

6.3 Directories and Naming

Directories

Name-Space Management

Desired Properties of Directories

No restrictions on names

Fast

Space-efficient



Copyright © William C. Cheng

S5FS Directories

Component	Name
inode	Number

directory entry

.	1
..	1
unix	117
etc	4
home	18
pro	36
dev	93

- each entry is 32 bytes long in S5FS (see "s5fs.h" in weenix)
- this is what get stored inside the "data blocks" of a directory
- i.e., a directory is implemented as an inode and the disk map inside the inode points directly or indirectly to its "data blocks"

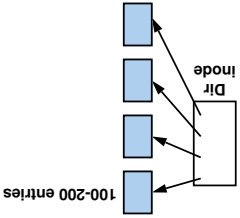


Copyright © William C. Cheng

Look Up Filename In A Directory

How to look for a file named "foo.c"?

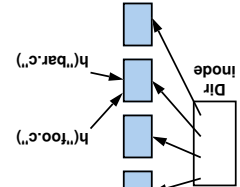
- $O(n)$ is no good
- linear search
- $O(\log n)$ is desirable
- B+ tree
- $O(1)$ is great
- hash table



Copyright © William C. Cheng

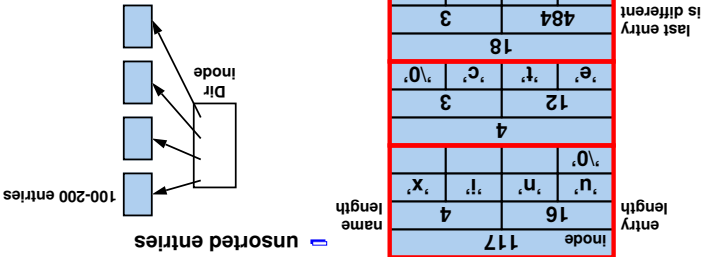
Hash Table

- Hash file names to disk blocks
- don't treat the directory contents as a sequential file
- treat it as an array of hash buckets
- h(filename) tells you which bucket (i.e., block) the file information is in
- one disk read
- The usual problem with hash tables
- collisions



Copyright © William C. Cheng

FFS Directory Format

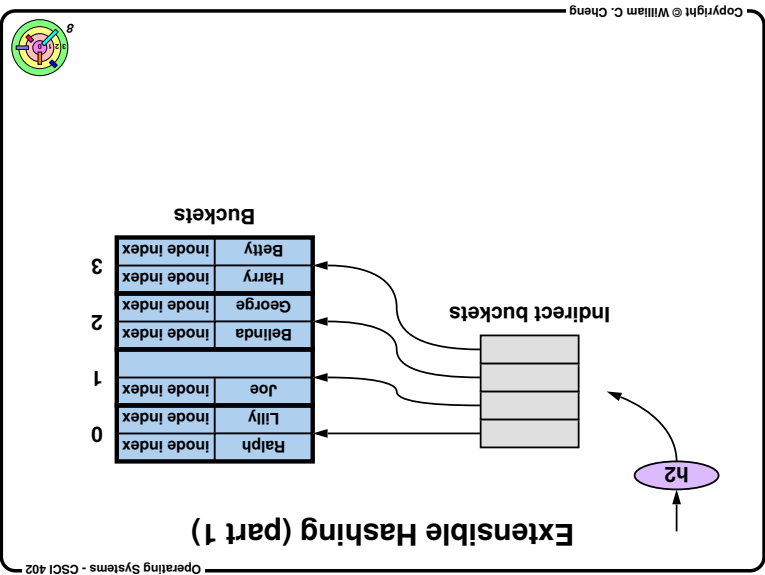
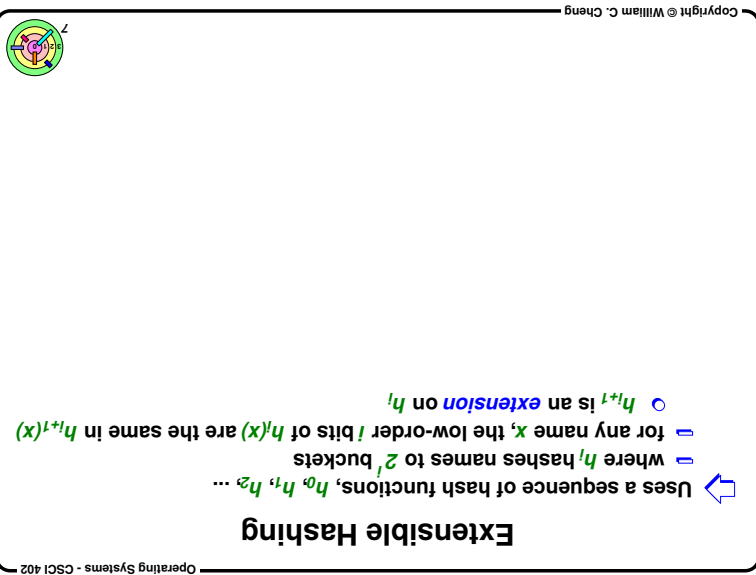
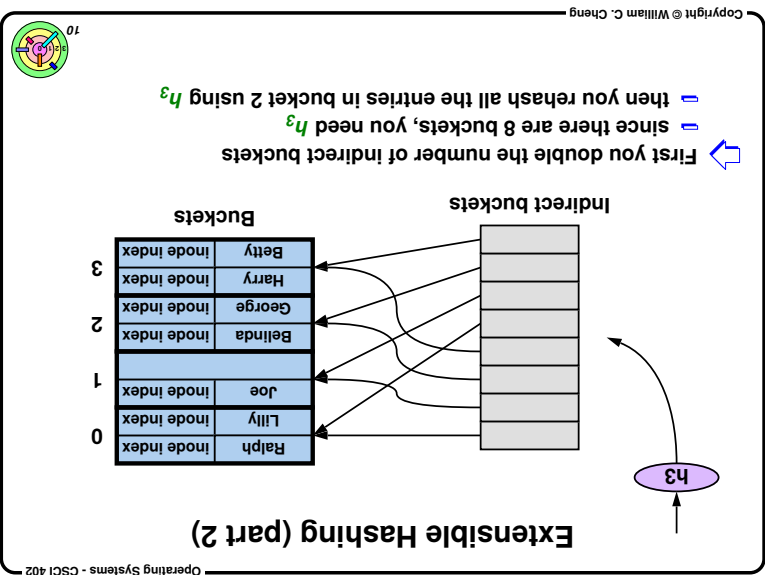
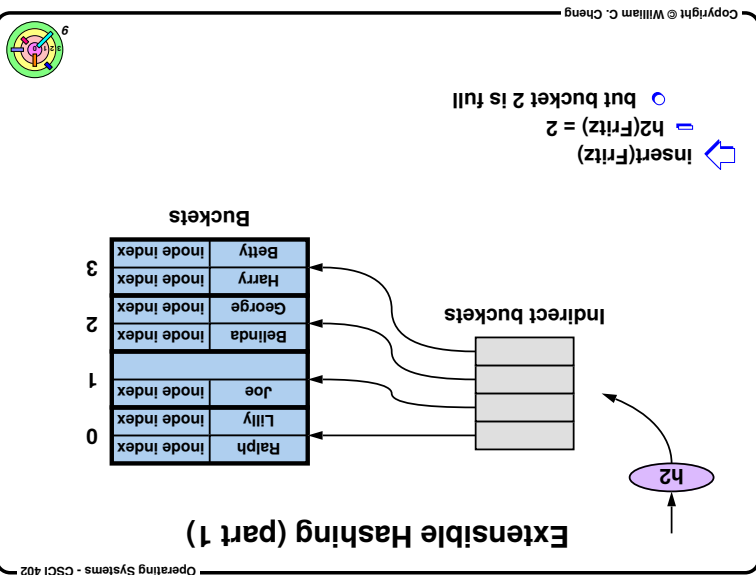
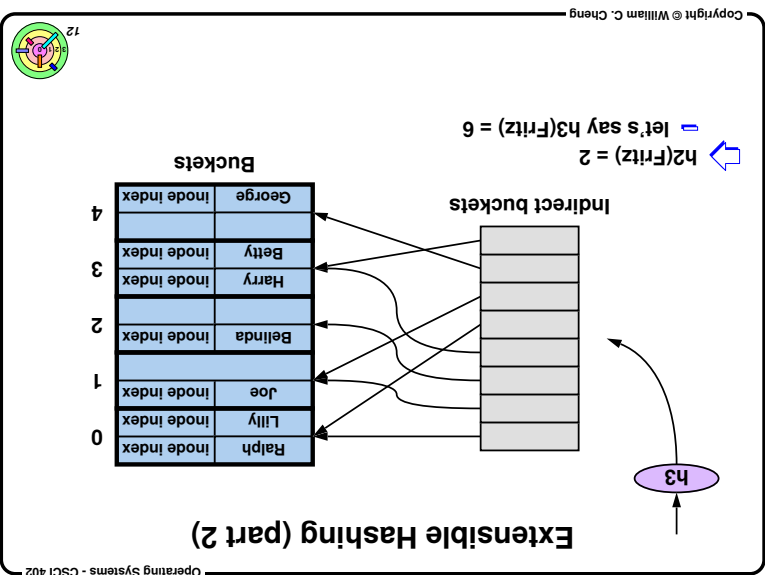
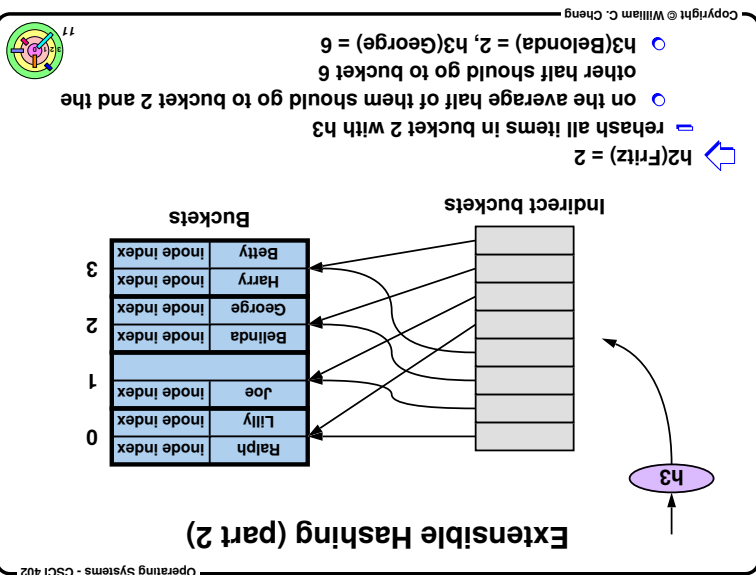


entry	length	name	inode
4	16	'u', 'n', 'x'	117
3	12	'e', 't', 'c'	4
18	484	'u', 's', 'r'	3

Directory Block



Copyright © William C. Cheng



Copyright © William C. Cheng

Extensible Hashing (part 3)

h3

Indirect buckets

Buckets

	node index
Ralph	0
Lilly	1
Joe	2
Belinda	3
Harry	4
Betty	5
Fritz	6
George	7

The power of *indirection*

- without the indirect buckets, it will be expensive

Copyright © William C. Cheng

B+ Trees (part 1)

Ex: a *B+ tree* of order 3

- every internal node has either 2 or 3 children
- if a node has > 3 children, it needs to split into two nodes
- if a node has < 2 children, it needs to merge with its neighbor

Copyright © William C. Cheng

B+ Trees (part 1)

Insertion: e.g., "Lucy"

Copyright © William C. Cheng

B Trees

A *B tree* of order *m* has the following properties:

- every node has $\leq m$ children
- every node (except root & leaves) has $\geq \text{ceil}(m/2)$ children
- the root has at least 2 children (unless it is also a leaf)
- all leaves appear at the *same level* and carry no keys
- a non-leaf node with *k* children contains *k-1* keys

guarantee that each node of size *m* is at least 50% full

Each *node* in a *B tree* corresponds to one *disk block*

B+ tree:

- internal nodes contain no data, just keys
- leaf nodes are linked to ease sorted sequential traversal

there are a lot of B tree variations in various application areas

We will illustrate the basic idea with an example

actual B+ tree algorithm is more complicated than depicted

Copyright © William C. Cheng

B+ Trees (part 1)

Insertion: e.g., "Lucy"

Copyright © William C. Cheng

B+ Trees (part 2)

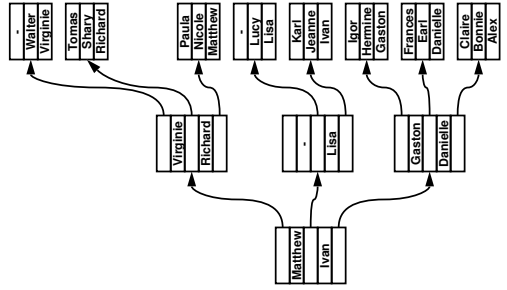
Insertion: e.g., "Lucy"

- may cause split at the root node
- increase depth



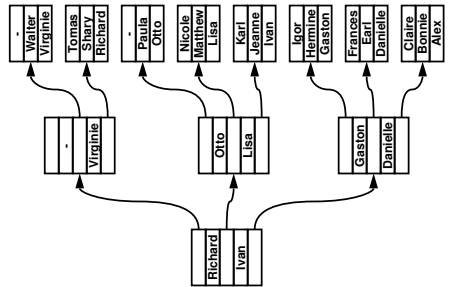
- Multiple partitions of hard drive
 - on Windows, you have C:, D:, etc.
 - on Unix, you have /dev/sda0 (ext2), /dev/sda1 (ext3), etc.
- USB (JFFS2), CDROM (ISO9660)
- Main challenge for name-space management
 - how do you make the name-space appear uniform?
 - Windows: drives
 - Unix: file system *mounting*
 - create an entry in the *inode* to point to the file system
 - only happen in *memory* (in the buffer cache) and not on disk


Name-Space Management



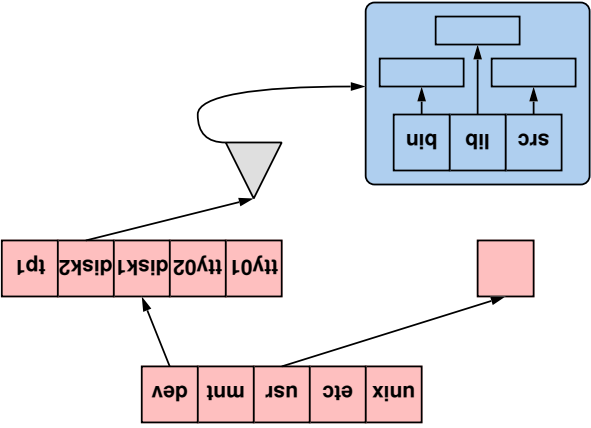
- ➡ **Deletion:** e.g., "Otto"
- ➡ merge nodes if necessary
- ➡ may decrease depth

B+ Trees (part 3)

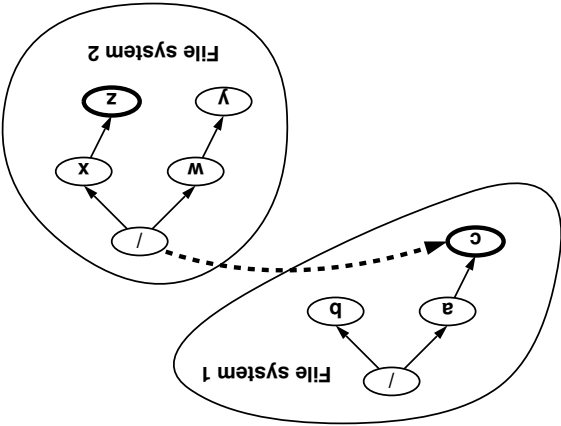


- 
Insertion: e.g., "Lucy"
 there is another way
 ○ "rotate" Nicole to the right (since cannot "rotate" Lisa to the left)

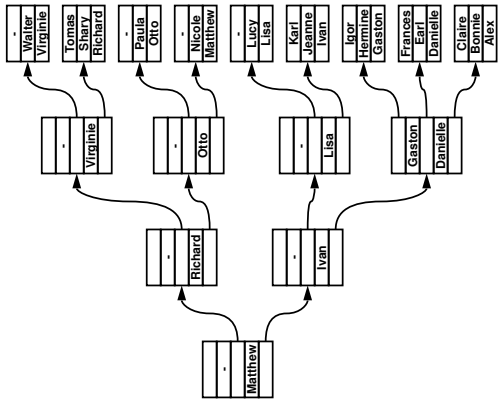
B+ Trees (part 2)



Mount Points (1)



Name-Space Management



- Deletion: e.g., "Otto"

B+ Trees (part 3)

Operating Systems - CSCI 402

Copyright © William C. Cheng

Mount Points (2)

```
graph TD; subgraph FS [File System]; src; lib; bin; end; mnt_dev[mnt dev]; tty01_tty02_disk1_disk2_tp1[tty01 tty02 disk1 disk2 tp1]; dir(( )); mnt_dev -.-> FS; mnt_dev --> dir; dir --> tty01_tty02_disk1_disk2_tp1; tty01_tty02_disk1_disk2_tp1 --> dir; label[mount /dev/disk2 /usr] --> dir;
```

25

Copyright © William C. Cheng

Operating Systems - CSCI 402

Copyright © William C. Cheng

File System Mounting

- mount /dev/disk2 /usr
- /usr/bin/bash is to be located at /dev/disk2/bin/bash
- mount /dev/usb0 /mnt/usb
- on Mac OS X, instead of /mnt, it uses /Volumes

Weenix

- Try MOUNTING=1 in Config.mk
- may be after the semester is over
- polymorphism

26

Copyright © William C. Cheng