

# CPSC313: Computer Hardware and Operating Systems

Unit 4: File Systems  
Naming

# Admin

- Quiz 4 is next week
  - Read the information, do the practice, reserve a time!
- Tutorial 9 is also next week
- Reserve your final exam time if you haven't already done so
- The pre-reading for Monday's class is longer than usual, so don't leave it to the last minute.
- Labs
  - Lab 8 is due Sunday.
  - Lab 10 (the last lab!) will be released at 17:00 today.

# Today

- Learning Outcomes
  - Explain how a file system translates a pathname to an inode number (how it finds the associated item on disk).
  - Given the contents of a disk, identify the sequence of disk blocks accessed to resolve a pathname.
- Reading
  - 10.6, 10.7

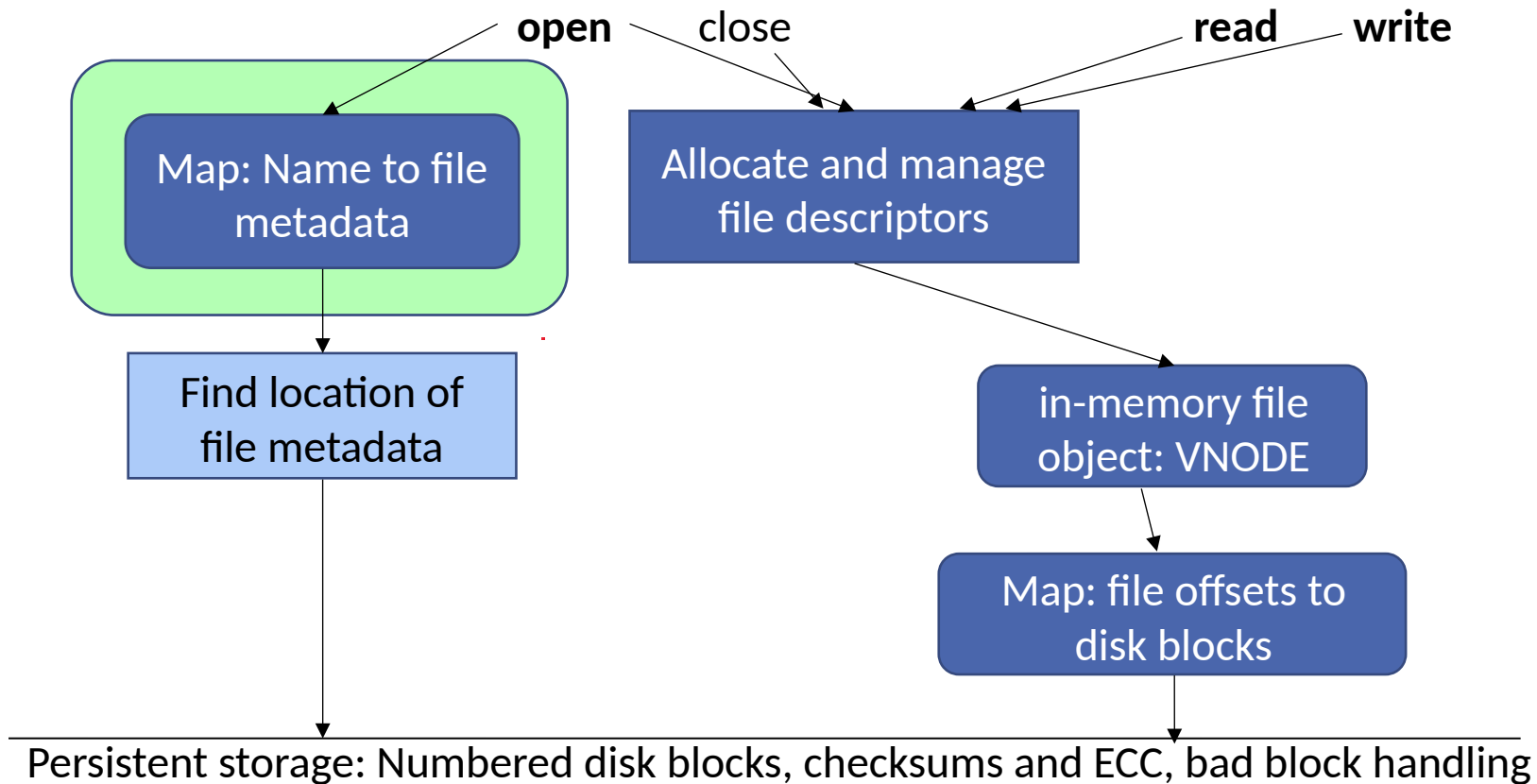
# Today

- We will use the term “directory” to refer to what you might call a “folder”. Both terms are ok, but saying “directory or folder” gets tiresome!
- **We’ll go fast** today, but you’ll explore these ideas in in-class exercises & Lab 9!

Posix API: hierarchical name space, byte-streams, open, close, read, write

---

We are  
here!

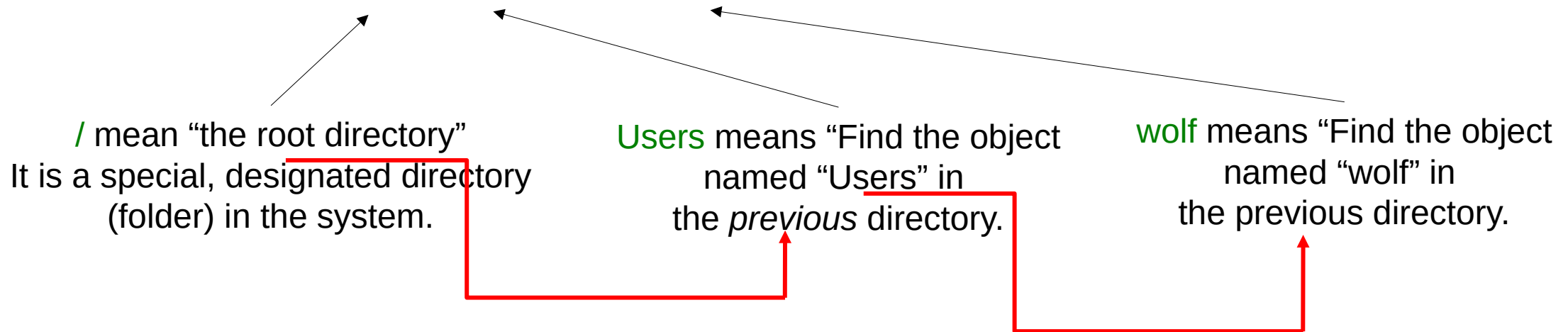


# What problem do we have to solve?

- Given a name like `/Users/wolf/midterm-answers.org`, we need to:
  - Figure out its inode number and thereafter,
  - Find and read its inode and thereafter,
  - Find all its blocks/data.

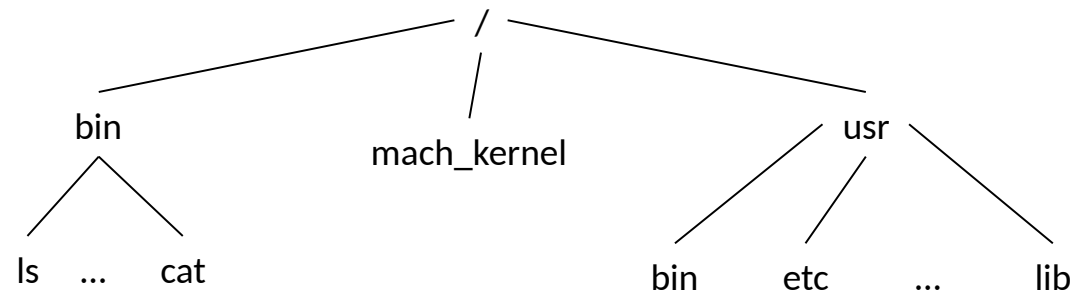
# What problem do we have to solve?

- What does: `/Users/wolf/midterm-answers.org` mean?



# Hierarchical Naming

- Generalized tree structure
  - Pros:
    - Makes names local to a directory (you and I can each have our own “lab8.c”).
    - Reuses whatever file implementation we have for directories.
  - Cons:
    - Lookup is an iterative (and potentially expensive) operation.



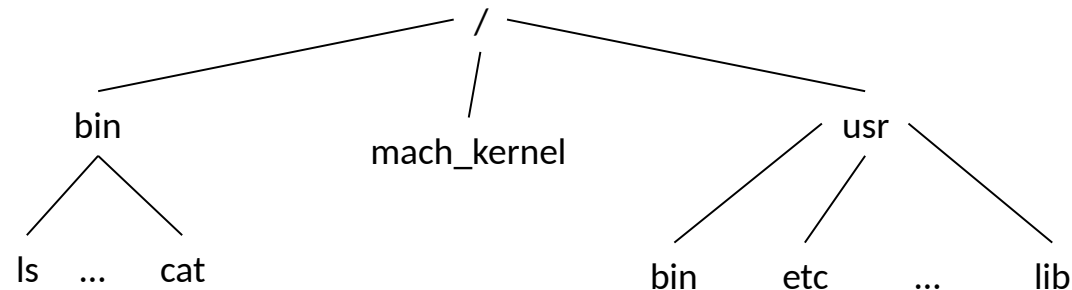


# Directory Representation

- Directories are regular files with a special format.
- A field in a file's inode indicates that the file is of type directory.
- A **directory entry** is a **pair** of **name** and associated **inode number**
  - User programs can read directories like files
  - Only OS can write directories (users could corrupt the directory structure!)

# Directories and Pathnames:

- Directories can contain names of other directories: **subdirectories**
- An **absolute pathname** is a sequence of directory-entry names, starting at the root ("/").
- A **relative pathname** is a sequence of directory-entry names, starting anywhere in the file system.
- Each (valid) name within a directory is unique.



# Traditional Directory Implementation

- A directory is a file with structured contents (directory entries: `struct dirent`).
  - `Name` (required)
  - `Inode` number (required)
  - `Type` (in some file systems' `dirent`)
- The root directory has a known, fixed inode number (`inumber`).

# Traditional Directory Implementation

- In POSIX (i.e., UNIX, Linux), every directory has two special entries:
  - The “.” entry refers to the directory itself.
  - The “..” entry refers to the parent directory.
  - This is how the file system implements paths such as ../lab9 and ./myprog

# The Directory Entry Structure

- The details of `struct dirent` are specific to a file system.
- For example, on MacOS (OSX), there are two possibilities.

If inode numbers are 32 bits:

```
struct dirent {
    ino_t d_ino;
    __uint16_t d_reclen;
    __uint8_t d_type;
    __uint8_t d_namlen;
    char d_name[255];
};
```

If inode numbers are 64 bits:

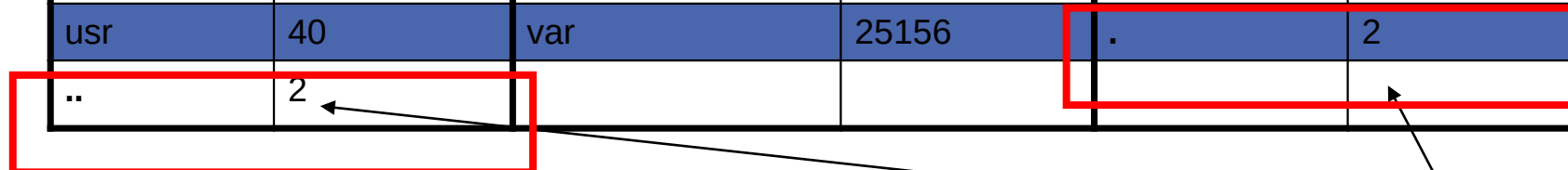
```
struct dirent {
    __uint64_t d_ino;
    __uint16_t d_reclen;
    __uint16_t d_namlen;
    __uint8_t d_type;
    char d_name[1024];
};
```

As far as C is concerned, these structs have a specific, fixed size.  
On the disk, their size is specified by `d_reclen`!

# The Root Directory

- This is the contents of the “/” directory on Margo's (old) machine (name and inode number only).

Name	inumber	Name	inumber	Name	inumber
Applications	113	Desktop Folder	844727	Developer	844731
Documents	937803	Library	213	Marketocracy	937813
Network	84416	System	37	Updaters	937816
Users	38892	Volumes	26447	bin	24377
cdrom	937840	cores	84418	dev	296
etc	25116	home	5	mach_kernel	552433
net	3	opt	937844	private	214
sbin	4512	sw	1024168	tmp	25155
usr	40	var	25156	.	2
..	2				



The root directory is the only time you will see the same inode number for . and ..

# Walking a Directory Path

- Given a path */C1/C2/C3 ...*
  - Start at the root directory (which is a directory at a fixed, known inumber) if the path begins with / or the current working directory otherwise.
    - 1 Let *inum* = starting inumber; current component = next component (*C1*, here).
    - 2 Read the contents of the object whose inumber is *inum* (this is a directory).
    - 3 Find the entry with the name equal to the current component.
    - 4 Find the associated inumber.

# Walking a Directory Path

- Given a path `/C1/C2/C3 ...`
  - Continuing...
    - 4 Find the associated inumber.
    - 5 Read the inode for that inumber
      - If this is the last component of the path, we're done.
      - If it is not a directory and we have more components, then this is a bad pathname.
      - If it is a directory, set `inum` to the inumber; set current component to next part of path and iterate back to step 2.



# Directory Example (the artist's rendition)

Assume:

- Inodes are in blocks 100 to 199.
- Blocks contain 8 inodes:
  - Block 100 has inodes 0-7.
  - Block 101 contains 8-15.
  - Etc.

	Disk block number	Contents			
inodes	100	200			
	101	202	203		
	102	204	205		
	...				
Data Blocks	200	., 2	.., 2	bin, 8	
		usr, 16	boot, 35	kadb, 27	
	201	There is	some	text or	stuff
		in	this file		
	202	., 8	.., 2	ls, 91	
		csh, 105			
	203	., 9	.., 16	libc.a, 55	
		font, 77			
	204	., 16	.., 2	lib, 9	
		share, 52	ucb, 15	old, 66	
					17

# Directory Example (the artist's rendition)

Assume:

- Inodes are in blocks 100 to 199.
- Blocks contain 8 inodes:
  - Block 100 has inodes 0-7.
  - Block 101 contains 8-15.
  - Etc.

Each horizontal 'chunk' represents a disk block.

	Disk block number	Contents			
inodes	100	200			
	101	202	203		
	102	204	205		
	...				
Data Blocks	200	., 2	.., 2	bin, 8	
		usr, 16	boot, 35	kadb, 27	
	201	There is	some	text or	stuff
		in	this file		
	202	., 8	.., 2	ls, 91	
		csh, 105			
	203	., 9	.., 16	libc.a, 55	
		font, 77			
	204	., 16	.., 2	lib, 9	
		share, 52	ucb, 15	old, 66	

# Directory Example (the artist's rendition)

Assume:

- Inodes are in blocks 100 to 199.
- Blocks contain 8 inodes:
  - Block 100 has inodes 0-7.
  - Block 101 contains 8-15.
  - Etc.

These blocks all contain inodes.  
The first one (100) contains inodes 0-7; the next one (101) contains 8-15, etc.

	Disk block number	Contents			
inodes	100			200	
	101	202	203		
	102	204	205		
	...				
Data Blocks	200	., 2	.., 2	bin, 8	
		usr, 16	boot, 35	kadb, 27	
	201	There is	some	text or	stuff
		in	this file		
	202	., 8	.., 2	ls, 91	
		csh, 105			
	203	., 9	.., 16	libc.a, 55	
		font, 77			
	204	., 16	.., 2	lib, 9	
		share, 52	ucb, 15	old, 66	

# Directory Example (the artist's rendition)

Assume:

- Inodes are in blocks 100 to 199.
- Blocks contain 8 inodes:
  - Block 100 has inodes 0-7.
  - Block 101 contains 8-15.
  - Etc.

	Disk block number	Contents			
inodes	100	200			
	101	202	203		
	102	204	205		
	...				
Data Blocks	200	., 2	.., 2	bin, 8	
		usr, 16	boot, 35	kadb, 27	
	201	There is	some	text or	stuff
		in	this file		
	202	., 8	.., 2	ls, 91	
		csh, 105			
	203	., 9	.., 16	libc.a, 55	
		font, 77			
	204	., 16	.., 2	lib, 9	
		share, 52	ucb, 15	old, 66	

Each entry in an **inode block** represents an inode; the number is the disk address of the (first) block of the object the inode describes.

# Directory Example (the artist's rendition)

Assume:

- Inodes are in blocks 100 to 199.
- Blocks contain 8 inodes:
  - Block 100 has inodes 0-7.
  - Block 101 contains 8-15.
  - Etc.

	Disk block number	Contents			
inodes	100	200			
	101	202 203			
	102	204 205			
	...				
Data Blocks	200	., 2	.., 2	bin, 8	
		usr, 16	boot, 35	kadb, 27	
	201	There is	some	text or	stuff
		in	this file		
	202	., 8	.., 2	ls, 91	
		csh, 105			
	203	., 9	.., 16	libc.a, 55	
		font, 77			
	204	., 16	.., 2	lib, 9	
		share, 52	ucb, 15	old, 66	

Each data block contains the contents of a file or a directory.

File

Directory

# Directory Example (the artist's rendition)

Assume:

- Inodes are in blocks 100 to 199.
- Blocks contain 8 inodes:
  - Block 100 has inodes 0-7.
  - Block 101 contains 8-15.
  - Etc.

	Disk block number	Contents			
inodes	100	200			
	101	202 203			
	102	204 205			
	...				
Data Blocks	200	., 2	., 2	bin, 8	
		usr, 16	boot, 35	kadb, 27	
	201	There is	some	text or	stuff
		in	this file		
	202	., 8	., 2	ls, 91	
		csh, 105			
	203	., 9	.., 16	libc.a, 55	
		font, 77			
	204	., 16	.., 2	lib, 9	
		share, 52	ucb, 15	old, 66	

Each box in a directory block is a directory entry (a name/inode number pair)

# Directory Example (the artist's rendition)

Assume:

- Inodes are in blocks 100 to 199.
- Blocks contain 8 inodes:
  - Block 100 has inodes 0-7.
  - Block 101 contains 8-15.
  - Etc.

Exercise: list all the blocks, in order, that you need to read to open */usr/lib/libc.a*.

	Disk block number	Contents			
inodes	100	200			
	101	202	203		
	102	204	205		
	...				
Data Blocks	200	., 2	.., 2	bin, 8	
		usr, 16	boot, 35	kadb, 27	
	201	There is	some	text or	stuff
		in	this file		
	202	., 8	.., 2	ls, 91	
		csh, 105			
	203	., 9	.., 16	libc.a, 55	
		font, 77			
	204	., 16	.., 2	lib, 9	
		share, 52	ucb, 15	old, 66	23

# Directory Example (1): Find the Root Directory

List all the blocks, in order, that you need to read to open `/usr/lib/libc.a`

- 1 The root inode has inumber 2, so we go to the first inode block (100) and access the third inode in that block.
- 2 Our picture indicates that the inode's first block (the contents of the root directory) is in disk block 200.

	Disk block number	Contents			
inodes	100	200			
	101	202	203		
	102	204	205		
	...				
Data Blocks	200	., 2	.., 2	bin, 8	
		usr, 16	boot, 35	kadb, 27	
	201	There is	some	text or	stuff
		in	this file		
	202	., 8	.., 2	ls, 91	
		csh, 105			
	203	., 9	.., 16	libc.a, 55	
		font, 77			
	204	., 16	.., 2	lib, 9	
		share, 52	ucb, 15	old, 66	



Blocks read : 100, 200

# Directory Example (2): Read the Root Directory

List all the blocks, in order, that you need to read to open `/usr/lib/libc.a`

- 3 Reading block 200 means we are reading the actual contents of the directory.
- 4 Look for an entry with the name "usr"; it is inumber 16.

	Disk block number	Contents			
inodes	100	200			
	101	202 203			
	102	204 205			
	...				
Data Blocks	200	., 2	.., 2	bin, 8	
		usr, 16	boot, 35	kadb, 27	
	201	There is	some	text or	stuff
		in	this file		
	202	., 8	.., 2	ls, 91	
		csh, 105			
	203	., 9	.., 16	libc.a, 55	
		font, 77			
	204	., 16	.., 2	lib, 9	
		share, 52	ucb, 15	old, 66	

Blocks read : 100, 200, 102

# Directory Example (3): Read the /usr inode

List all the blocks, in order, that you need to read to open */usr/lib/libc.a*

- 5 Number 16 is the 17th inode, so it lives in the 3rd inode block.
- 6 Inode 16 is the first inode in this block.

	Disk block number	Contents			
inodes	100	200			
	101	202	203		
	102	204	205		
	...				
Data Blocks	200	., 2	.., 2	bin, 8	
		usr, 16	boot, 35	kadb, 27	
	201	There is	some	text or	stuff
		in	this file		
	202	., 8	.., 2	ls, 91	
		csh, 105			
	203	., 9	.., 16	libc.a, 55	
		font, 77			
	204	., 16	.., 2	lib, 9	
		share, 52	ucb, 15	old, 66	

Blocks read : 100, 200, 102, 204

# Directory Example (4): Read the /usr directory

List all the blocks, in order, that you need to read to open */usr/lib/libc.a*

- 7 Block 204 contains the contents of the /usr directory.
- 8 We need to find "lib." It has inumber 9.

	Disk block number	Contents			
inodes	100	200			
	101	202 203			
	102	204 205			
	...				
Data Blocks	200	., 2	.., 2	bin, 8	
		usr, 16	boot, 35	kadb, 27	
	201	There is	some	text or	stuff
		in	this file		
	202	., 8	.., 2	ls, 91	
		csh, 105			
	203	., 9	.., 16	libc.a, 55	
		font, 77			
	204	., 16	.., 2	lib, 9	
		share, 52	ucb, 15	old, 66	

Blocks read : 100, 200, 102, 204, 101

# Directory Example (5): Read the /usr/lib inode

List all the blocks, in order, that you need to read to open */usr/lib/libc.a*

- 9 Number 9 is the 10th inumber; it will be in the second block, 101
- 10 The 10th inumber is the second in this block; its contents are in block 203.

	Disk block number	Contents			
inodes	100	200			
	101	202	203		
	102	204	205		
	...				
Data Blocks	200	., 2	.., 2	bin, 8	
		usr, 16	boot, 35	kadb, 27	
	201	There is	some	text or	stuff
		in	this file		
	202	., 8	.., 2	ls, 91	
		csh, 105			
	203	., 9	.., 16	libc.a, 55	
		font, 77			
	204	., 16	.., 2	lib, 9	
		share, 52	ucb, 15	old, 66	

Blocks read : 100, 200, 102, 204, 101, 203

# Directory Example (6): Read the /usr/lib directory

List all the blocks, in order, that you need to read to open */usr/lib/libc.a*

11 Block 203 contains the contents of the /usr/lib directory

12 We want to find the entry for libc.a, which has inumber 55.

	Disk block number	Contents			
inodes	100	200			
	101	202 203			
	102	204 205			
	...				
Data Blocks	200	., 2	.., 2	bin, 8	
		usr, 16	boot, 35	kadb, 27	
	201	There is	some	text or	stuff
		in	this file		
	202	., 8	.., 2	ls, 91	
		csh, 105			
	203	., 9	.., 16	libc.a, 55	
		font, 77			
	204	., 16	.., 2	lib, 9	
		share, 52	ucb, 15	old, 66	

Blocks read : 100, 200, 102, 204, 101, 203

# Directory Example (7): Read the /usr/lib/libc.a inode

List all the blocks, in order, that you need to read to open */usr/lib/libc.a*

13 We then need to go read the inode corresponding to number 55.

Which block would it be in?

	Disk block number	Contents			
inodes	100	200			
	101	202 203			
	102	204 205			
	...				
Data Blocks	200	., 2	.., 2	bin, 8	
		usr, 16	boot, 35	kadb, 27	
	201	There is	some	text or	stuff
		in	this file		
	202	., 8	.., 2	ls, 91	
		csh, 105			
	203	., 9	.., 16	libc.a, 55	
		font, 77			
	204	., 16	.., 2	lib, 9	
		share, 52	ucb, 15	old, 66	

Blocks read : 100, 200, 102, 204, 101, 203, 106

# Directory Example (7): Read the /usr/lib/libc.a inode

List all the blocks, in order,  
that you need to read to  
open */usr/lib/libc.a*

13 Which block is inode 55 in?  
55 / 8 = inode block #6  
Inode block #6 is at disk address  
106  
55 % 8 = 7th (last) inode in the  
block

	Disk block number	Contents			
inodes	100	200			
	101	202 203			
	102	204 205			
	...				
Data Blocks	200	., 2	.., 2	bin, 8	
		usr, 16	boot, 35	kadb, 27	
	201	There is	some	text or	stuff
		in	this file		
	202	., 8	.., 2	ls, 91	
		csch, 105			
	203	., 9	.., 16	libc.a, 55	
		font, 77			
	204	., 16	.., 2	lib, 9	
		share, 52	ucb, 15	old, 66	

# Hard Links (1)

- **Creating a new file**
  - Creates a link between “its” name and inode (every file name is just a hard-link; the file’s real name is its inumber!)
- **Adding a hard link: `ln <target-file> <new-name>`**
  - Creates an extra name for the file: a new link from a new name to the inode
  - inodes are reference counted (see with `ls -l`)
    - Reference count == link count == # of directory links to a file (i.e., # of directory entries that contain its inumber)



# Hard Links (2)

- Removing a link ...
  - We think of this as removing a file, but it's not quite that.
  - `rm foo` removes the name `foo` and decrements the link count of the inode it references.
  - The inode and its file are removed when the inode's link-count goes down to 0.

# Creating hard links

- Creating an extra hard link
  - Assume file foo appears in /
  - Then we type `ln foo bar`

*Link count tells you how many names an inode has.*

After we do this, both foo and bar are equally “real” names for the file!

Before

	Disk block number	Contents							
inodes	100			200					
	101	400							
Data	200	., 2		.., 2		foo, 8			
	201								

Link count = 1

After

	Disk block number	Contents							
inodes	100			200					
	101	400							
Data	200	., 2		.., 2		foo, 8		bar, 8	
	201								34

Link count = 2

# Questions

- What is link count of a directory that ...
  - is empty?
  - contains three subdirectories?
- Can you create a hard-link between file systems?

# Questions

- What is link count of a directory that ...
  - is empty?  
2
  - contains three subdirectories?  
5
- Can you create a hard-link between file systems?
  - No: an inumber only has meaning within its home file system, just like a street address (2366) is meaningless on its own.

# Symbolic (soft) Links

- A file that contains a pathname to another file
  - `ln -s <target-pathname> <new-alias-name>`
  - Creates a new inode that stores the target pathname as a string
    - does not increment target's link count or do anything with target inode
- Resolving a symbolic link
  - When the OS encounters symbolic link, it continues pathname processing using the pathname stored in the file
  - Continuing from the current directory unless the pathname in the symbolic link is an absolute name (i.e., starts with `/`)

# Symbolic (soft) Links

- Removing a symbolic link
  - Removes the link but does nothing to the target file
- Removing the file a symbolic link points to
  - Does nothing to the link (it now points to a non-existent file). This is just like a dangling pointer in memory.
- Question
  - Can you create a soft link between two directories and/or filesystems?

# Symbolic (soft) Links

- Removing a symbolic link
  - Removes the link but does nothing to the target file
- Removing the file a symbolic link points to
  - Does nothing to the link (it now points to a non-existent file). This is just like a dangling pointer in memory.
- Question
  - Can you create a soft link between two directories and/or filesystems?  
Yes! (If you can name it, you can symbolically link it.)

# Symbolic (soft) Links

- Creating a symbolic link
  - Assume file **foo** appears in **/**
  - Then we type **ln -s foo bar**

## Differences from hard link:

1. Allocates a new inode
2. The contents of the new object contains the path of the symbolic link
3. Interpretation: Go read the file whose name appears in the new object

## Before

	Disk block number	Contents							
inodes	100			200					
	101	400							
Data	200	., 2		.., 2		foo, 8			
	201								

Link count = 1

## After

	Disk block number	Contents							
inodes	100			200					
	101	400	201						
Data	200	., 2		.., 2		foo, 8		bar, 9	
	201	"foo"							

Link count = 1



# Working Directory

- Using full pathnames all the time is annoying (and inefficient for the OS!)
- POSIX maintains a “**current working directory**” (cwd) for each *process*. The OS stores that inumber on behalf of each user process.
- If a path’s first character is “/”, the OS begins decoding the path at the root. If not it begins in the cwd.
- A search path is a list of “current working directories” that can be searched one after another to try to resolve a name (like when you type **ls** at the shell!).

# Directory tree examples

Assume **cat** is in the **bin** directory and CWD is **/bin**

Hard link examples

1. **ln cat dog**

2. **ln ls ../usr/newls**

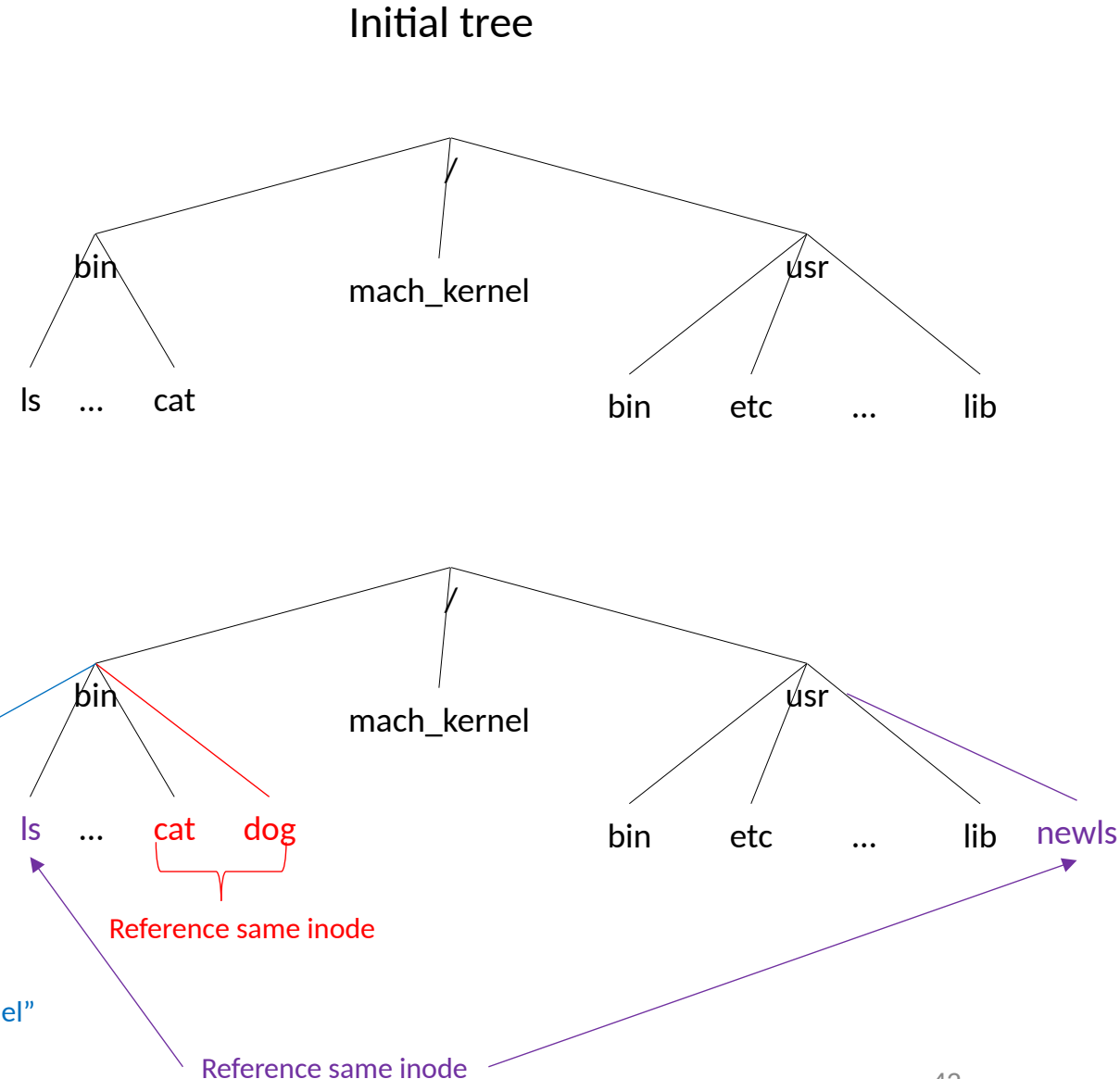
Soft link examples

3. **ln -s ../mach\_kernel thekernel**

If I am anywhere in the tree and I try to open “/bin/thekernel”, path name resolution will do the following:

- Search / for “bin”
- Search bin for “thekernel”
- <discovers the symlink>
- Search bin for “..”
- Search / for “mach\_kernel”

This object has its own inode  
This object contains “../mach\_kernel”



# Your Turn: Go Read Directories!

- It may help to remember some things you already learned like...
  - little-endian byte order,
  - four bits in a hex digit, and
  - two hex digits in a byte.
- And some things you may not have like:
  - inumber 0 is unused (represents something that was deleted)
  - hexdump dumps 16 bytes (i.e., 0x10 bytes) per line