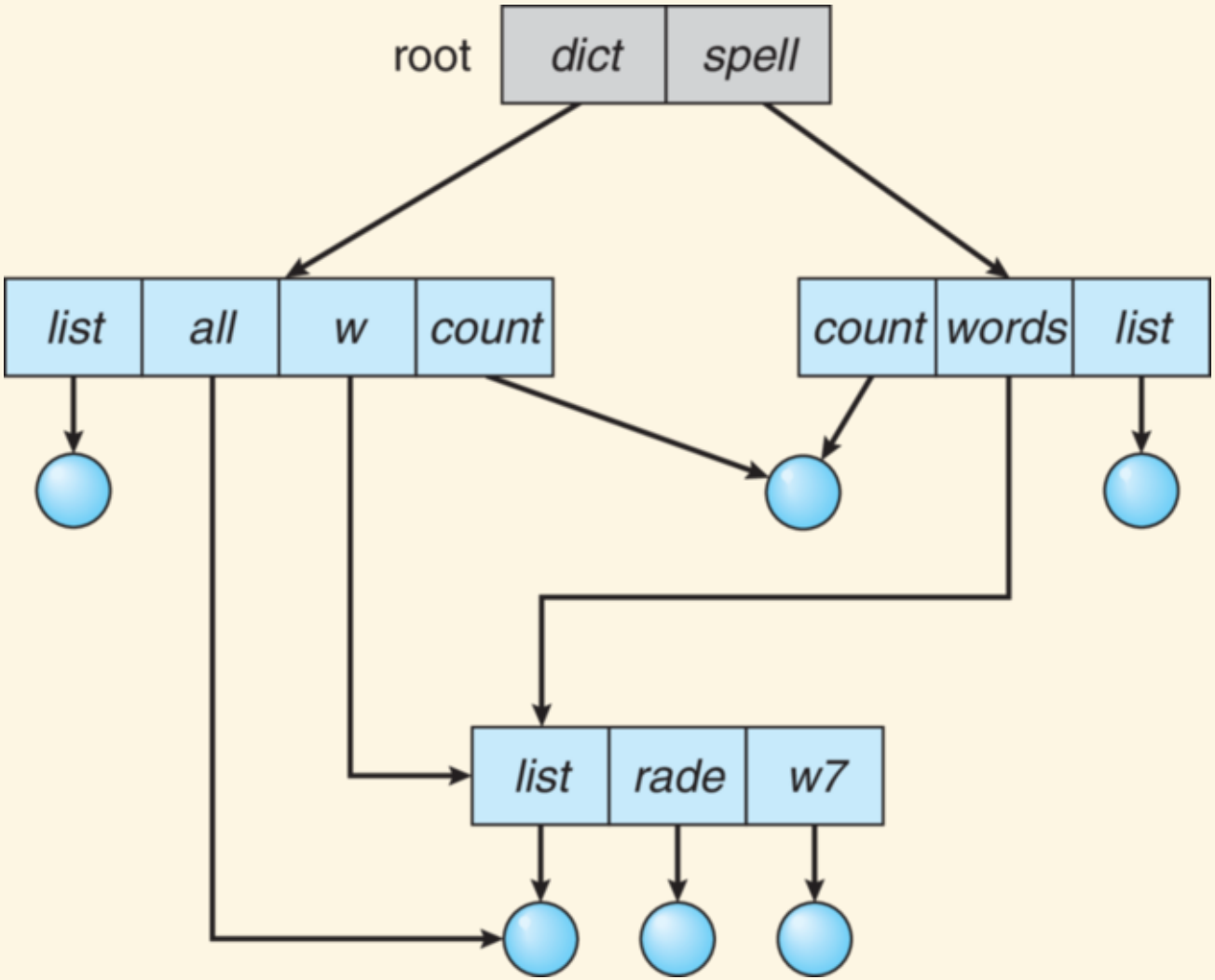
CIS 452 - Operating Systems Concepts

Nathan Bowman

Images taken from Silberschatz book

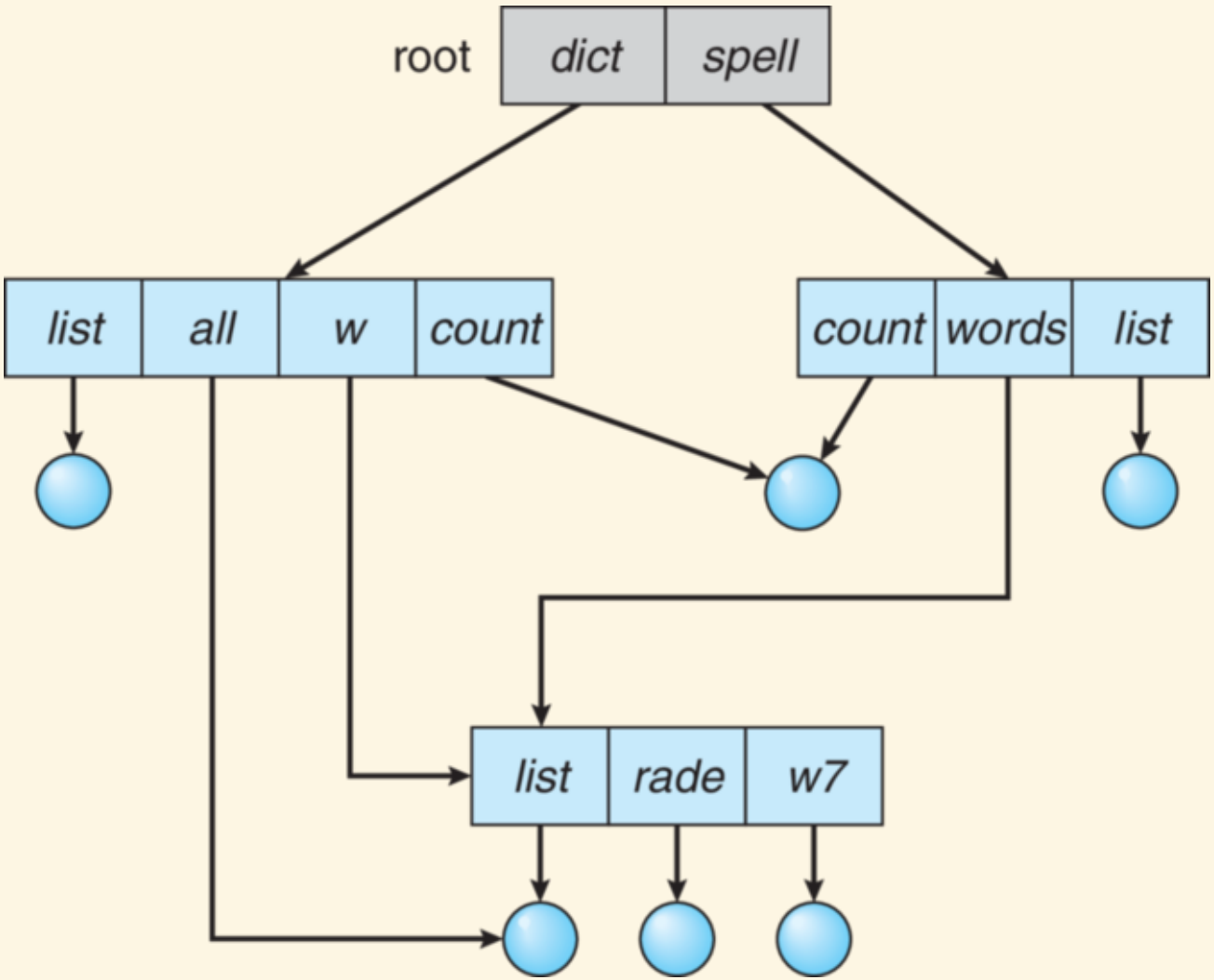
Links

`Link` exists when file or directory has more than one name


root
dict
spell
list
all
w
count
count
words
list
list
rade
w7

Links can be nonsymbolic (hard) or symbolic (soft)

Nonsymbolic (hard) links associate directory entry with another file


root
dict
spell
list
all
w
count
count
words
list
list
rade
w7

```
/dict

| Name  | inode    |
| ---   | ---      |
| list  | 00000003 |
| all   | 00000004 |
| w     | 00000005 |
| count | 00000006 |
```

```
/spell

| Name  | inode    |
| ---   | ---      |
| count | 00000006 |
| words | 00000005 |
| list  | 00000008 |
```

If `/dict/count` is updated, changes available immediately in `/spell/count`

Hard link is pointer to same underlying file (really, inode)

Note that names do not matter -- `/dict/w` is linked to `/spell/words`

Also, two or more files in same directory could be hard linked

```
/spell

| Name  | inode    |
| ---   | ---      |
| count | 00000006 |
| words | 00000005 |
| list  | 00000008 |
| list2 | 00000008 |
```

If files can be hard linked, OS must keep track of number of links to file

Space holding file cannot be reclaimed until all links removed

Symbolic (soft) links associate directory entry with another *path*

If `/spell/list2` were soft-linked to `/spell/list`, directory would look like

```
/spell

| Name  | inode    |
| ---   | ---      |
| count | 00000006 |
| words | 00000005 |
| list  | 00000008 |
| list2 | 00000009 |  <-- special link file
```

`list2` is a different file, but a special kind of file containing only filename `/spell/list1`

If program edits `list2`, it actually modifies whatever file resides at `/spell/list1`

What if different file is moved to `/spell/list1`?

```
/spell

| Name  | inode    |
| ---   | ---      |
| count | 00000006 |
| words | 00000005 |
| list  | 00000010 |
| list2 | 00000009 |
```

`list2` considers only the path

Changes to `list2` now affect inode 10 instead of inode 8

Similarly, if `/spell/list1` is deleted, `list2` will be meaningless

Some programs do not "follow" symbolic links, meaning they act on link itself (e.g., act on inode 00000009 directly in below example)

See `man 7 symlink` for details of how this works in Linux

```
/spell

| Name  | inode    |
| ---   | ---      |
| count | 00000006 |
| words | 00000005 |
| list  | 00000008 |
| list2 | 00000009 |
```

Hard link

```
$ ls -li
12063523 -rw-r--r-- 1 nate nate 12 Nov 30 17:32 prog1.c

$ ln prog1.c hard.c
$ ls -li
12063523 -rw-r--r-- 2 nate nate 12 Nov 30 17:32 hard.c
12063523 -rw-r--r-- 2 nate nate 12 Nov 30 17:32 prog1.c
```

```
$ rm prog1.c
$ ls -li
12063523 -rw-r--r-- 1 nate nate 12 Nov 30 17:32 hard.c
```

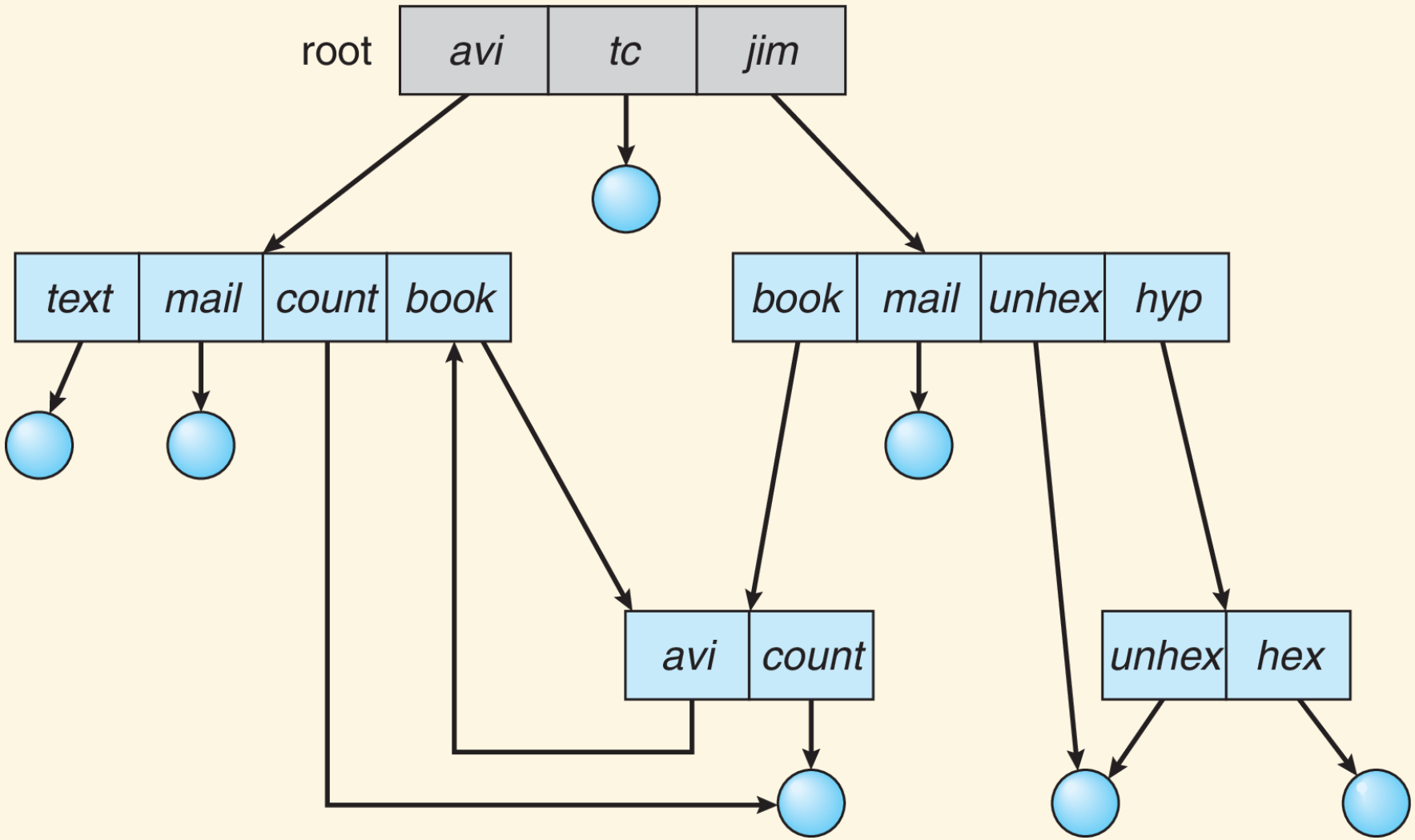
Soft link

```
$ ls -li
12063523 -rw-r--r-- 1 nate nate 12 Nov 30 17:32 prog1.c

$ ln -s prog1.c soft.c
$ ls -li
12063523 -rw-r--r-- 1 nate nate 12 Nov 30 17:32 prog1.c
12063524 lrwxrwxrwx 1 nate nate  7 Nov 30 17:37 soft.c -> prog1.c
```

```
$ rm prog1.c
$ ls -li
12063524 lrwxrwxrwx 1 nate nate 7 Nov 30 17:37 soft.c -> prog1.c
```

Links are very useful, but they complicate structure


root
avi
tc
jim
text
mail
count
book
book
mail
unhex
hyp
avi
count
unhex
hex

Problem with links, hard or soft, is that they can create cycles

Carelessly designed program can find itself in an infinite loop when traversing file tree

OS could disallow loops, but it would be expensive to run loop detection algorithm every time link is created

Linux disallows hard links to directories for this reason, but allows soft links to directories

Programs must be careful how they treat symbolic links when traversing file tree

See `man 7 symlink` for details