Announcements

- Reading
  - Today Chapter 11 (8th ed) or 12 (6th ed)
  - Tuesday Chapter 12 (8th ed) or 13 (6th ed)
- Midterm #2 will be returned on Tuesday
- Project #5 is on the web
  - Deadline is Friday May 7th

Implementing Directories

- Linear List
  - array of names for files
  - must search entire list to find or allocate a filename
  - sorting can improve search performance, but adds complexity
- Hash table
  - use hash function to find filenames in directory
  - needs a good hash function
  - need to resolve collisions
  - must keep table small and expand on demand since many directories are mostly empty

Unix Directories

- Space for directories are allocated in units called *chunks*
  - Size of a chunk is chosen so that each allocation can be transferred to disk in a single operation
  - Chunks are broken into variable-length directory entries to allow filenames of arbitrary length
  - No directory entry can span more than one chunk
  - Directory entry contains
    - pointer to inode (file data-structure)
    - size of entry
    - length of filename contained in entry (up to 255)
    - remainder of entry is variable length - contains file name

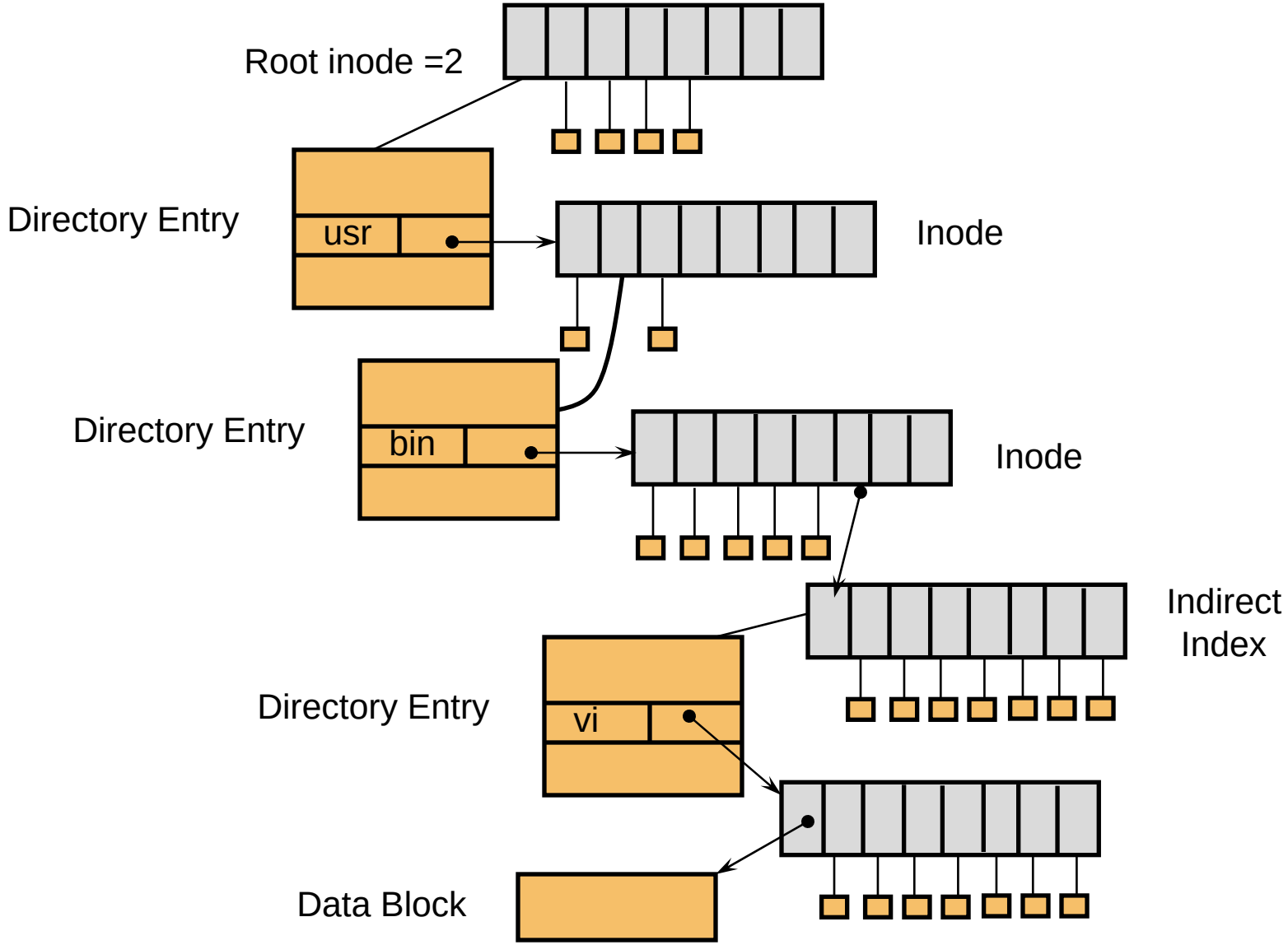
inodes

- File index node
- Contains:
  - Pointers to blocks in a file (direct, single indirect, double indirect, triple indirect)
  - Type and access mode
  - File's owner
  - Number of references to file
  - Size of file
  - Number of physical blocks

Unix directories - links

- Each file has unique inode but it may have multiple directory entries in the same filesystem to reference inode
- Each directory entry creates a hard link of a filename to the file's inode
  - Number of links to file are kept in reference count variable in inode
  - If links are removed, file is deleted when number of links becomes zero
- Symbolic or soft link
  - Implemented as a file that contains a pathname
  - Symbolic links do not have an effect on inode reference count

File Lookup (/usr/bin/vi)



Using UNIX filesystem data structures

- Example: find /usr/bin/vi
  - from Leffler, McKusick, Karels and Quarterman
  - Search root directory of filesystem to find /usr
    - root directory inode is, by convention, stored in inode #2
    - inode shows *where data blocks are* for root directory - *these blocks* (not the inode itself) *must* be retrieved and searched for entry user
    - we discover that the directory user's inode is inode #4
  - Search user for bin
    - access blocks pointed to by inode #4 and search contents of blocks for entry that gives us bin's inode
    - we discover that bin's inode is inode #7
  - Search bin for vi
    - access blocks pointed to by inode #7 and search contents of block for an entry that gives us vi's inode
    - we discover that vi's inode is inode #7
  - Access inode #7 - this is vi's inode

How to Improve Speed?

- Use A Cache
- Name-to-Inode lookup
  - Hash on full path name
  - Find inode without and disk accesses on a hit

Mount System Call

- How to attach a file system into a name space?
- Simple Idea:
  - use letters C, D, E, etc.
  - use volume names (VMS) – fixed length string
- Better Idea:
  - Allow attachment at arbitrary points in namespace
  - Designate one tree as the “root” file system
  - Others are attached to the root
- Mount used in:
  - UNIX
  - Windows (NTFS mount points)
  - GeekOS