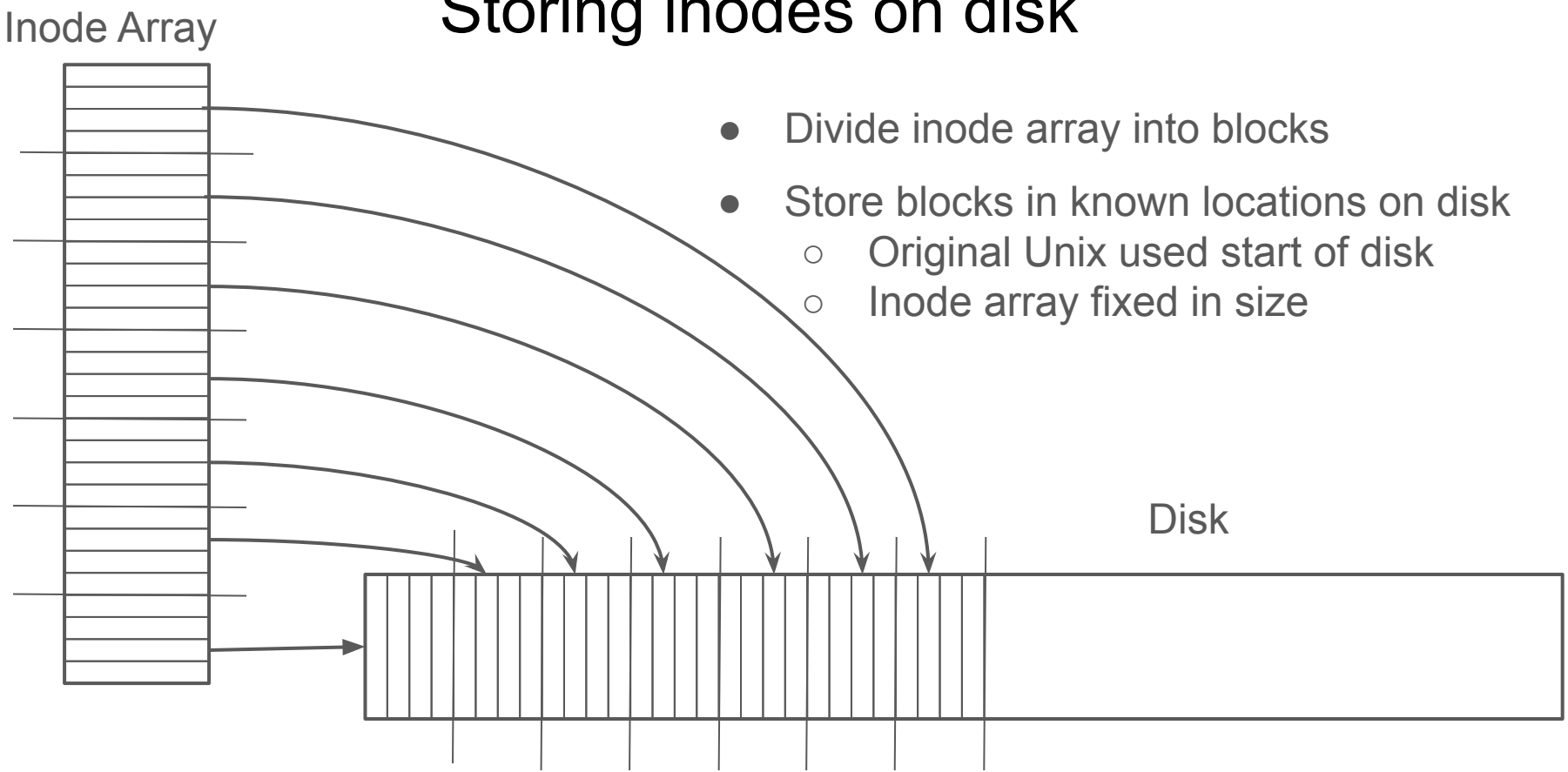
Naming inodes - i-number

Inode array



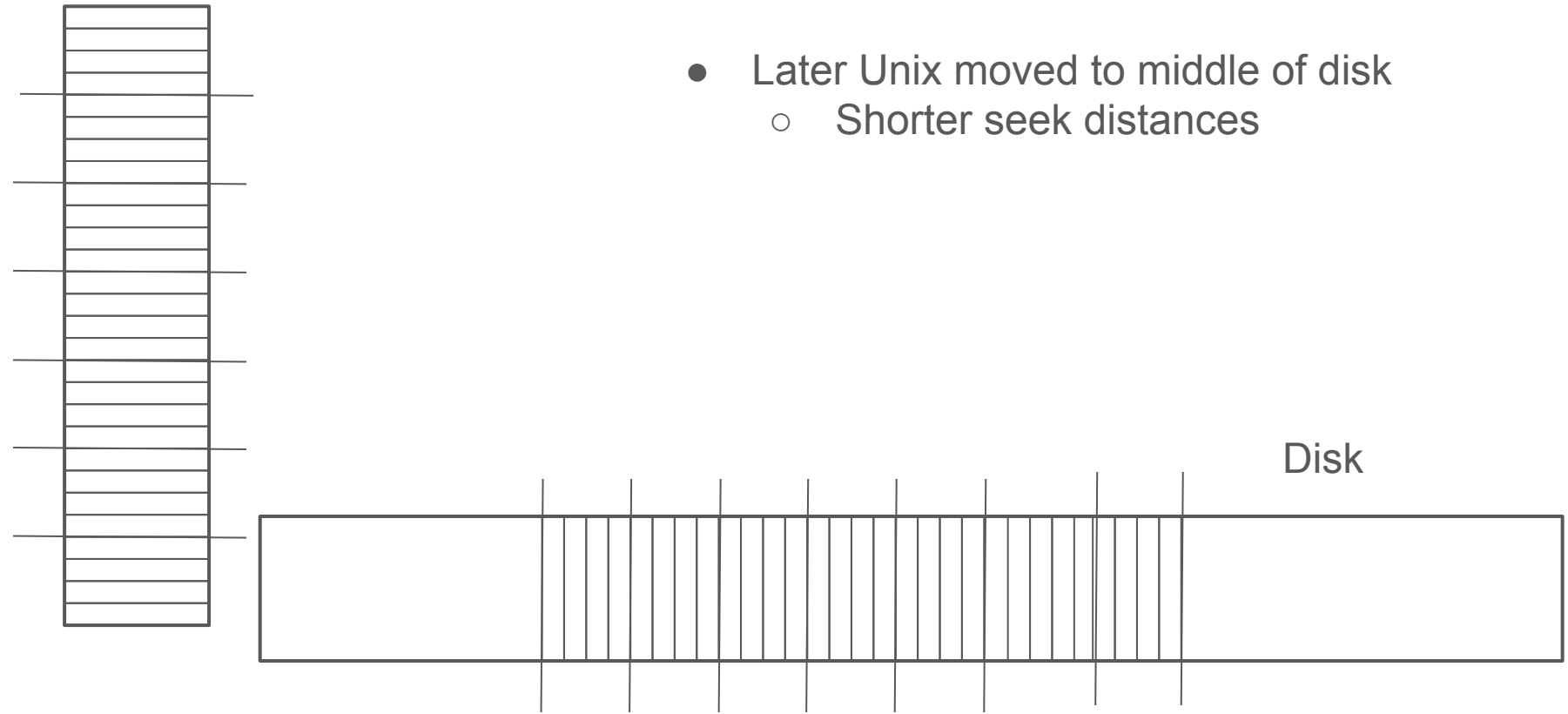
i-number - Index into inode array
Uniquely identify inode
OS uses i-number to identify file

Storing inodes on disk



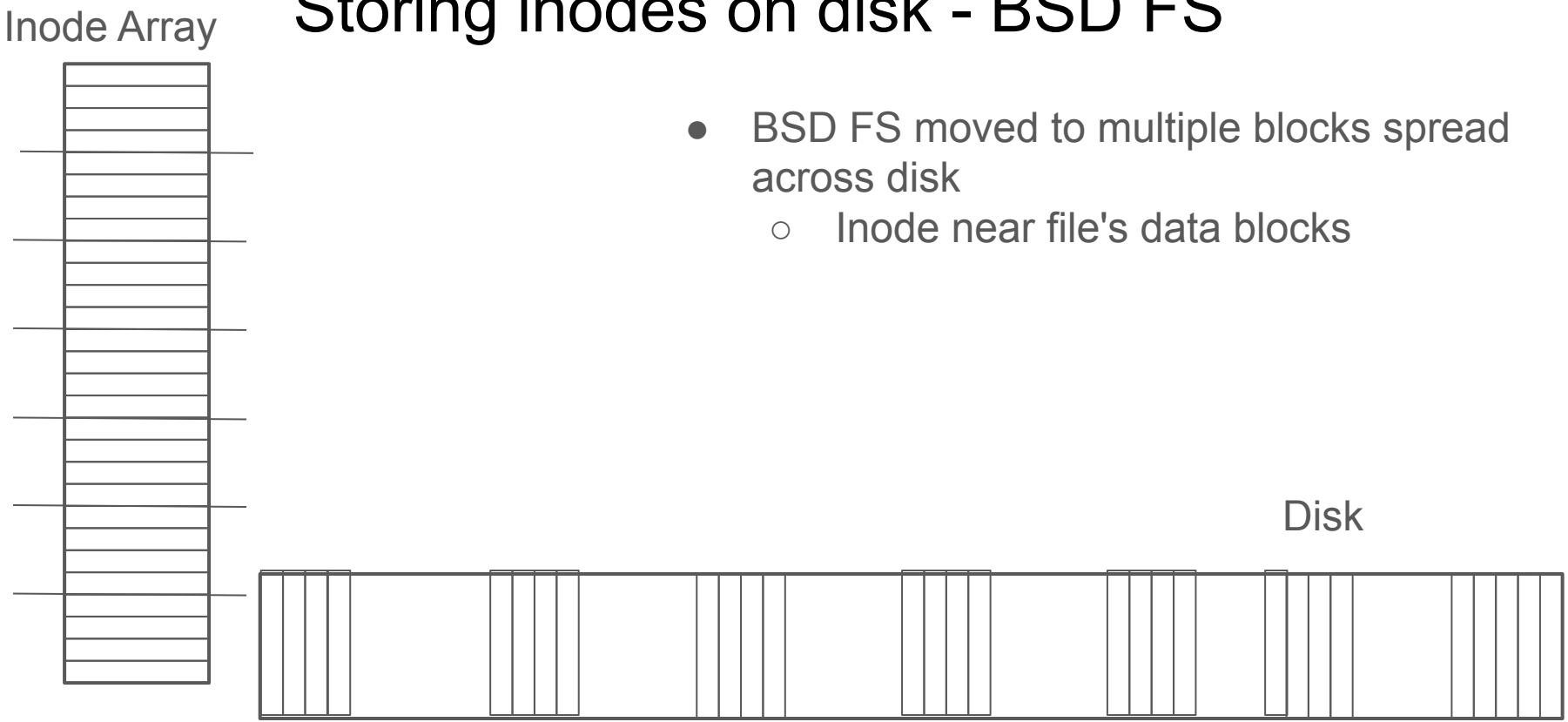
- Divide inode array into blocks
- Store blocks in known locations on disk
  - Original Unix used start of disk
  - Inode array fixed in size

Storing inodes on disk - middle inode



- Later Unix moved to middle of disk
  - Shorter seek distances

Storing inodes on disk - BSD FS



- BSD FS moved to multiple blocks spread across disk
  - Inode near file's data blocks

Inodes and open files

- When a file is open, its inode is kept in main
  - Read/write operations can find the blocks on disk
- When the file is closed, the inode is written back to disk
  - Even read access modifies inode access time fields

Need to be able to go from name (e.g. `/a/b/c.cc`) to i-number/inode

File naming

- Users want to use text names to refer to files
- Special disk structures called **directories**
  - Map names to i-numbers
- Most modern file system support hierarchical names
  - Directories have names that reference other directories

Early approaches to directory management

- A single directory for the entire disk
  - Many early personal computers worked this way.
- A single directory for each user (e.g. TOPS-10):
  - Avoids problems between users
- Would like more directories to help organize information

Hierarchical directory structures

- Modern systems support hierarchical directory structures
  - Linux, Windows, MacOS all do
- Linux and MacOS
  - Names have slashes separating the levels of the tree (e.g. `/usr/class/cs111`)
- Windows
  - Names have backslashes separating the levels of the tree (e.g. `\usr\class\cs111`)

UNIX/Linux Directories

- Stored just like normal files
  - Inode has type == directory
  - Only the operating system can write (read too these days)
- Collection of pairs: <name, i-number>
  - struct with 14 bytes of name and number
  - No particular order
- i-numbers can refer to files or child directories
  - Hierarchical tree structure
- Root directory (/): i-number 1
- Similar but slightly different on Windows

assign0 directory

Makefile	1443
movietest.cc	96
---	0
samples	228
movie.h	1441
---	0
questions.txt	656
movie.cc	780
buggy.c	6312
…	

open("/a/b/c")

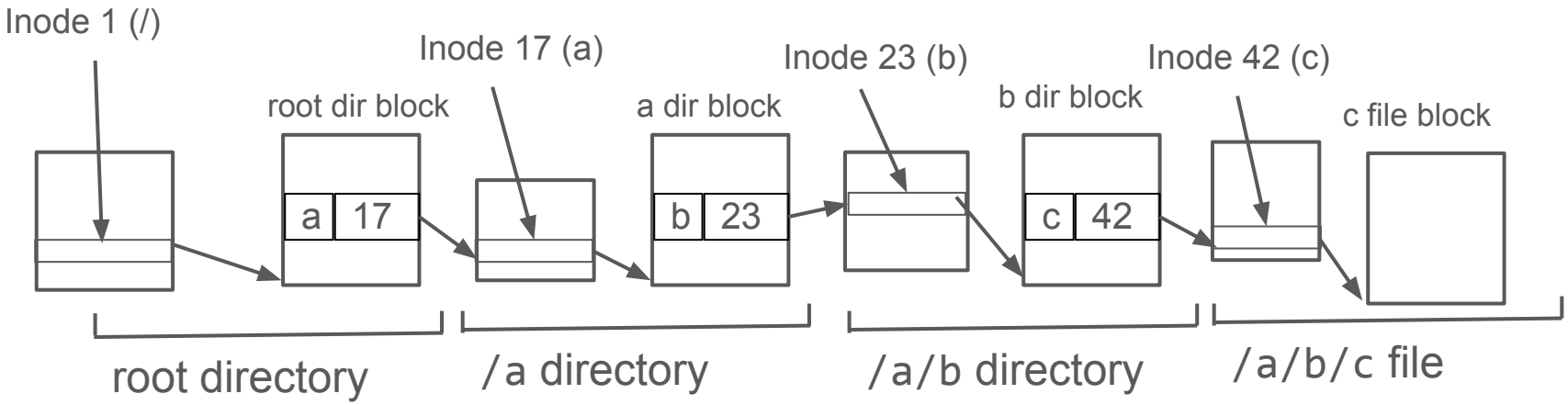
Need `map("/a/b/c")` ⇒ i-number (don't have)

Break pathname into components: "/", "a", "b", "c"

1. "/" - Read inode for i-number == 1 (well know root directory i-number)
2. "a" - Read blocks of inode 1 looking for `"a"`, return corresponding i-number
3. "a" - Read inode for `"a"` i-number
4. "b" - Read blocks of inode (`"a"` i-number) looking for `"b"`, return i-number
5. "b" - Read inode `"b"` i-number
6. "c" - Read blocks of inode (`"b"` i-number) looking for `"c"`, return i-number
7. "c" - Read inode `"c"` i-number, complete `open` system call

`fd = open("/a/b/c"); read(fd,buf,32);`

Break pathname into components: "/", "a", "b", "c"



Working directories

- Cumbersome to specify the full path name for all files
- **Working directory:** OS stores one directory (name/i-number) per process
  - `pwd` command on Linux and MacOS prints name
- Pathame doesn't start with "/": Look up starts with the working directory
- Pathname starting with "/" : Look up starts with the root directory (1)

Hard links (Linux/Unix)

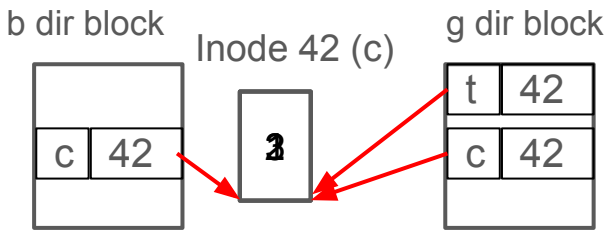
- A directory entry with a name and i-number pair is referred as a **link**
- Can have more than one directory entries with same i-number
  - Two or more names refer to a single file
- Reference counts in inodes track links referring to a file
  - Files are deleted when the link goes away
- Referred to as **hard links** now

Hard link example

We have `/a/b/c/` and current working directory is `/g`

```
ln /a/b/c c
```

```
ln /a/b/c t
```



Hard links you may have seen

- `rm` command - removes a hard link
- Every directory has two entries:
  - `"."` - A hard link to the directory itself
    - `./movietest N` command from assignment 0
  - `".."` - A hard link to the directory's parent
    - `cd ../..`

Limitations of hard links

- Can not create hard links to directories
  - Need to prevent circularities
- Can not create hard links to files in different file systems/disks
  - I-number only meaningful within a file system
- BSD Unix added **symbolic links**

Symbolic links

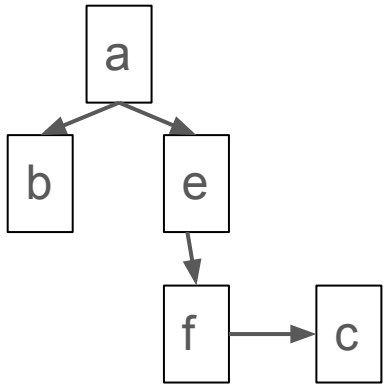
- A file whose contents are another pathname
  - Inode has type == symbolic link
- During file lookup, a symbolic link is prepend to remaining path, continue lookup
  - If symbolic link starts with "/", resume lookup in root directory
  - Otherwise, continue lookup in same directory as the one containing the symbolic link

```
cd /a;
ln -s e/f b
```

```
cat /a/b/c
```

1) read `/a` find symbolic link b - symlink "e/f"
2) continue lookup `e/f/c`

Same as `cat /a/e/f/c`



Symbolic links

- Works across file systems and to directories
- Some symbolic links problems not present in hard links:
  - Loops possible
  - Symbolic links to non-existent directories and files
    - Dangling links easy to create (sometimes on purpose)