

CS241 System Programming

File System II

Klara Nahrstedt

Lecture 21

3/10/2006



Content

- File systems basic concepts
- Summary

Directory

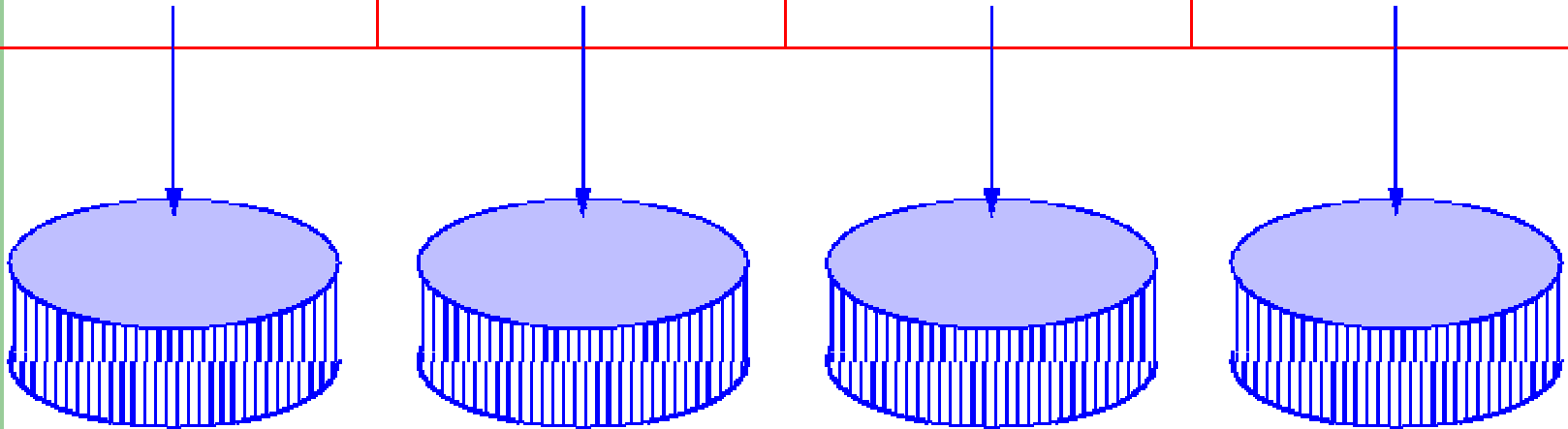
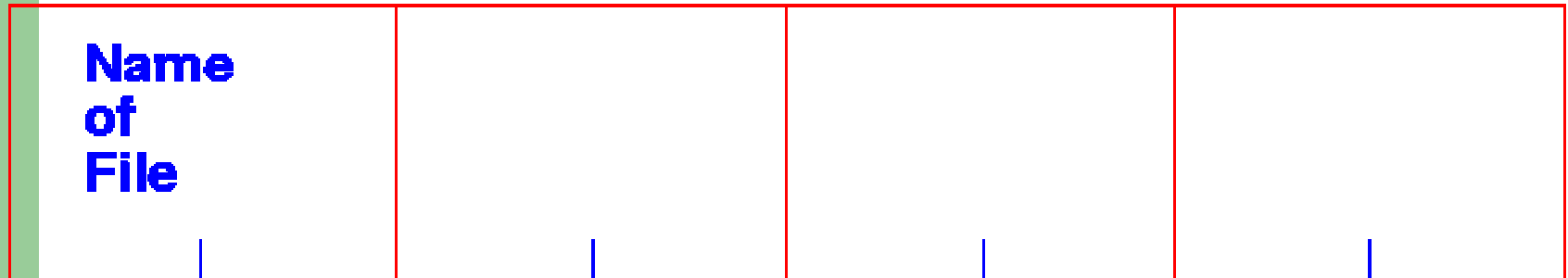
- To keep track we need
 - Directories
 - Folders
- **Directory** is a file containing correspondence between filenames and file locations

Directory Contents

- file name symbolic name
- file type indicates format of file
- location device and location
- size
- protection
- creation, access, and modification date
- owner identification

Single Level Directory

Directory



Files

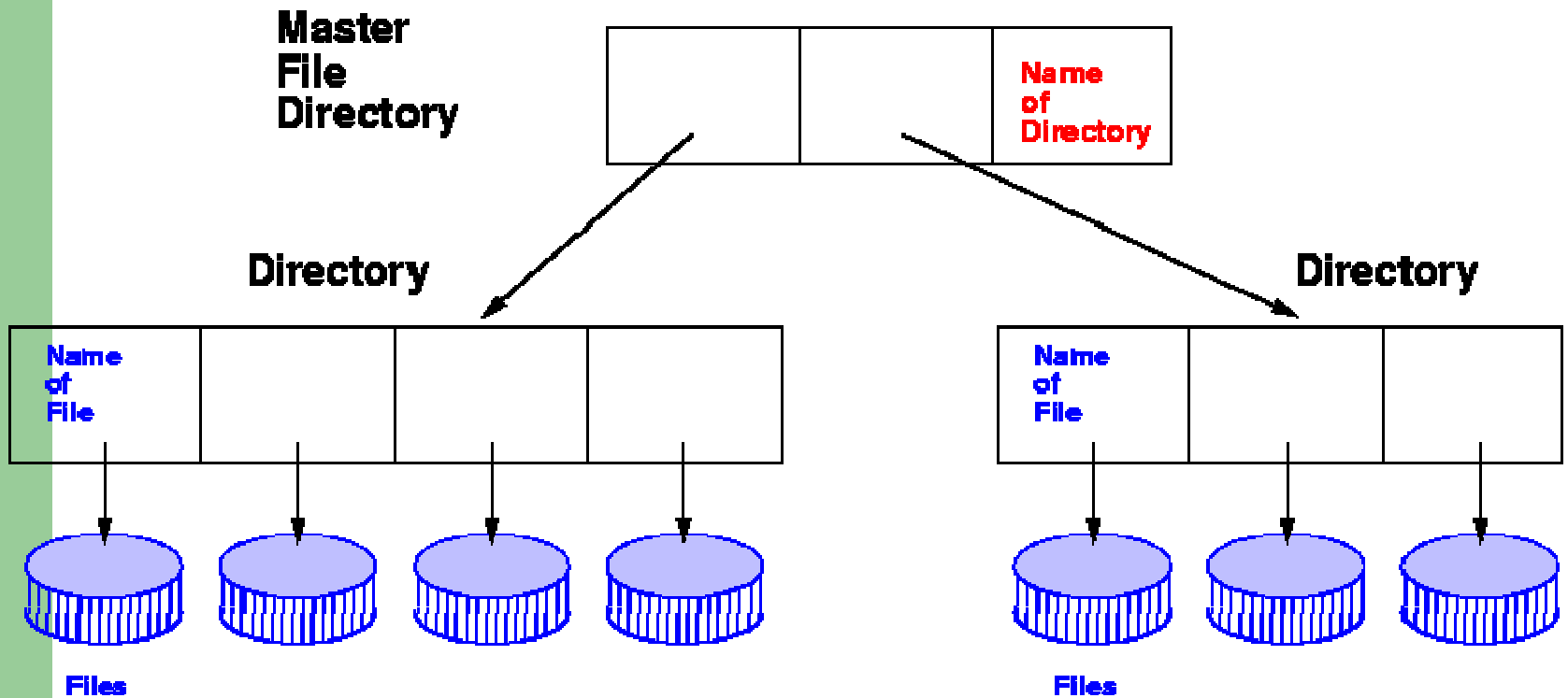
Problems With Single Level Directory

- more than one user
- large file systems
- moving files from one system to another
- name clashes
- modularity

Two-level Directory

- introduced to remove naming problems between users
- first level contains list of user directories
- second level contains user files
- system files kept in separate directory or level 1
- sharing accomplished by naming other users files

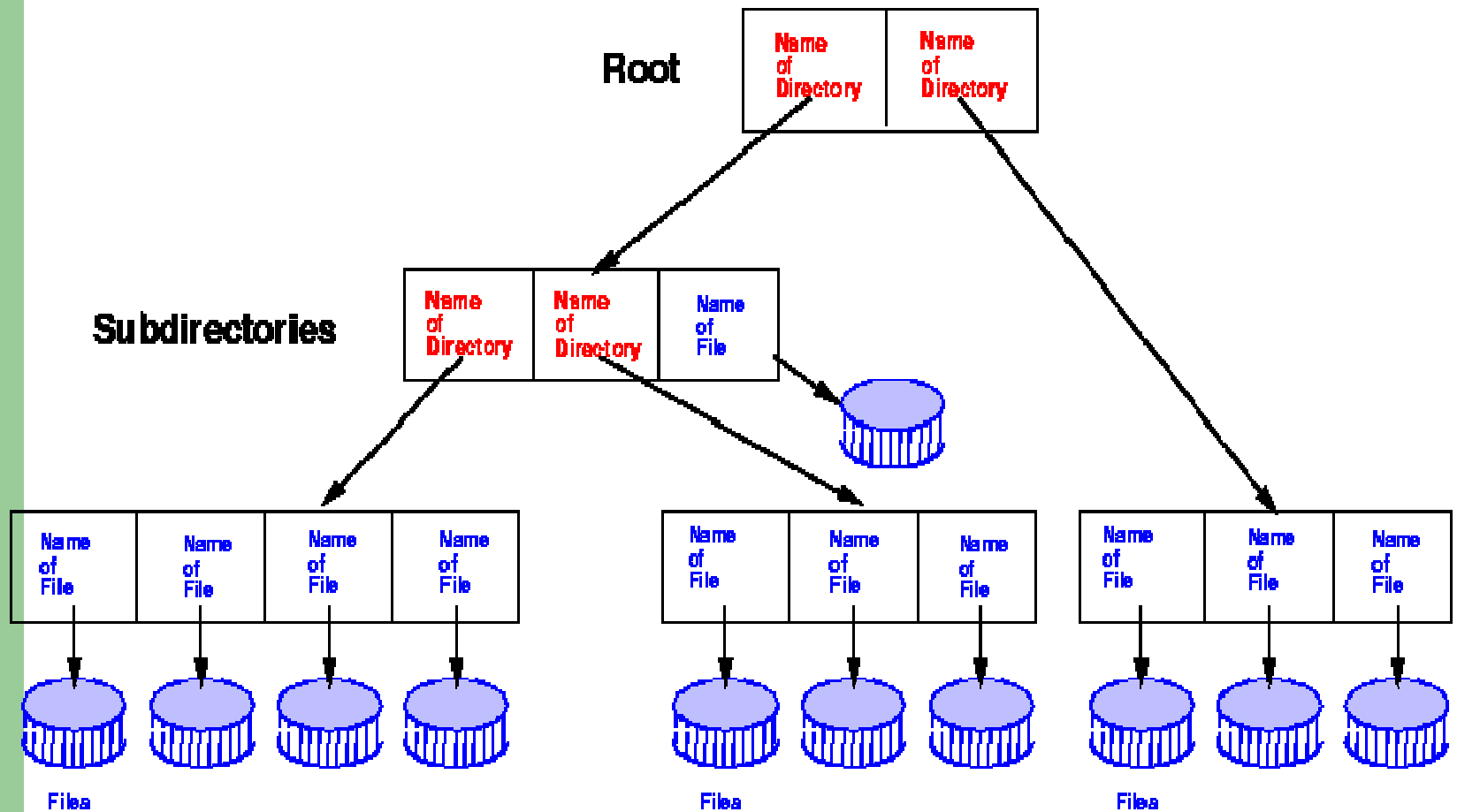
Two-level Directory



Tree Structured Directories

- arbitrary depth of directories
- leaf nodes are files
- interior nodes are directories
- path name lists nodes to traverse to find node
- use absolute paths from root
- use relative paths from current working directory pointer

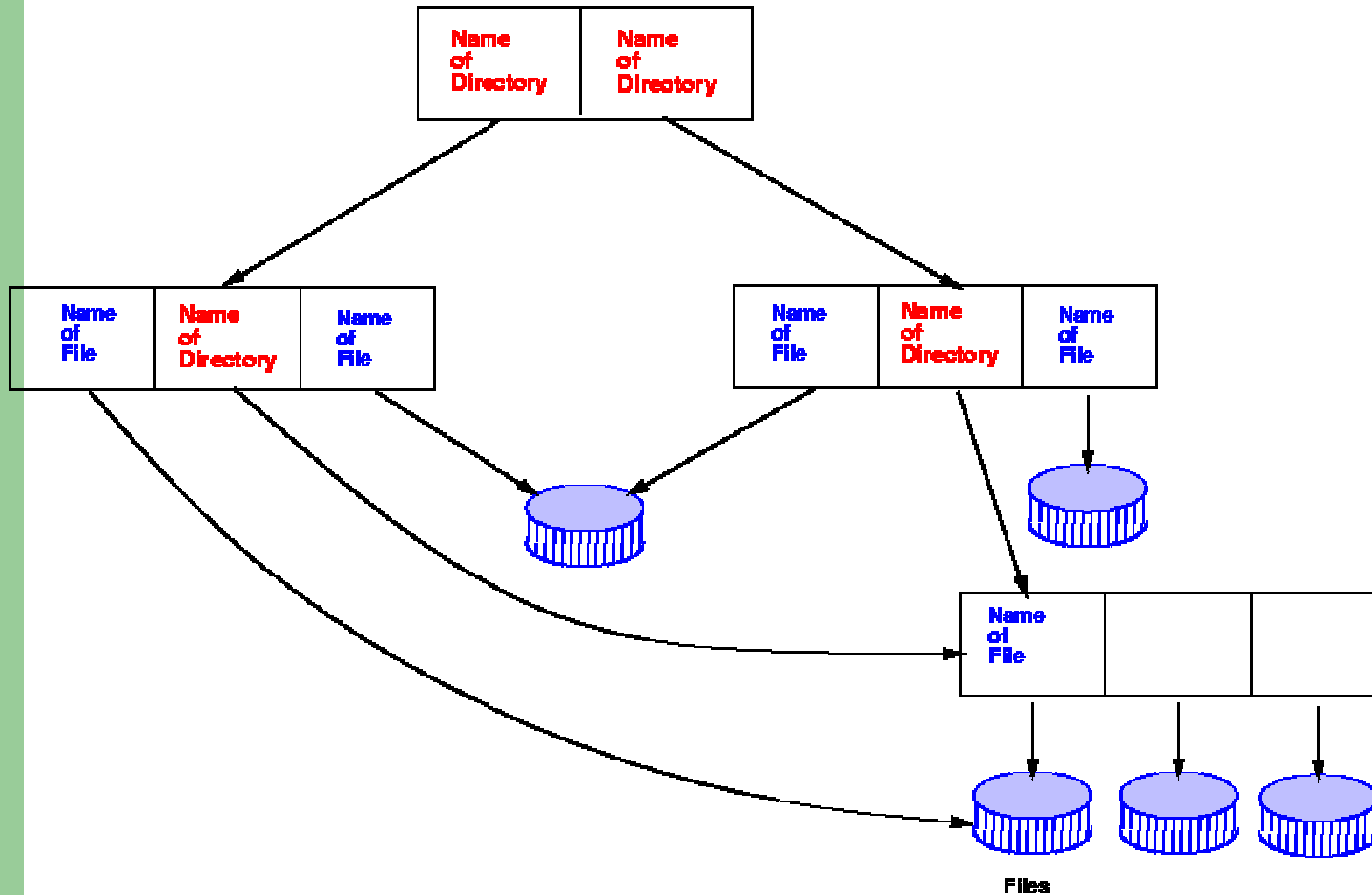
Tree Structured Directories



Acyclic Graph Structured Directories

- Acyclic graphs allow sharing
- two users can name same file
- implementation by links - use logical names of files (file system and file)
- implementation by symbolic links maps pathname into a new pathname
- duplicate paths complicate backup copies
- need reference counts for hard links

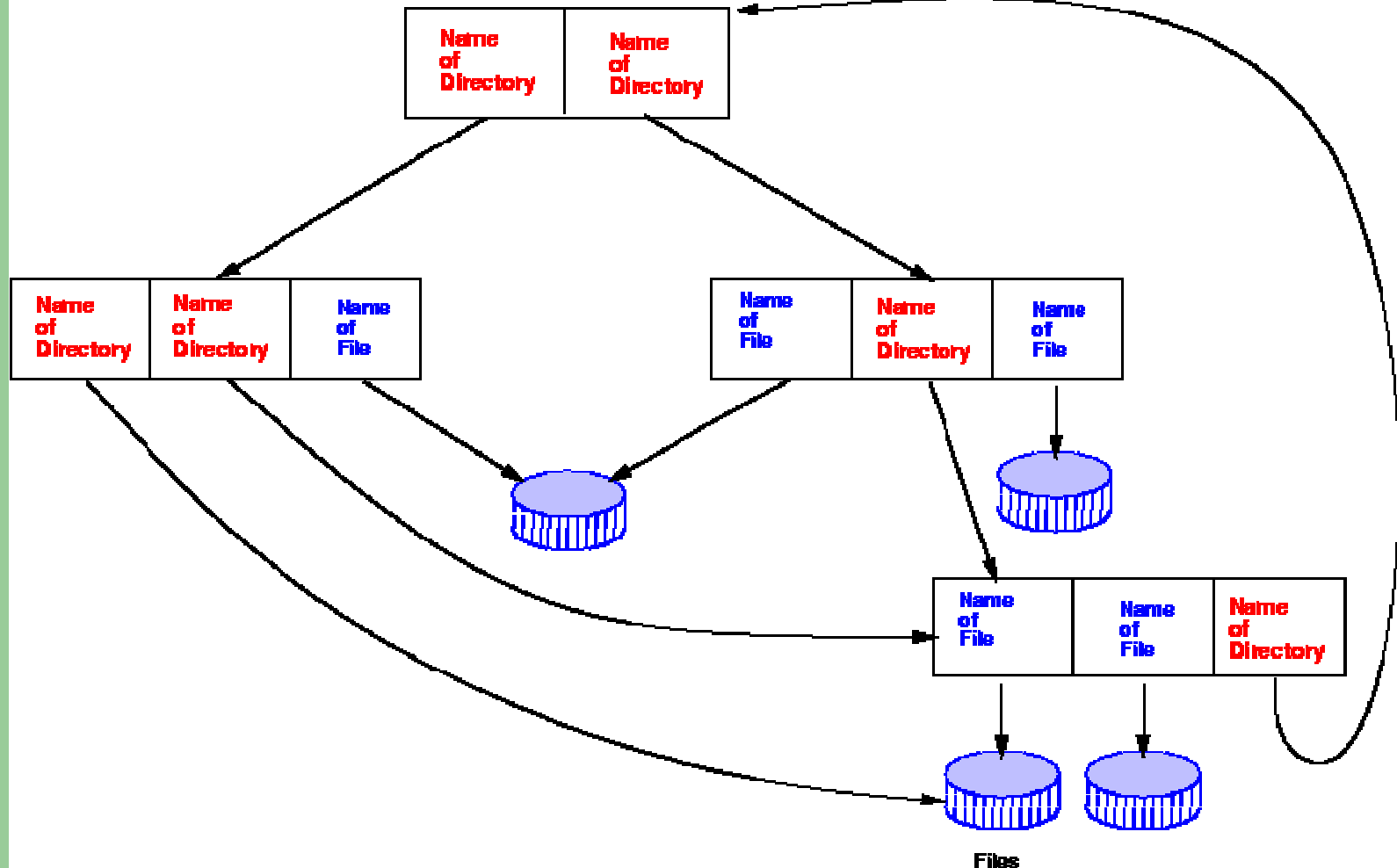
Acyclic Graph Structured Directories



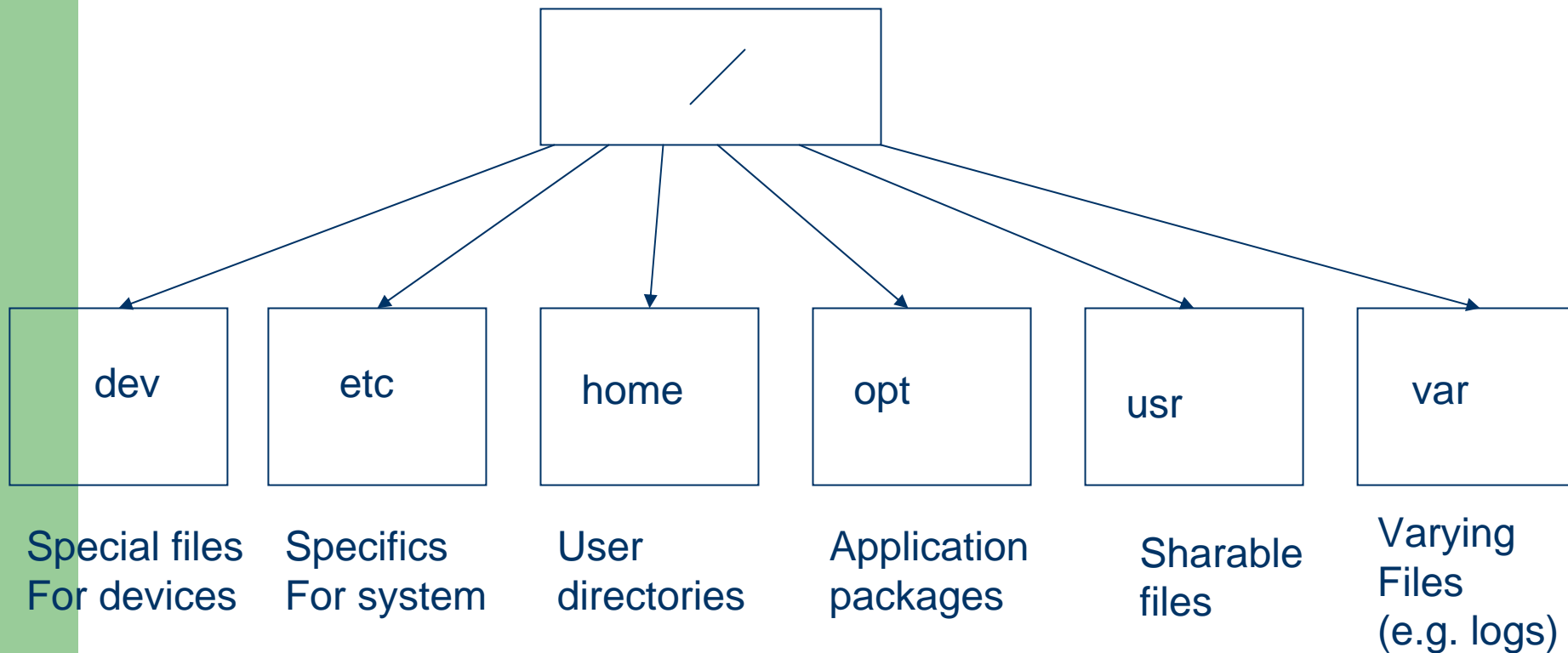
General Graph Structured Directories

- cycles
- more flexible
- more costly
- need garbage collection (circular structures)
- must prevent infinite searches

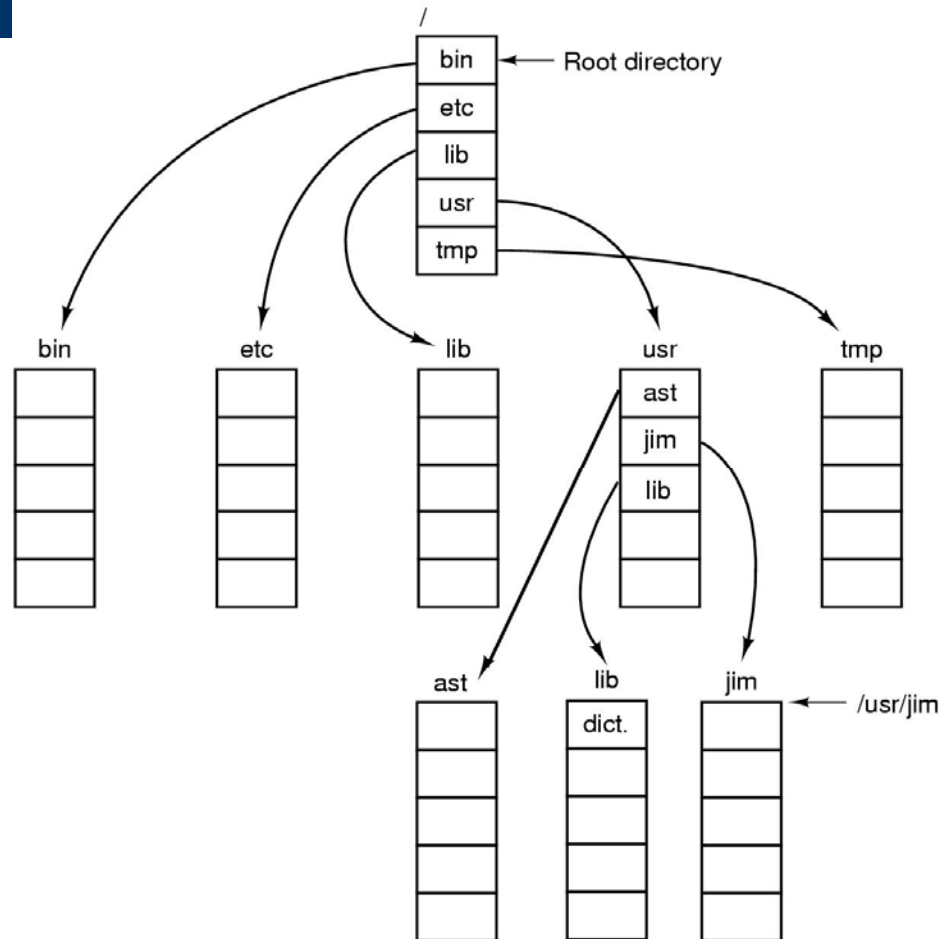
General Graph Structured Directories



UNIX File System Structure (Example)



Path Names



Absolute names:

- `/usr/ast/mailbox`
- Unique
- Start from root

Relative names:

- Working/current directory
- `mailbox`

Examples:

- `cp /usr/lib/dictionary .`
- `cp ../lib/dictionary .`

Looking up absolute path name

- Lookup absolute path name `/usr/ast/file`;
- System
 - Locate the root directory
 - Lookup string 'usr' in the root directory, get i-node of the /usr directory
 - Fetch i-node of /usr and extract disk blocks
 - Search for the string 'ast' in /usr, once the entry is found, the i-node number for /usr/ast directory is taken from it
 - Read i-node of /usr/ast and extract directory blocks
 - Lookup 'file' and find i-node of 'file'
- Conclusion: use of a relative path name is not only more convenient, but also saves a substantial amount of work for the system!!!

The UNIX V7 File System

Root directory

1	.
1	..
4	bin
7	dev
14	lib
9	etc
6	usr
8	tmp

Looking up
usr yields
i-node 6

I-node 6
is for /usr

Mode
size
times
132

I-node 6
says that
/usr is in
block 132

Block 132
is /usr
directory

6	.
1	..
19	dick
30	erik
51	jim
26	ast
45	bal

/usr/ast
is i-node
26

I-node 26
is for
/usr/ast

Mode
size
times
406

I-node 26
says that
/usr/ast is in
block 406

Block 406
is /usr/ast
directory

26	.
6	..
64	grants
92	books
60	mbox
81	minix
17	src

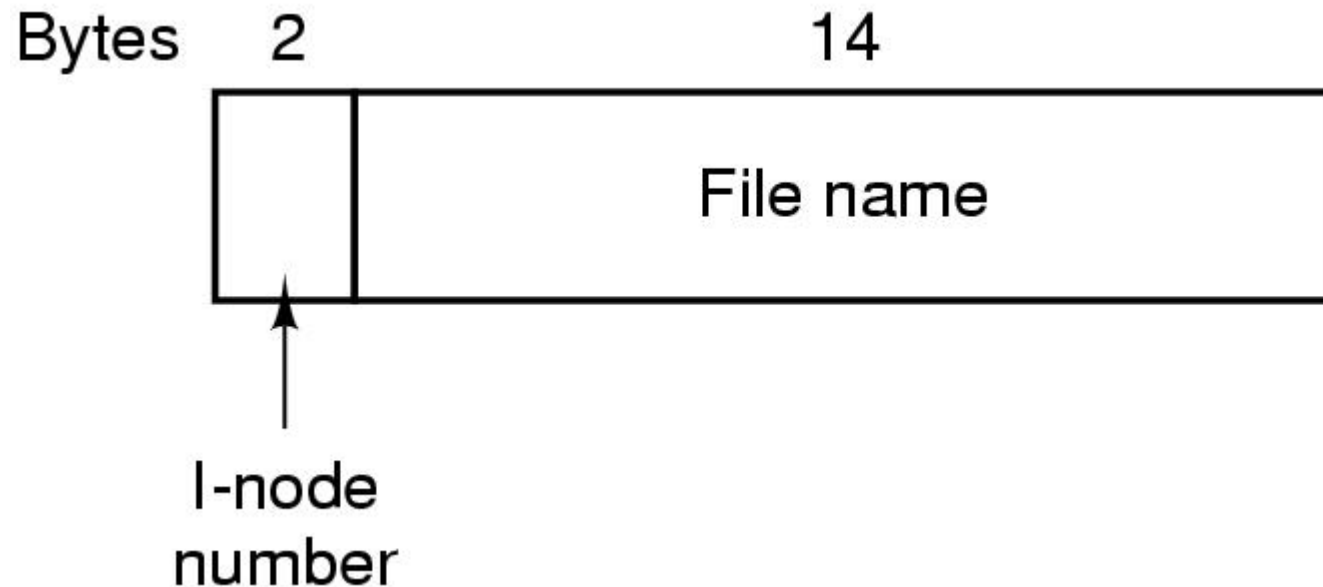
/usr/ast/mbox
is i-node
60

The steps in looking up /usr/ast/mbox

Directory Operations

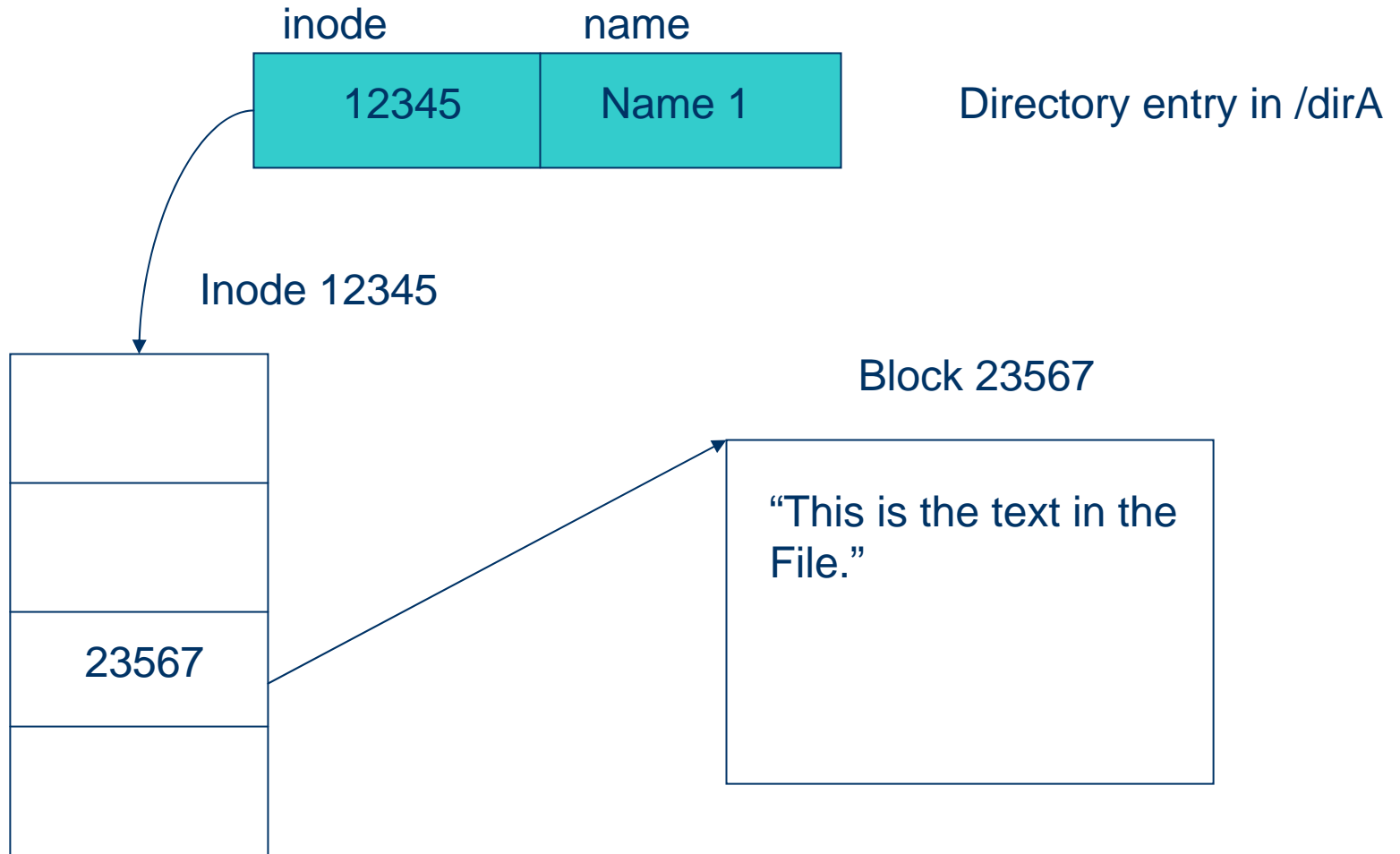
1. **Create**
 - Directory creation
 - Empty except . And ..
2. **Delete**
 - Only empty directory can be deleted in some systems
 - Directory with . And .. Is considered empty
3. **Opendir**
 - This program opens the directory to list all files in the directory
 - Before a directory can be read, it must be opened, analogous to opening and reading a file
4. **Closedir**
 - When a directory has been read, it should be closed to free up internal table space
5. **Readdir**
 - Returns the next entry in the open directory
6. **Rename**
 - Rename directory just like any other file
7. **Link**
 - Linking is a technique that allows a file to appear in more than one directory
 - Creates a link between an existing file and the name specified by the path
 - Link of this kind is called '**hard link**'
 - Each file which has links pointed to it, includes a **reference counter** in its i-node to keep track of the number of directory entries containing the file
8. **Unlink**
 - If the file being unlinked is only present in one directory, it is removed from the file system
 - If the file is present in multiple directories, only the path name specified is removed.

UNIX Directory Entry



A UNIX V7 directory entry

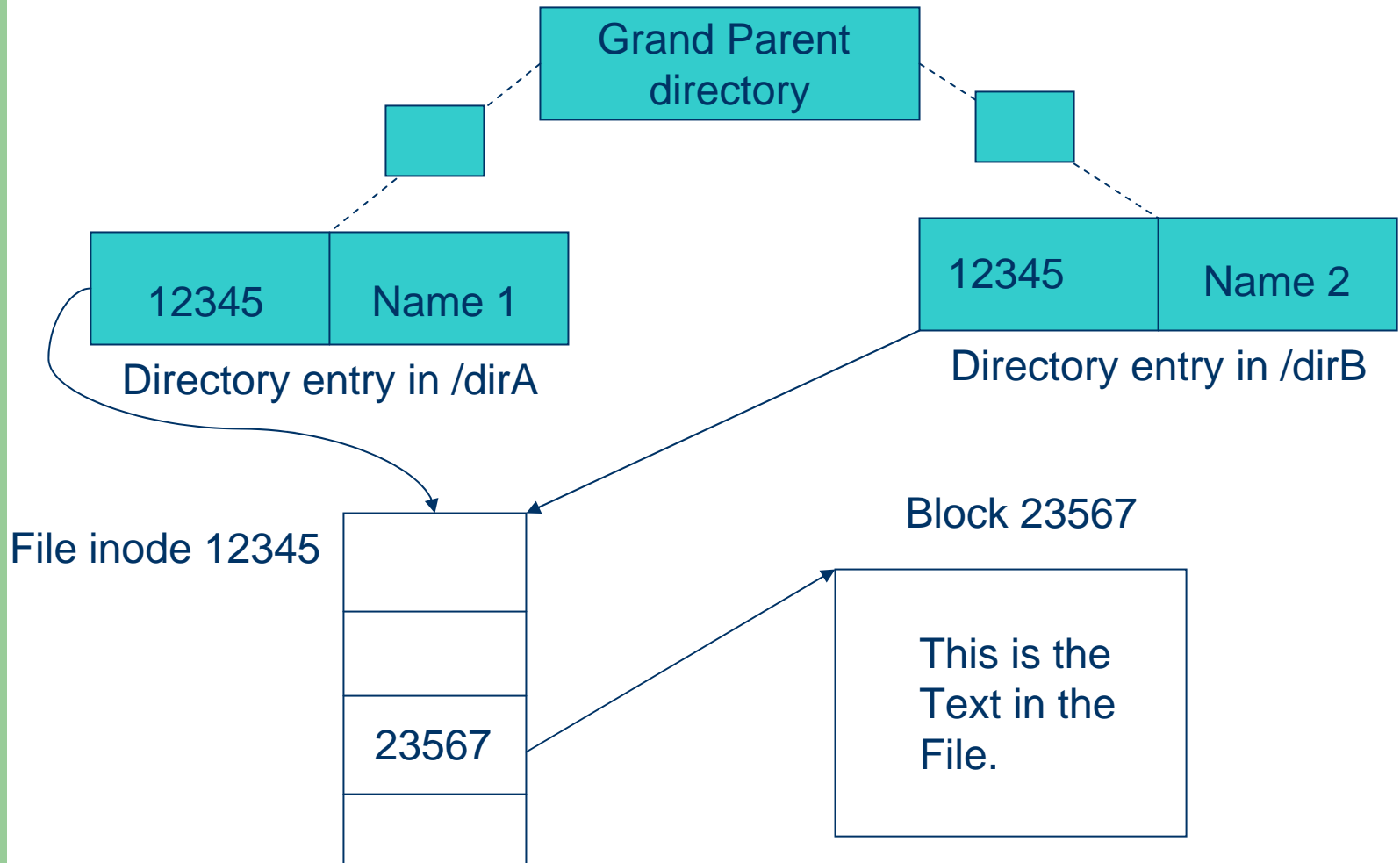
UNIX Directory Example



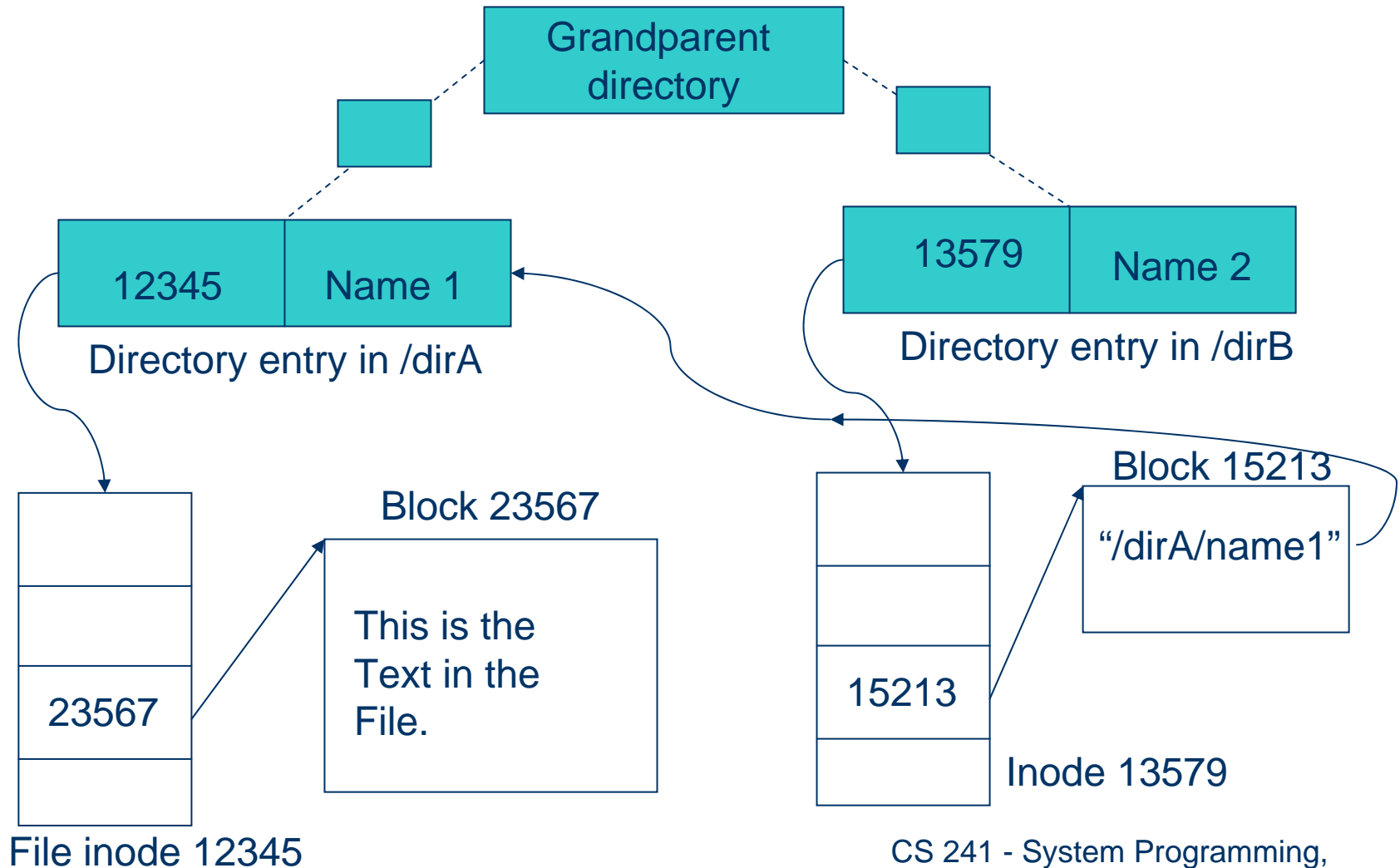
Hard Links and Symbolic Links

- A file may be accessed through multiple paths
 - Such linking between multiple names is known as **link** (also known as **hard link**)
 - UNIX allows users to make a new directory entry that points to an existing file
- 4.3 BSD also supported **symbolic links**, which are files containing the path name of another file or directory
 - **Soft (symbolic) links**, unlike hard links, could point to directories and could cross file-system boundaries

Hard Link Example



Symbolic Link Example



Summary

- Different directory organizations must be considered for file systems due to
 - size of the device
- Single directory (embedded devices/sensors)
- One directory per user (handheld devices)
- Arbitrary tree per user (PCs/laptops/workstations)