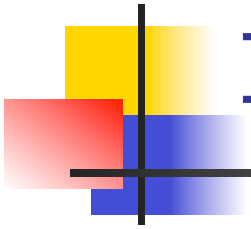
Operating Systems Design (CS 423)

Elsa L Gunter

2112 SC, UIUC

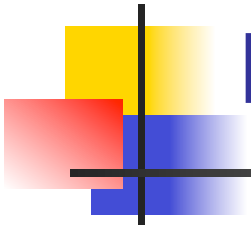
http://www.cs.illinois.edu/class/cs423/

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



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

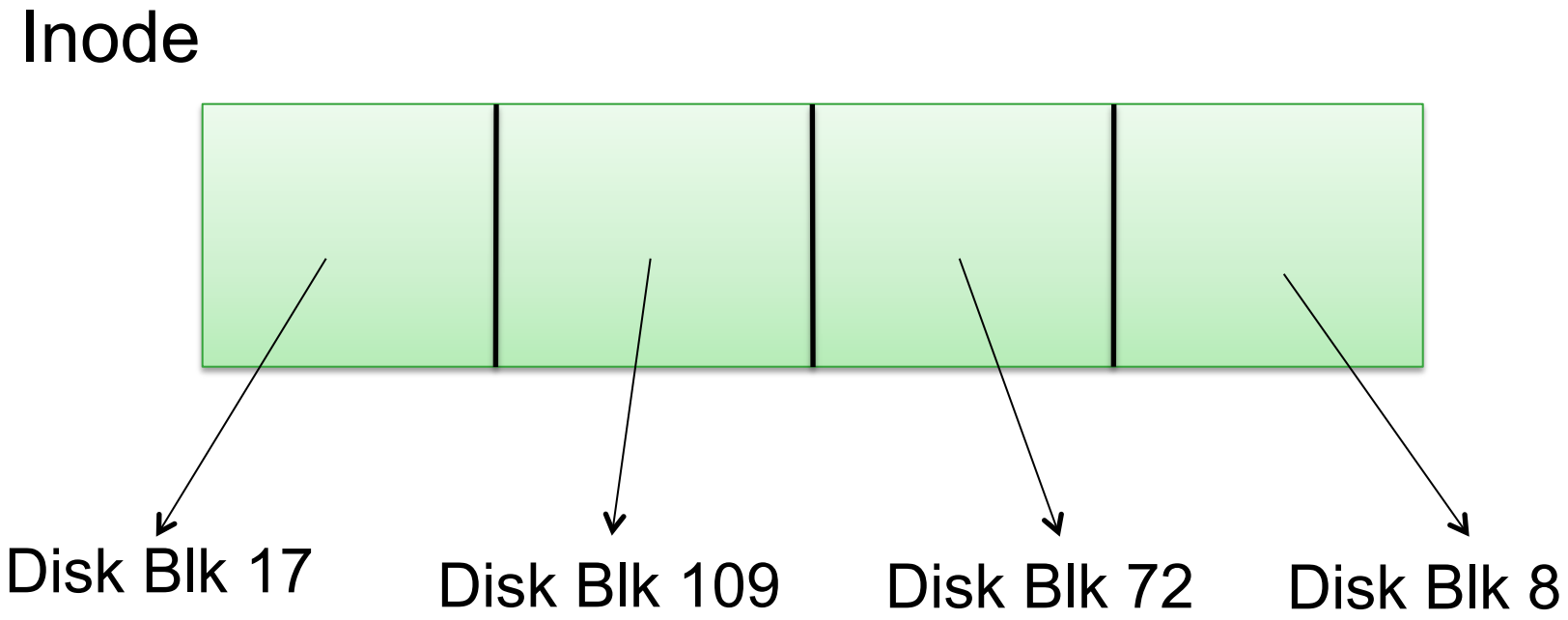


Large Files

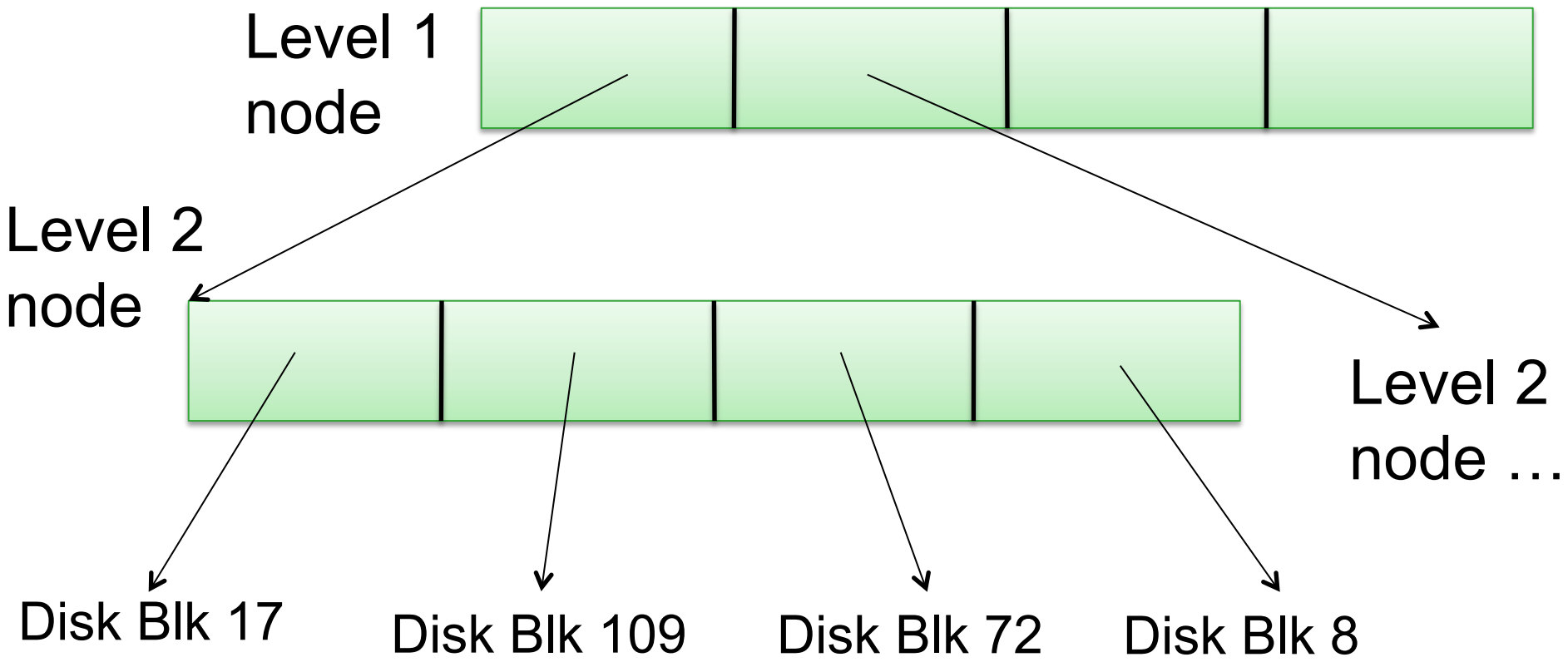
- How to deal with large files?
  - Could you assume file might get really large, allocate lots of space in file header?
  - Could you use larger block size, eg4MB?
- Solution: more sophisticated data structure for file header

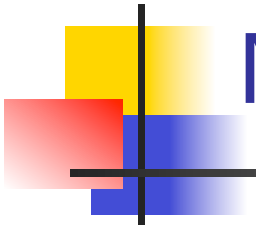
Indexed Files

- Indexed files are like a shallow tree



Muti-level Indexed Files



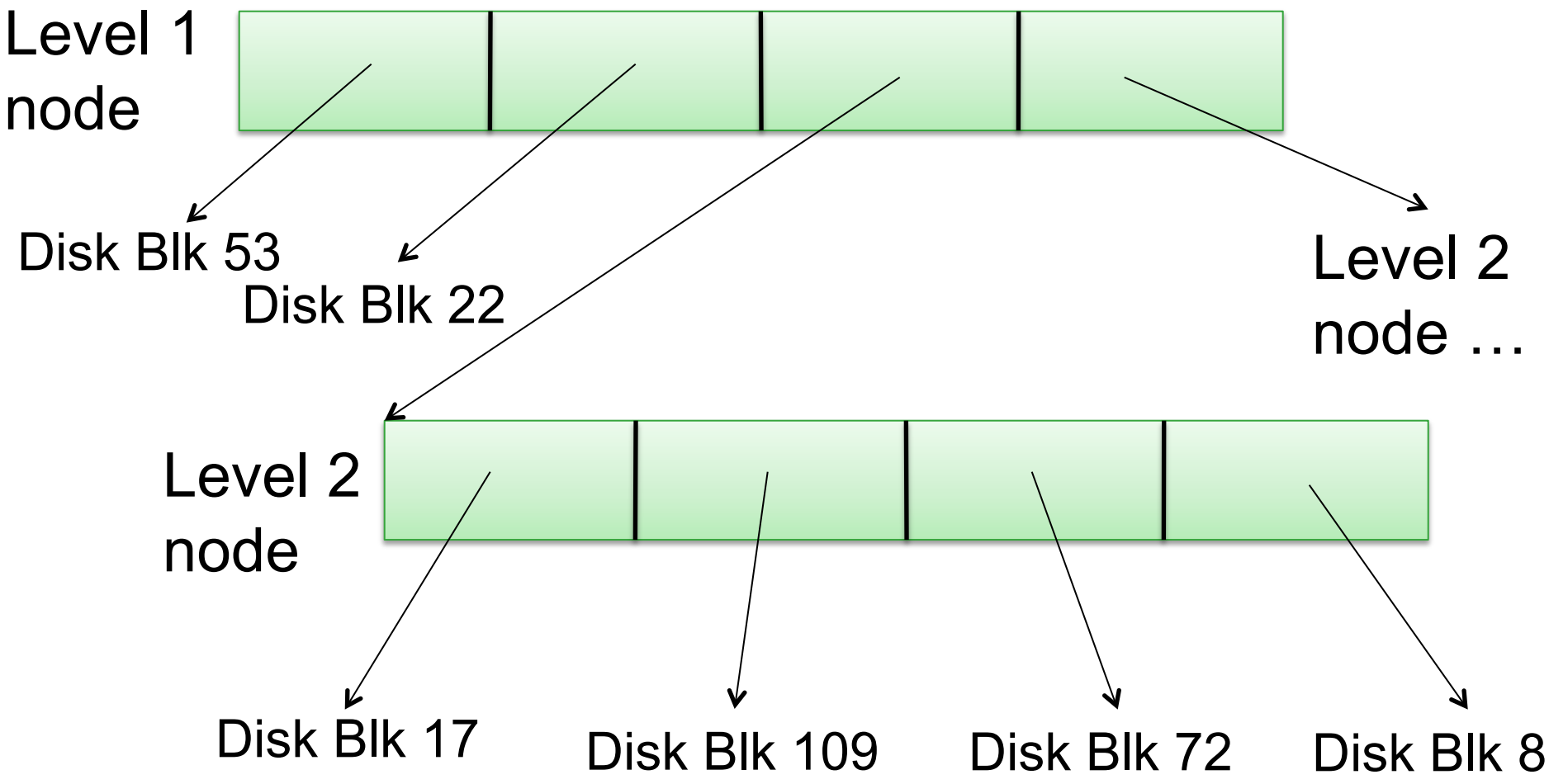


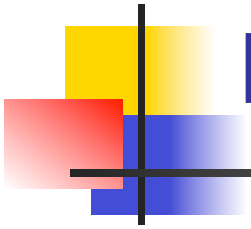
Muti-level Indexed Files

- How many disk accesses to get 1 block of data?

- How do you solve this?

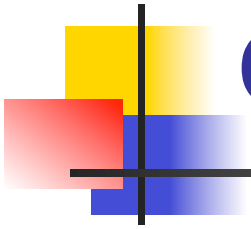
Non-Uniform Muti-level Indexed Files





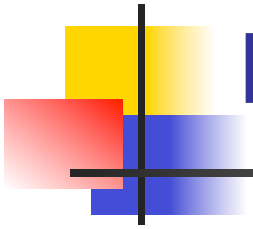
Non-Uniform Muti-level Indexed Files

- 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



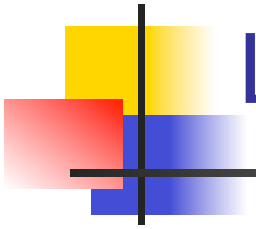
On Disk File Structures

- 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?



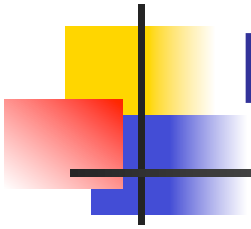
Naming Files

- 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



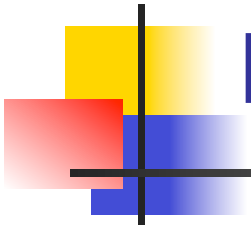
Locating File Header Disk Block

- 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**



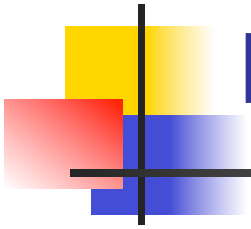
Directories

- 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



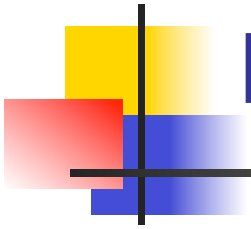
Directories

- 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?



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 form name to file inode disk block #]
    - Including `home`



Directory Organization

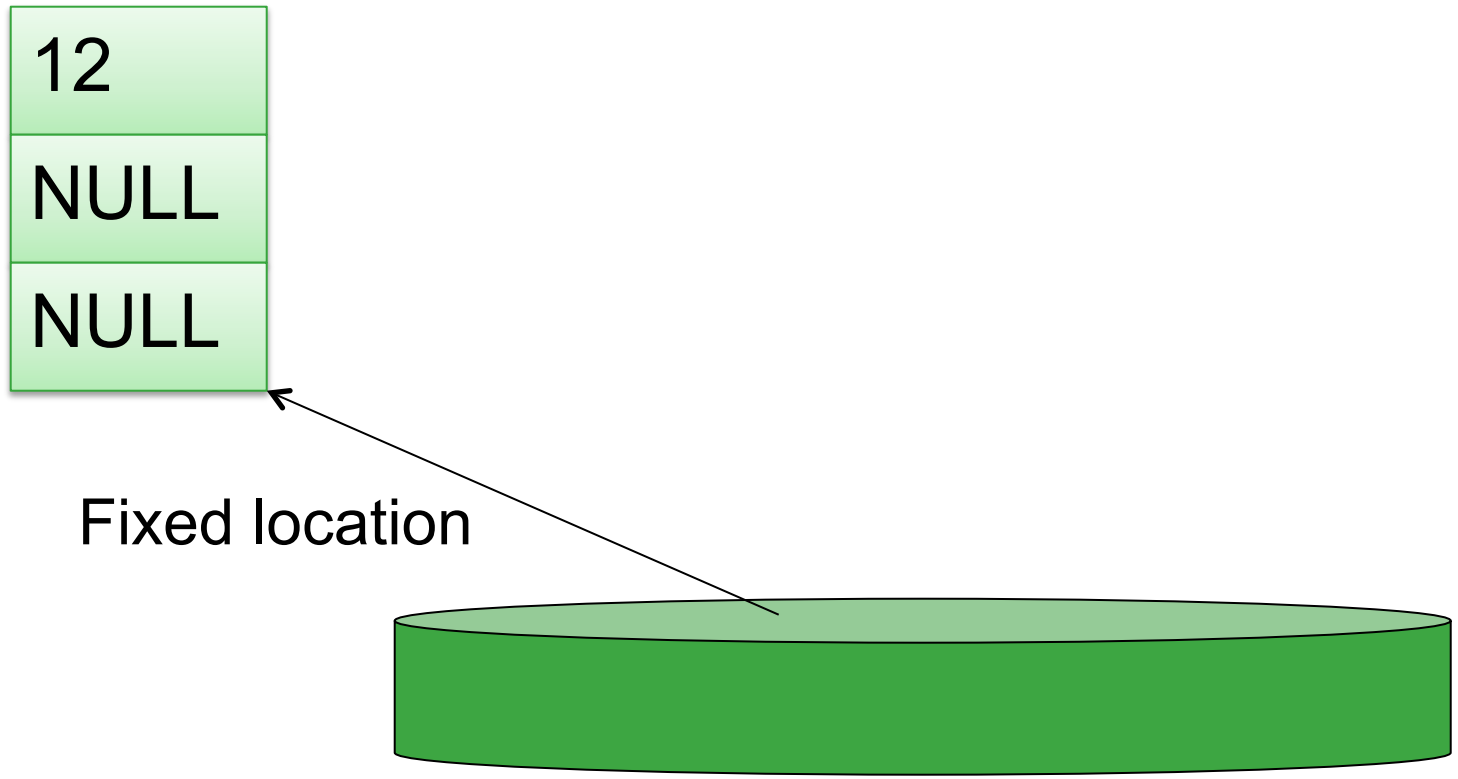
- `home` is directory entry within / dir
  - Contains list of files and directories
  - One dir in `home` is `cs423`
- `/home/cs423` names directory within `/home` directory
  - Contains list of files and directories
  - One file is `index.html`
  - How many disk I/Os to access first bytes of

    `/home/cs423/index.html`

    assuming no caching

Solution

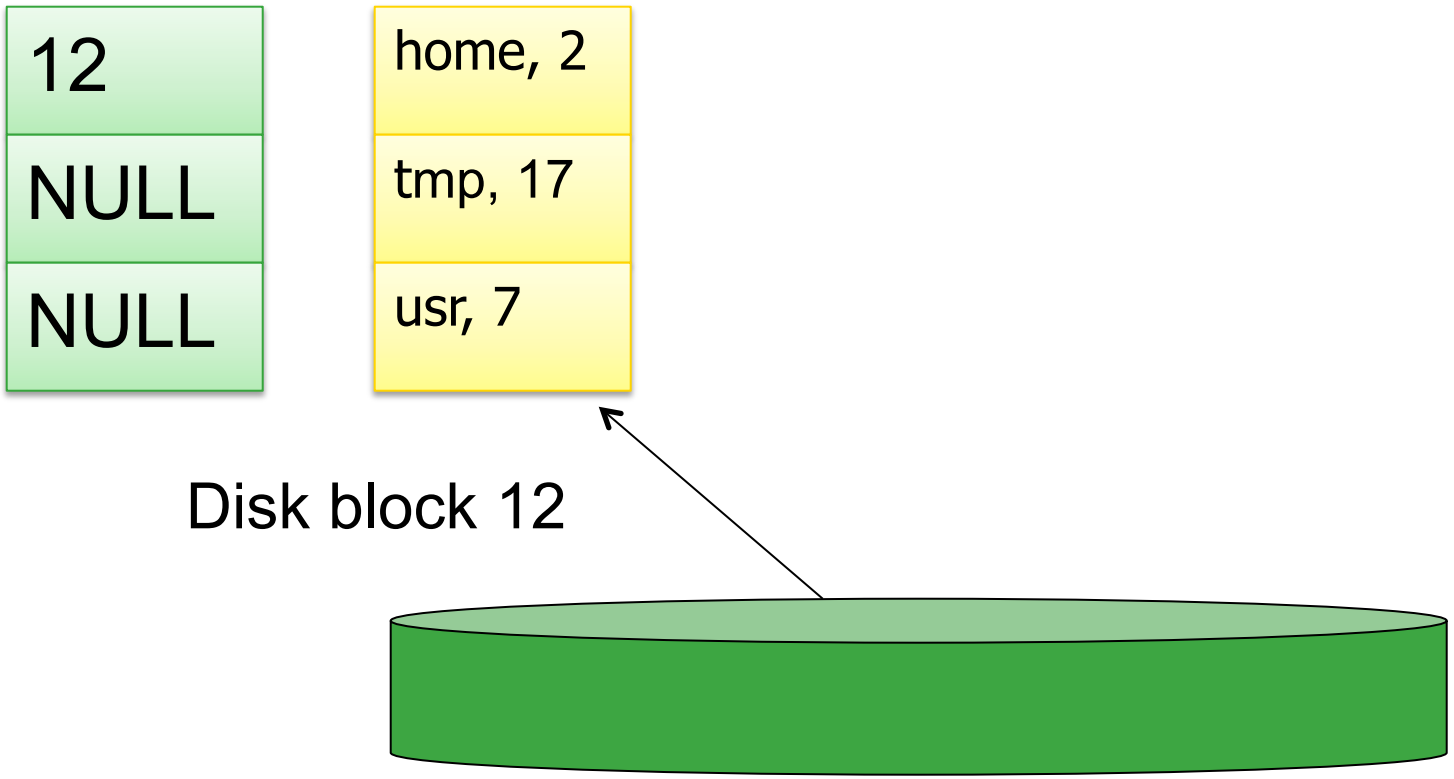
root header
size = 1024



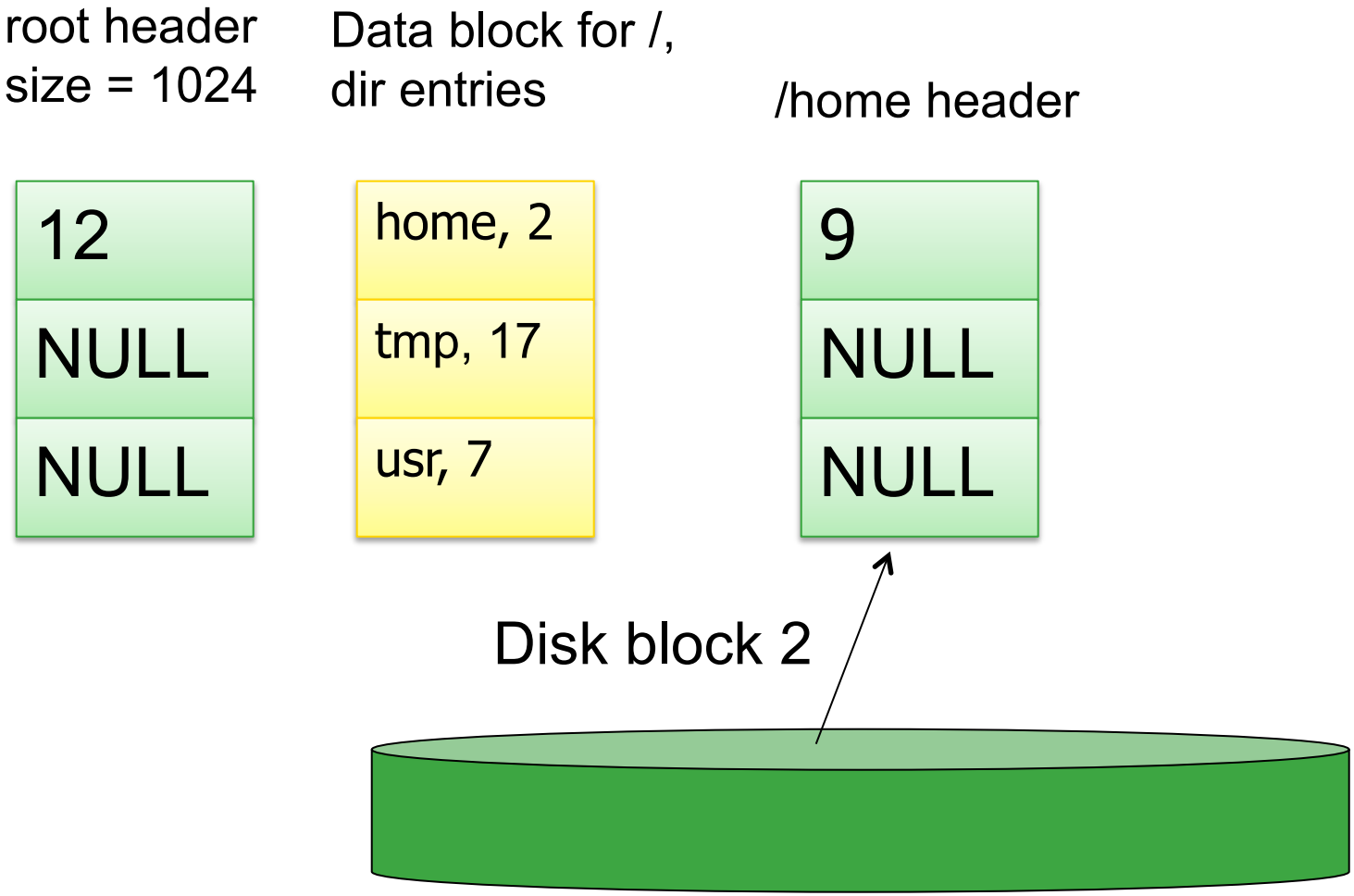
Solution

root header
size = 1024

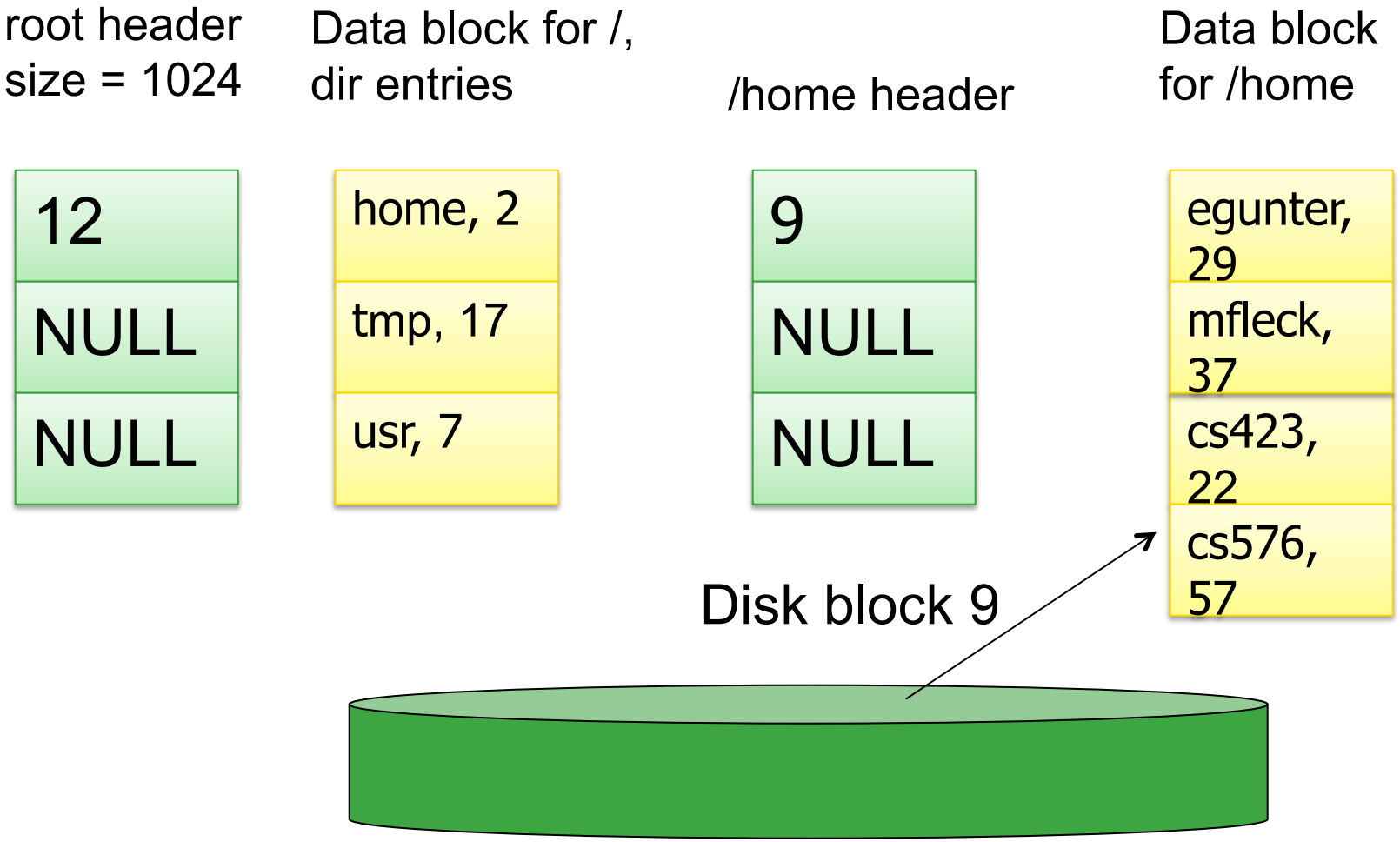
Data block for /,
dir entries



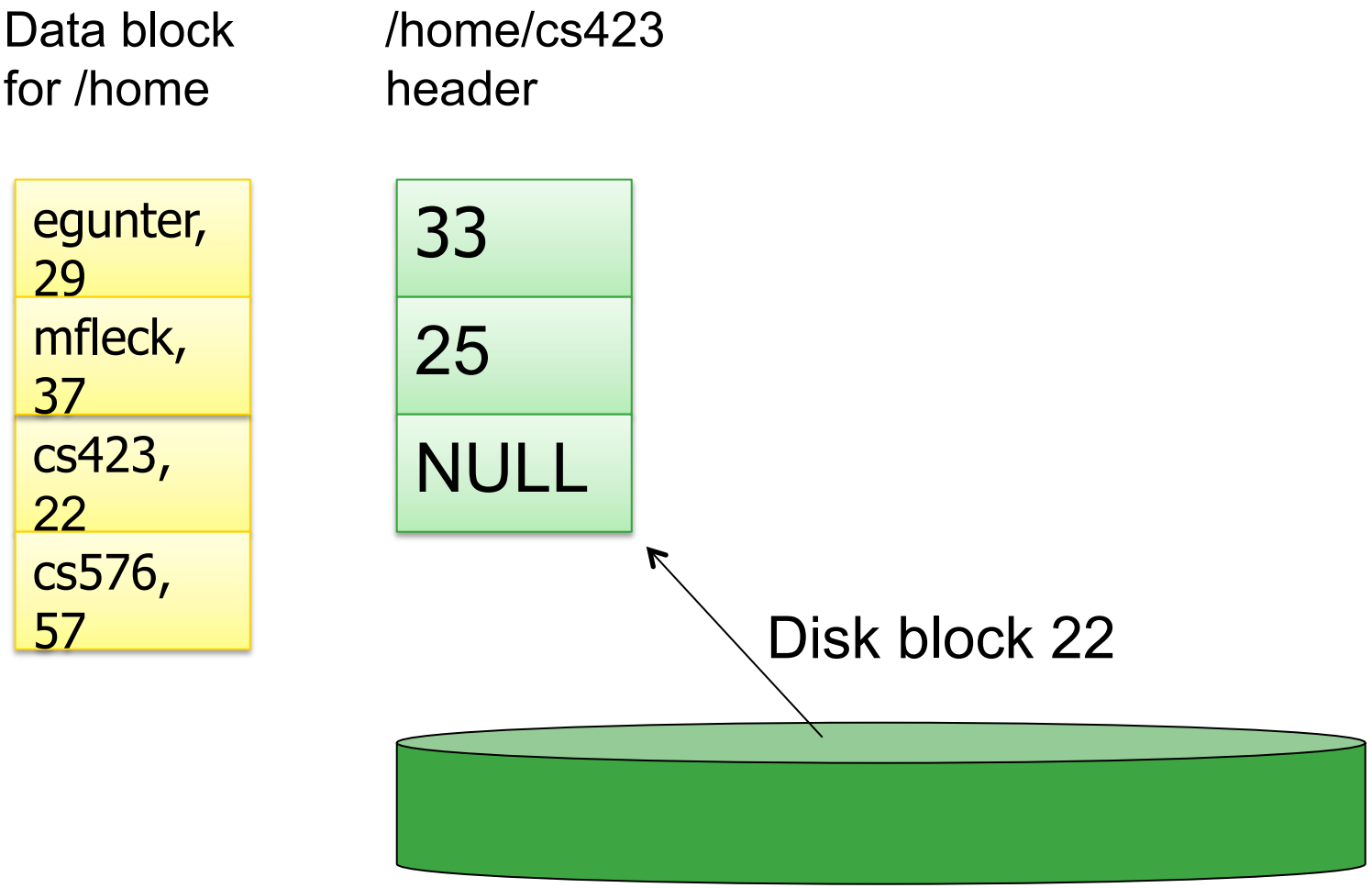
Solution


root header
size = 1024
Data block for /,
dir entries
/home header
12
NULL
NULL
home, 2
tmp, 17
usr, 7
9
NULL
NULL
Disk block 2

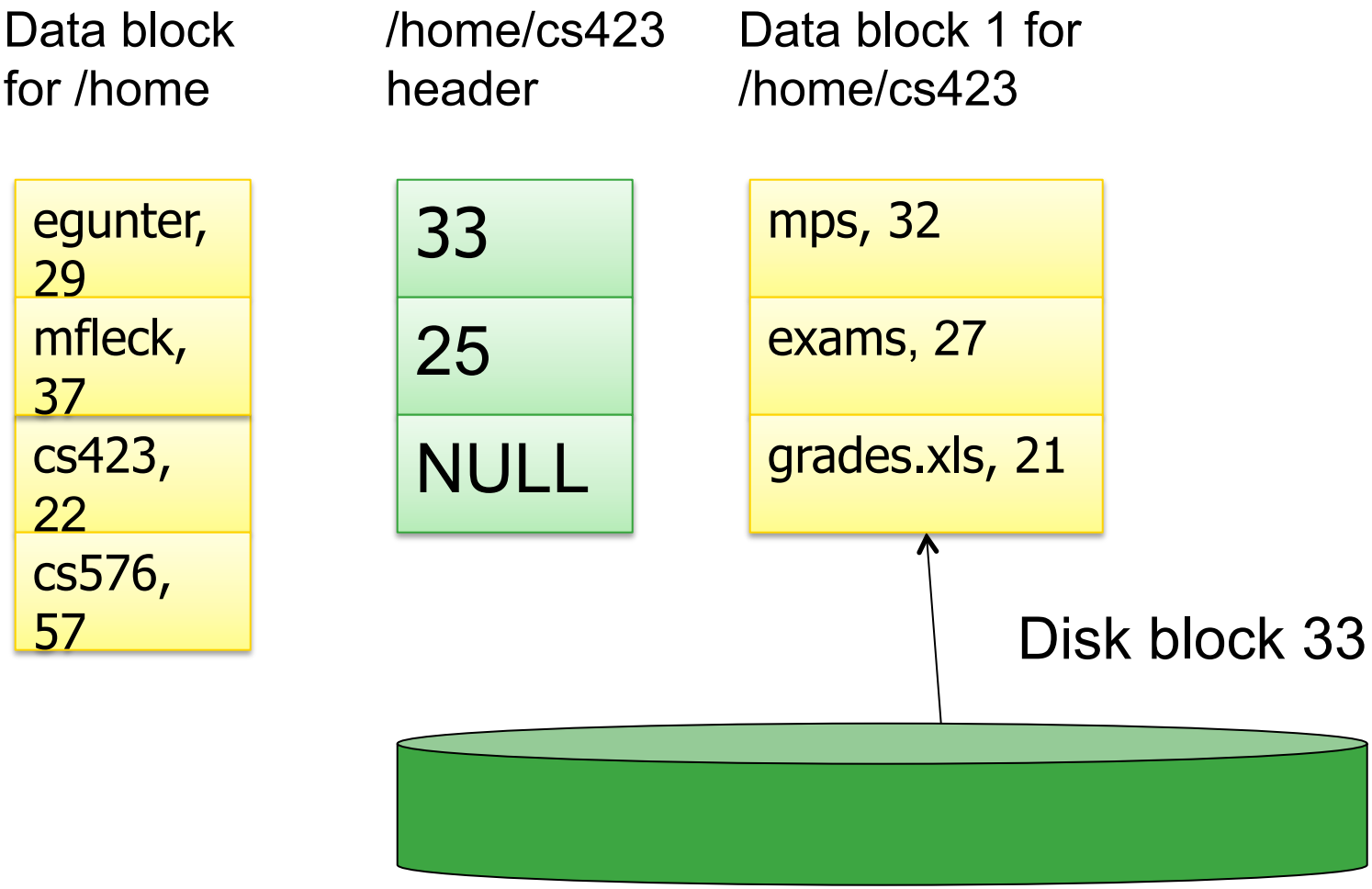
Solution


root header
size = 1024
Data block for /,
dir entries
/home header
Data block
for /home
12
NULL
NULL
home, 2
tmp, 17
usr, 7
9
NULL
NULL
egunter,
29
mfleck,
37
cs423,
22
cs576,
57
Disk block 9

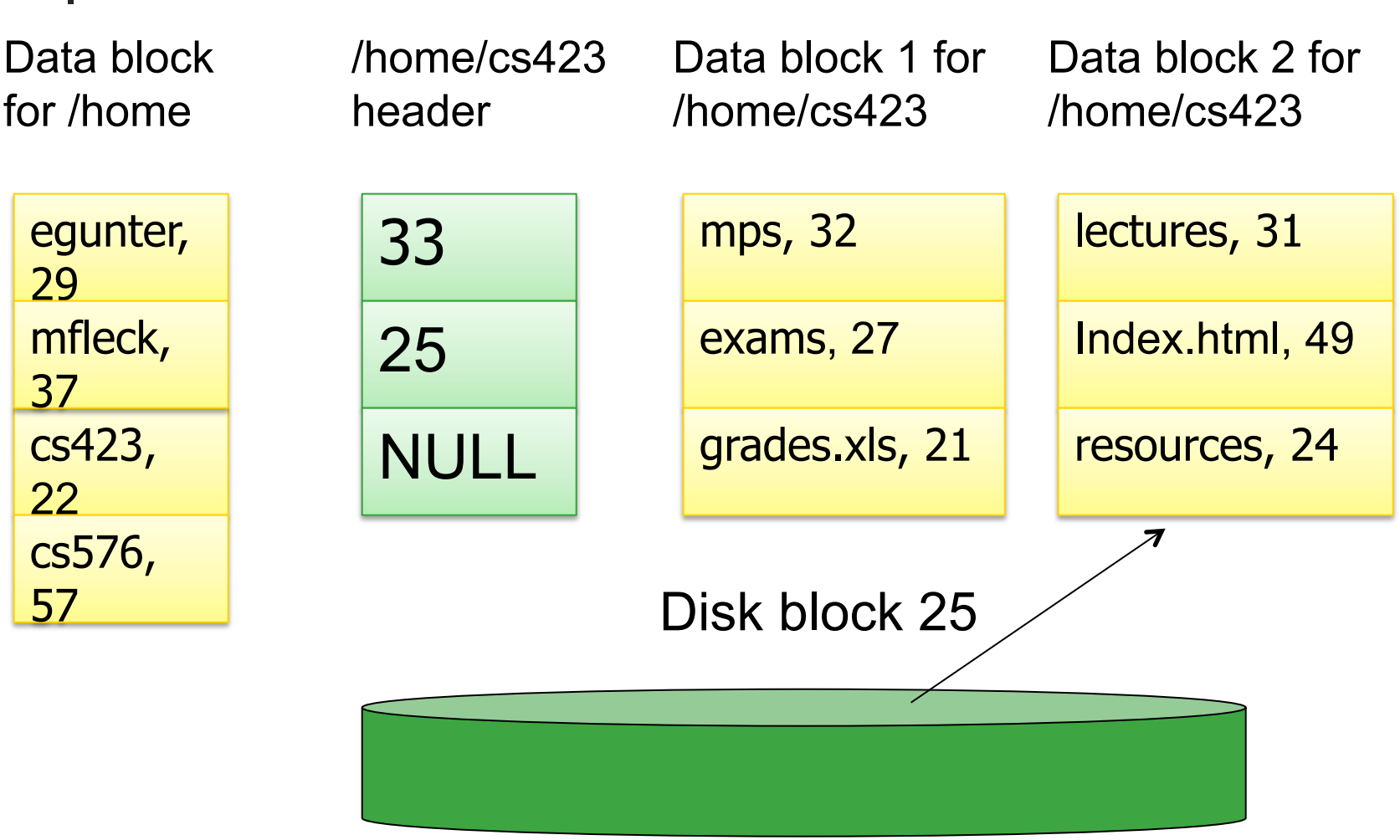
Solution


Data block for /home
/home/cs423 header
egunter, 29
mfleck, 37
cs423, 22
cs576, 57
33
25
NULL
Disk block 22

Solution


Data block for /home
/home/cs423 header
Data block 1 for /home/cs423
egunter, 29
mfleck, 37
cs423, 22
cs576, 57
33
25
NULL
mps, 32
exams, 27
grades.xls, 21
Disk block 33

Solution


Data block for /home
/home/cs423 header
Data block 1 for /home/cs423
Data block 2 for /home/cs423
egunter, 29
mfleck, 37
cs423, 22
cs576, 57
33
25
NULL
mps, 32
exams, 27
grades.xls, 21
lectures, 31
Index.html, 49
resources, 24
Disk block 25

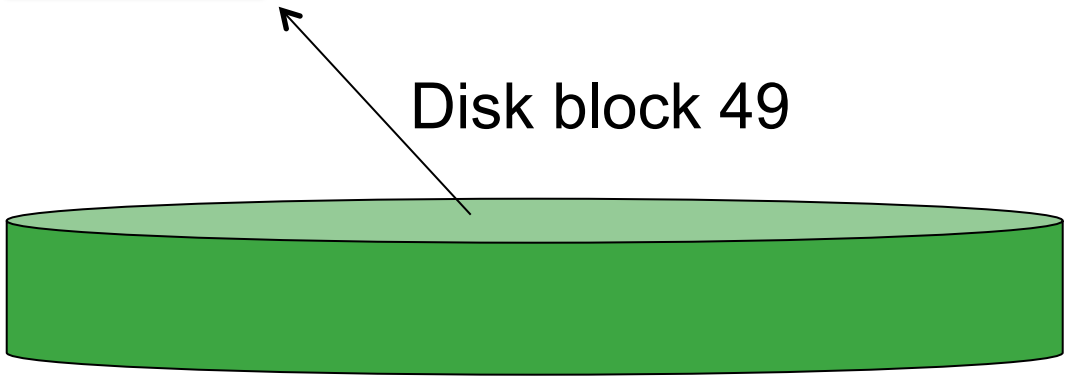
Solution

Data block 2 for /home/cs423

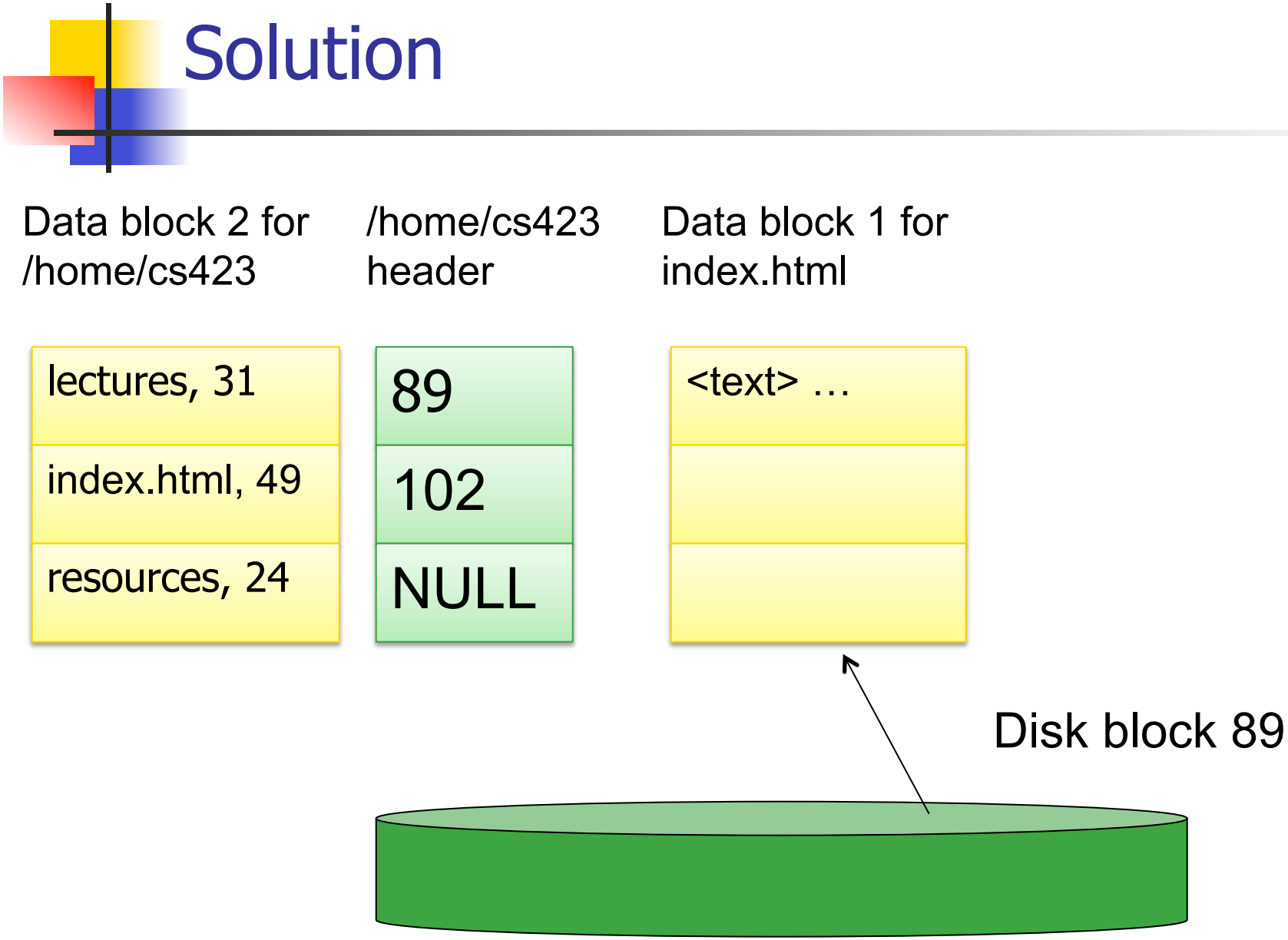
lectures, 31
index.html, 49
resources, 24

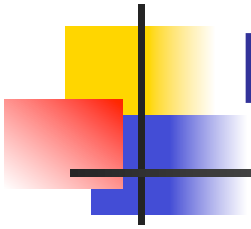
/home/cs423 header

89
102
NULL



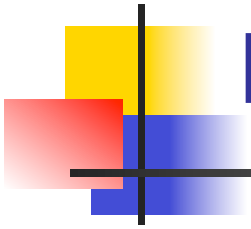
Solution


Data block 2 for /home/cs423
/home/cs423 header
Data block 1 for index.html
lectures, 31
index.html, 49
resources, 24
89
102
NULL
<text> …
Disk block 89



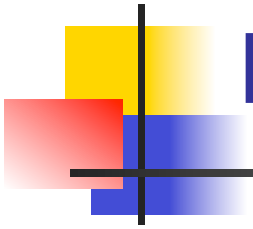
Directories

- 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



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 files system as permanent
- Multi-step updates cause problems if crash happens in the middle

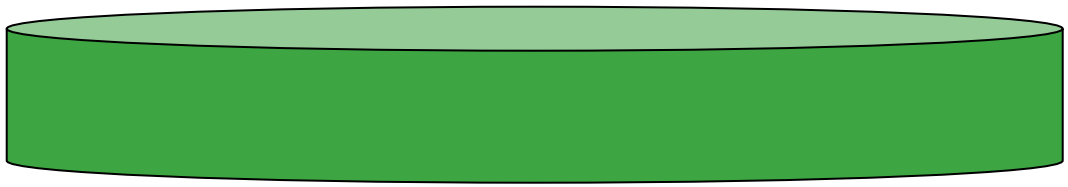


Multi-step updates and reliability

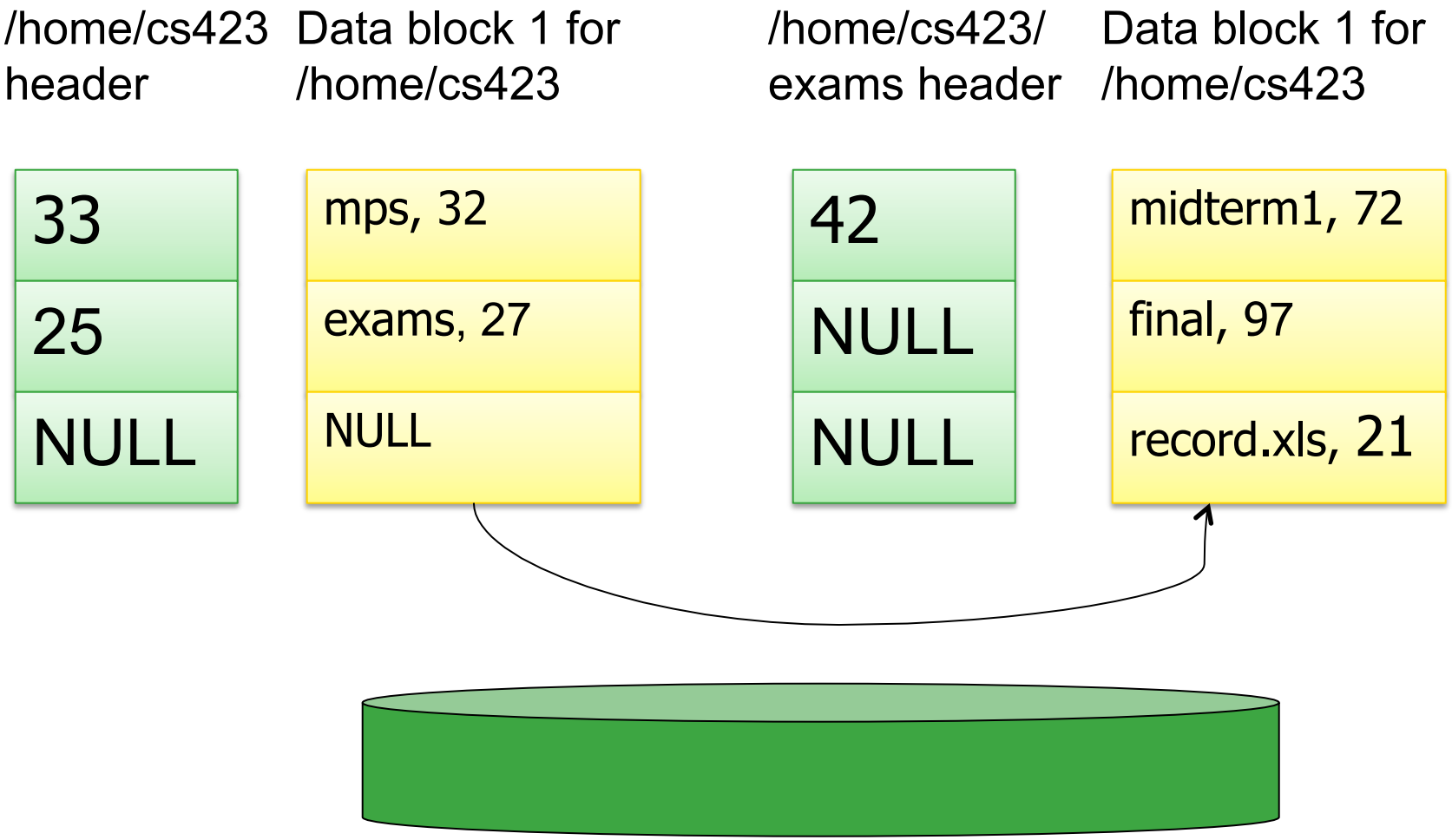
- Transfer$100 from my checking account my savings account
  1. Deduct $100 from my checking account
  2. Add $100 to savings account
- Move file from one directory to another
  1. Delete file from old directory
  2. Add file to new directory
- Create new file
  1. Update directory to point to new file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

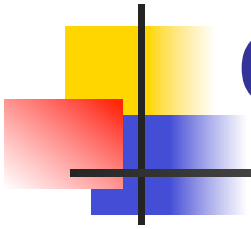
Moving a File: grades.xml-> exams/rec.xml

/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



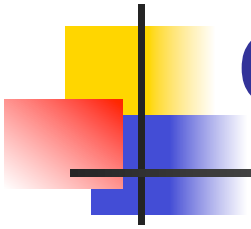
Moving a File: grades.xml-> exams/record.xml





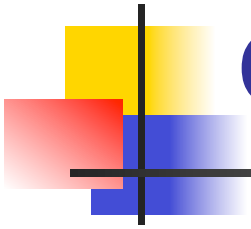
Careful Ordering

- 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?



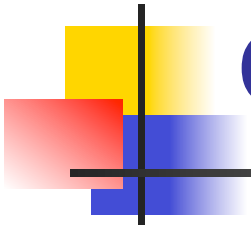
Careful Ordering

- 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)



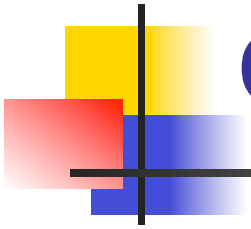
Careful Ordering

- 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



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!!!



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