CS 3204
Operating Systems

Lecture 34

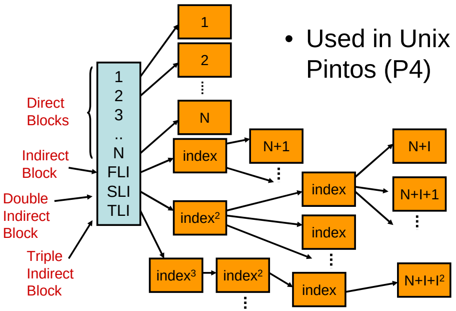
Godmar Back

Announcements

- Project 4 due Wed, May 3, 11:59pm
- Recommended reading
  - Chapter 11.1-11.5, 11A
  - Chapter 12, in particular 12.7
- Project 4 help sessions next Monday & Tuesday

Filesystems

Multi-Level Indices



- Used in Unix & Pintos (P4)

Storing Inodes

- Unix v7, BSD 4.3

Superblock	I_0 I_1 I_2 I_3 I_4	Rest of disk for files & directories

- FFS (BSD 4.4)

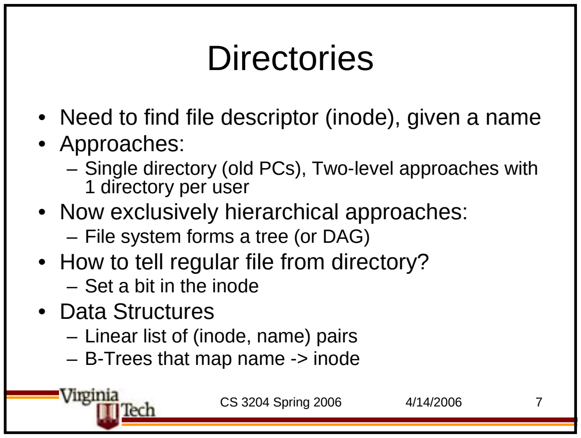
CG_i

SB1	I_0 I_1 …	Files …	SB2	I_3 I_4	Files …	SB3	I_8 I_9	Files …

- Cylindergroups have superblock+bitmap+inode list+file space
- Try to allocate file & inode in same cylinder group to improve access locality

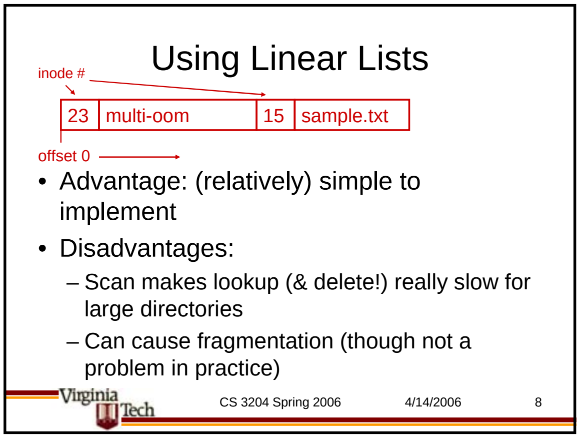
Positioning Inodes

- Putting inodes in fixed place makes finding inodes easier
  - Can refer to them simply by inode number
  - After crash, there is no ambiguity as to what are inodes vs. what are regular files
- Disadvantage: limits the number of files per filesystem at creation time
  - Use “df –ih” on Linux to see how many inodes are used/free



Directories

- Need to find file descriptor (inode), given a name
- Approaches:
  - Single directory (old PCs), Two-level approaches with 1 directory per user
- Now exclusively hierarchical approaches:
  - File system forms a tree (or DAG)
- How to tell regular file from directory?
  - Set a bit in the inode
- Data Structures
  - Linear list of (inode, name) pairs
  - B-Trees that map name -> inode



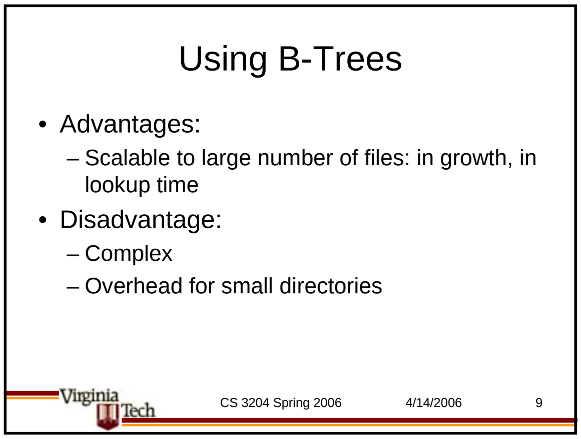
Using Linear Lists

inode #

23	multi-oom	15	sample.txt

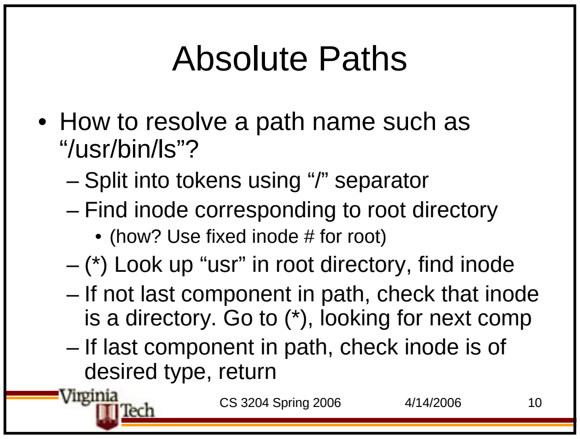
offset 0

- Advantage: (relatively) simple to implement
- Disadvantages:
  - Scan makes lookup (& delete!) really slow for large directories
  - Can cause fragmentation (though not a problem in practice)



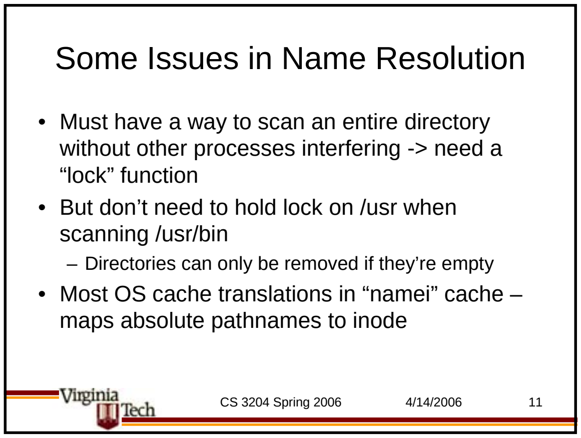
Using B-Trees

- Advantages:
  - Scalable to large number of files: in growth, in lookup time
- Disadvantage:
  - Complex
  - Overhead for small directories



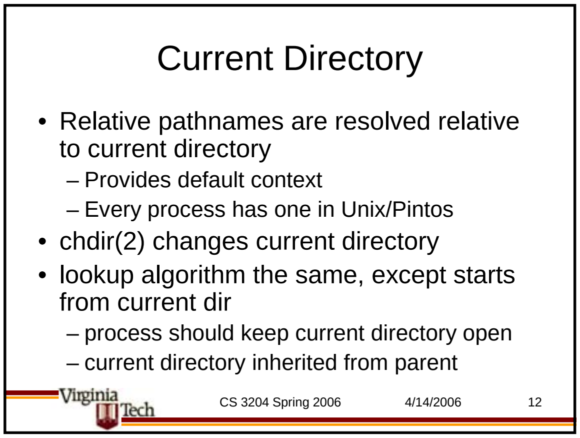
Absolute Paths

- How to resolve a path name such as “/usr/bin/ls”?
  - Split into tokens using “/” separator
  - Find inode corresponding to root directory
    - (how? Use fixed inode # for root)
  - (*) Look up “usr” in root directory, find inode
  - If not last component in path, check that inode is a directory. Go to (*), looking for next comp
  - If last component in path, check inode is of desired type, return



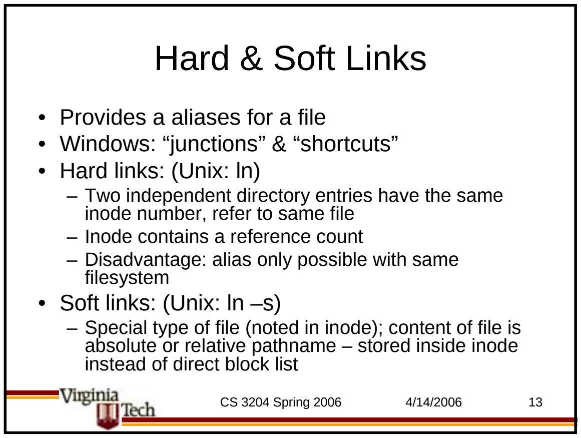
Some Issues in Name Resolution

- Must have a way to scan an entire directory without other processes interfering -> need a “lock” function
- But don’t need to hold lock on /usr when scanning /usr/bin
  - Directories can only be removed if they’re empty
- Most OS cache translations in “namei” cache – maps absolute pathnames to inode



Current Directory

- Relative pathnames are resolved relative to current directory
  - Provides default context
  - Every process has one in Unix/Pintos
- chdir(2) changes current directory
- lookup algorithm the same, except starts from current dir
  - process should keep current directory open
  - current directory inherited from parent



Hard & Soft Links

- Provides a aliases for a file
- Windows: “junctions” & “shortcuts”
- Hard links: (Unix: ln)
  - Two independent directory entries have the same inode number, refer to same file
  - Inode contains a reference count
  - Disadvantage: alias only possible with same filesystem
- Soft links: (Unix: ln –s)
  - Special type of file (noted in inode); content of file is absolute or relative pathname – stored inside inode instead of direct block list