

Operating Systems Design (CS 423)

Elsa L Gunter
2112 SC, UIUC

<http://www.cs.illinois.edu/class/cs423/>

Based on slides by Roy Campbell, Sam King, and
Andrew S Tanenbaum

4/29/11

1

Indexed Files

Pros

- Can easily grow (up to # of blocks allocated in header)
- Easy random access of loc. Calculation

Cons

- Lots of seeks for sequential access
 - How can you make this faster without pre-allocation?
 - Try to keep sequential access in same cylinder on disk
- Can't easily grow beyond # blocks allocation

4/29/11

2

Large Files

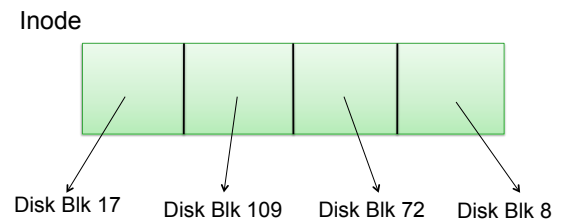
- How to deal with large files?
 - Could you assume file might get really large, allocate lots of space in file header?
 - Could you use larger block size, eg 4MB?
- Solution: more sophisticated data structure for file header

4/29/11

3

Indexed Files

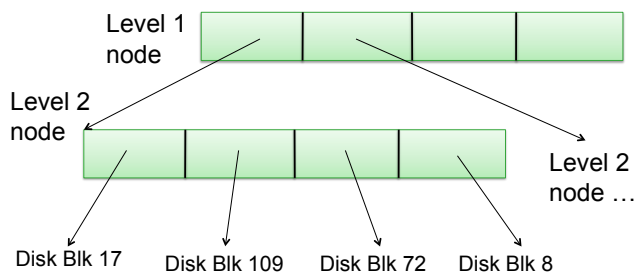
- Indexed files are like a shallow tree



4/29/11

4

Multi-level Indexed Files



4/29/11

5

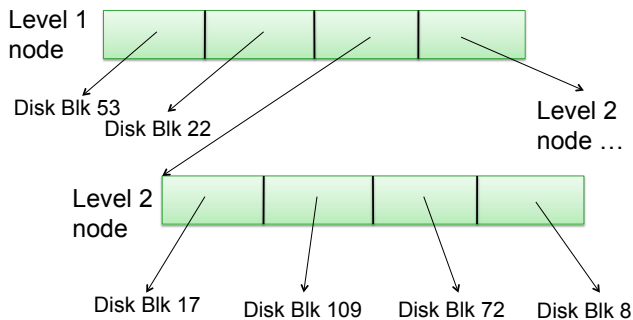
Multi-level Indexed Files

- How many disk accesses to get 1 block of data?
- How do you solve this?

4/29/11

6

Non-Uniform Multi-level Indexed Files



4/29/11

7

Non-Uniform Multi-level Indexed Files

- Pros
 - Files can expand easily
 - Small files don't pay full overhead of deep trees
- Cons
 - Lots of indirect blocks for big files
 - Lots of seeks for sequential access

4/29/11

8

On Disk File Structures

- Could have other dynamically allocated data structures for file header
- Key feature: have location of file header on disk NOT change when file grows
 - Why?

4/29/11

9

Naming Files

- How do you specify which file you want to access?
 - Eventually OS must find file header you want on disk
 - Need disk block address (number)
- Typically user uses symbolic name
 - OS translates name to numeric file header
 - Possible alternative is to describe contents of file

4/29/11

10

Locating File Header Disk Block

- Could use hash table, expandable array
 - Key is finding disk block number of file inode; then getting contents is easy
- Data structure for mapping file name to inode block number is called a **Directory**

4/29/11

11

Directories

- Directory – mapping for set of files
 - Name->file header's disk block # for that file
 - Often simple array of (name, file header's disk block #) entries
 - Table is stored in a normal file as normal data
 - Eg: **ls** implemented by reading file and parsing contents

4/29/11

12

Directories

- Often treat directories and files in same way
 - Same storage structure for data
 - Directory entry points to either "ordinary" file or another directory
- Can we allow user to read/write directories directly, arbitrarily?

4/29/11

13

Directory Organization

- Directories typically have hierarchical structure
 - Directory **A** has mapping to files and *directories* in directory **A**
- `/home/cs423/index.html`
- `/` is root directory
 - Contains list of root's contents, including home
 - For each elt, has mapping from name to file inode disk block #]
 - Including home

4/29/11

14

Directory Organization

- home is directory entry within `/` dir
 - Contains list of files and directories
 - One dir in home is `cs423`
- `/home/cs423` names directory within `/home` directory
 - Contains list of files and directories
 - One file is `index.html`
 - How many disk I/Os to access first bytes of `/home/cs423/index.html` assuming no caching

4/29/11

15

Solution

root header
size = 1024



Fixed location



4/29/11

16

Solution

root header
size = 1024

Data block for `/`,
dir entries



Disk block 12



4/29/11

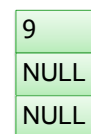
17

Solution

root header
size = 1024

Data block for `/`,
dir entries

`/home` header



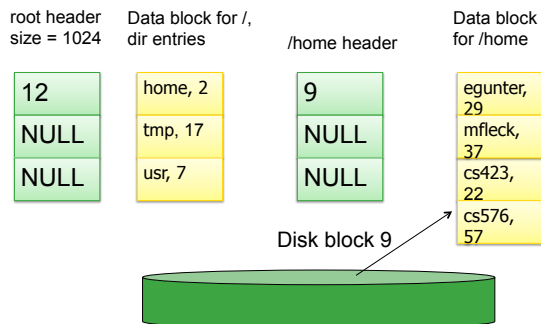
Disk block 2



4/29/11

18

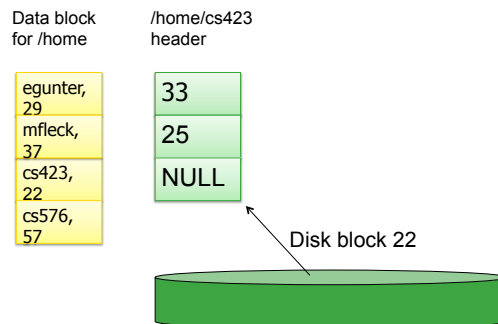
Solution



4/29/11

19

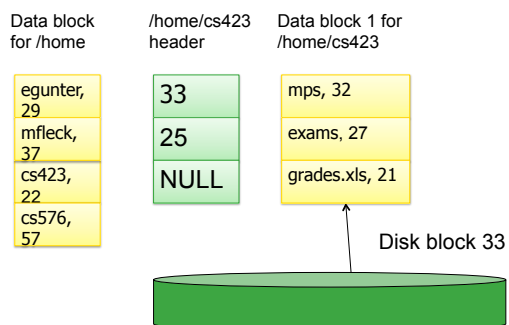
Solution



4/29/11

20

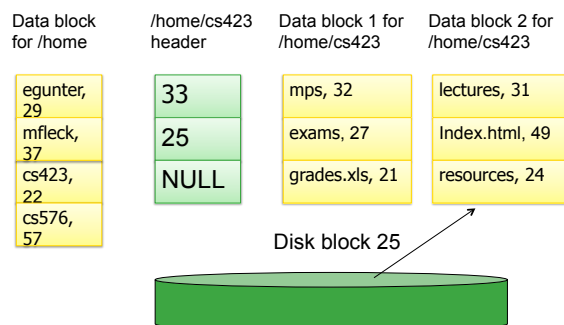
Solution



4/29/11

21

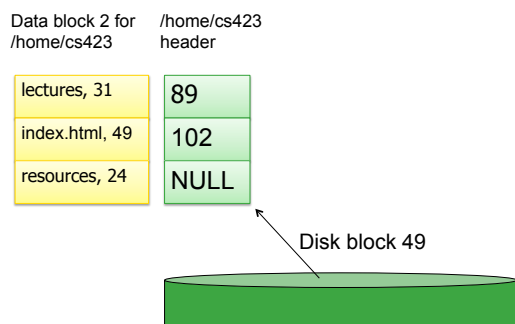
Solution



4/29/11

22

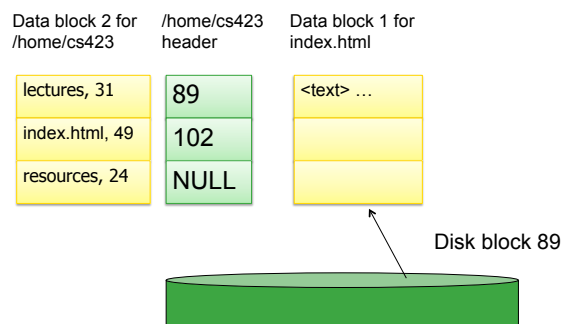
Solution



4/29/11

23

Solution



4/29/11

24

Directories

- Only reasonable performance improvement will be from caching
- Temporal locality is favored heavily
- E.g., if /home/cs423 file header cached, can skip half the work

4/29/11

25

Multi-step updates and reliability

- Reliability (durability) is especially important to file systems
- Data in a process address space need not survive a system crash
- Users consider data in file system as permanent
- Multi-step updates cause problems if crash happens in the middle

4/29/11

26

Multi-step updates and reliability

- Transfer \$100 from my checking account to my savings account
 - Deduct \$100 from my checking account
 - Add \$100 to savings account
- Move file from one directory to another
 - Delete file from old directory
 - Add file to new directory
- Create new file
 - Update directory to point to new file header
 - Write new file header to disk
- What happens if you crash between steps 1 & 2?

4/29/11

27

Moving a File: grades.xml -> exams/rec.xml

/home/cs423 header	Data block 1 for /home/cs423	/home/cs423/ exams header	Data block 1 for /home/cs423
33	mps, 32	42	midterm1, 72
25	exams, 27	NULL	final, 97
NULL	grades.xls, 21	NULL	NULL

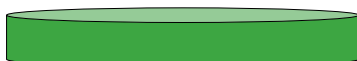


4/29/11

28

Moving a File: grades.xml -> exams/record.xml

/home/cs423 header	Data block 1 for /home/cs423	/home/cs423/ exams header	Data block 1 for /home/cs423
33	mps, 32	42	midterm1, 72
25	exams, 27	NULL	final, 97
NULL	NULL	NULL	record.xls, 21



4/29/11

29

Careful Ordering

- Create new file example
 - Write new file header to disk
 - Update directory to point to new file header
- What happens if you crash after step 1, does new file exist?

4/29/11

30



Careful Ordering

- What if I need also to update disk block containing list of free disk blocks?
- Does this work?
 - Write new file header to disk
 - Update directory to point to new file header
 - Write the new free map (to reflect that the file header is not free)

4/29/11

31



Careful Ordering

- What if I need also to update disk block containing list of free disk blocks?
- Does this work?
 - Write new file header to disk
 - Update directory to point to new file header
 - Write the new free map (to reflect that the file header is not free)
- No. Crash can lead to file header being over-written

4/29/11

32



Careful Ordering

- Does this work?
 - Write new file header to disk
 - Write the new free map (to reflect that the file header is not free)
 - Update directory to point to new file header
- Side note: disk scheduler has to honor this order!!!

4/29/11

33



Careful Ordering

- Careful ordering works for these simple cases, but not in general
- Can careful ordering alone solve the problem of transferring money from checking account to savings account?
- What other CS423 concept does this remind you of?
- File systems borrow heavily from database transactions

4/29/11

34