File System (Interface)

Dave Eckhardt
de0u@andrew.cmu.edu

Synchronization

- How ***about*** that P3?
  - Checkpoint 2
- Today
  - Chapter 11, File system interface
    - Not: remote/distributed

What's a file?

- Abstraction of ***persistent storage***
  - Hide details of storage devices
    - sector addressing: CHS vs. LBA
    - SCSI vs. IDE
- ***Logical*** grouping of data
  - May be ***physically*** scattered
- Programs, data
- Some internal structure

File attributes

- Name – 14? 8.3? 255? 6-bit? Unicode?
- Identifier - “file number”
- Type (or not)
- Location – device, location
- Size – real or otherwise
- Protection – Who can do what?
- Time, date, last modifier – monitoring, curiousity

Operations on Files

- Create – locate space, enter into directory
- Write, Read – according to position pointer
- Seek – adjust position pointer
- Delete – remove from directory, release space
- Truncate
  - Trim data from end
  - Often all of it
- Append, Rename

Open-file State

- Expensive to specify name for each read()/write()
  - String-based operation
  - Directory look-up
- Open-file structure
  - File number
  - Read vs. write
  - Cursor position

Open files (Unix)

- “In-core” file state
  - Mirror of on-disk structure
  - Disk, location, size, permissions
- “Open-file” state
  - Cursor position, read vs. write
  - ***Shared*** by multiple processes
    - “copied” by fork()
    - inherited across exec()

Example

```
int fd1, fd2, fd3;
off_t pos2, pos3;
char buf[10];

fd1 = open(“foo.c”, O_RDONLY, 0);
fd2 = dup(fd1);
fd3 = open(“foo.c”, O_RDONLY, 0);
read(fd1, &buf, sizeof (buf));

pos2 = lseek(fd2, 0L, SEEK_CUR);/*10*/
pos3 = lseek(fd3, 0L, SEEK_CUR);/*0*/
```

File types (or not)

- Goal
  - Avoid printing a binary executable file
  - Find program which “understands” a file
- Filter file names
  - *.exe are executable, *.c are C
- Tag file
  - MacOS: 4-byte ***type***, 4-byte ***creator***
- Unix: Both/neither – Leave it (mostly) up to users

File Structure

- What's ***in*** a file?
  - Stream of bytes?
    - What character set? US-ASCII? Roman-1? Unicode?
  - Stream of records?
  - ***Array*** of records? ***Tree*** of records?
- Record structure?
  - End of “line”
    - CR, LF, CRLF
  - Fixed-length? Varying? Bounded?

File Structure - Unix

- OS ***needs to know*** about executables
  - “Magic numbers” in first two bytes
    - A.OUT OMAGIC, NMAGIC, ZMAGIC
    - ELF
    - #! script
- Otherwise, ***array of bytes***
  - User/application remembers meaning (hope!)
- Try the “file” command
  - Read /usr/share/magic

File Structure – MacOS

- Data fork
  - Array of bytes
  - Application-dependent structure
- Resource fork
  - Table of resources
    - Icon, Menu, Window, Dialog box
  - Many resources are widely used & understood
    - Desktop program displays icons from resource fork

Access Methods

- Provided by OS or program library
- Sequential
  - Like a tape
  - read() next, write() next, rewind()
  - Sometimes: skip forward/backward
- Direct/relative
  - Array of fixed-size records
  - Read/write any record, by #

Access Methods – Indexed

- File contains ***records***
- Records contain ***keys***
- ***Index*** maps keys -> records
  - Sort data portion by key
  - Binary search in multi-level list
- Fancy extensions
  - Multiple keys, multiple indices
  - Are we having a database yet?

Disk data structures (Intro)

- Split disk into ***partitions***/slices/minidisks/...
  - Or: glue disks together into ***volumes***/logical disks
- Partition may contain...
  - Paging area
    - Indexed by memory structures
    - “random garbage” when OS shuts down
  - File system
    - Block allocation: file # -> block list
    - Directory: name -> file #

Directory Operations

- Lookup(“index.html”)
- Create(“index.html”)
- Delete(“index.html”)
- Rename(“index.html”, “index.html~”);
- Iterate over directory contents
- Scan file system
  - Unix “find” command
  - Backup program

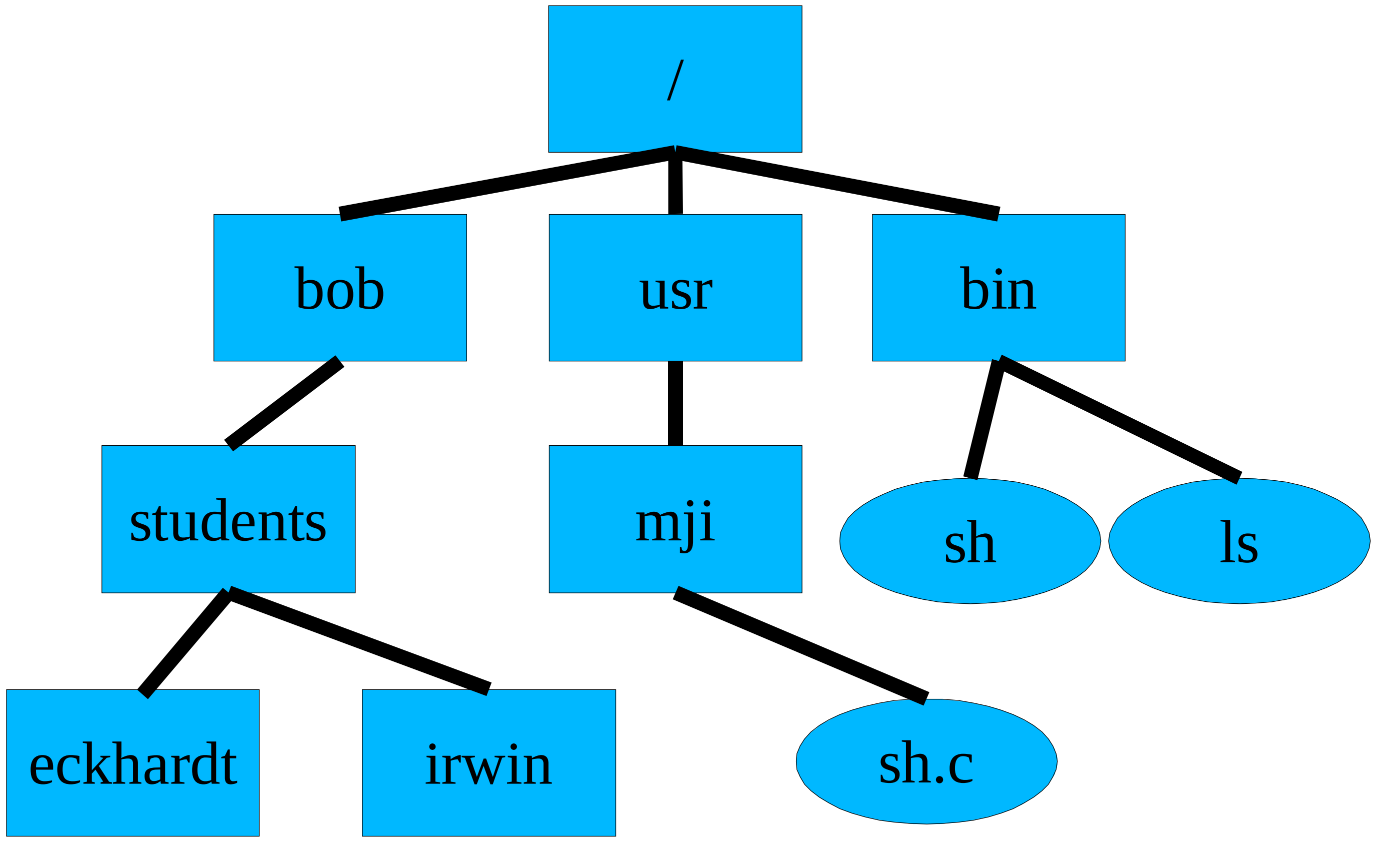
Directory Types

- Single-level
  - Flat global namespace – only ***one*** test.c
  - Ok for floppy disks (maybe)
- Two-level
  - Every user has a directory
  - One test.c ***per user***
  - Typical of early timesharing
- Are we having fun yet?

Tree Directories

- ***Absolute*** Pathname
  - Sequence of directory names
  - Starting from "root"
  - Ending with a file name

Tree Directories

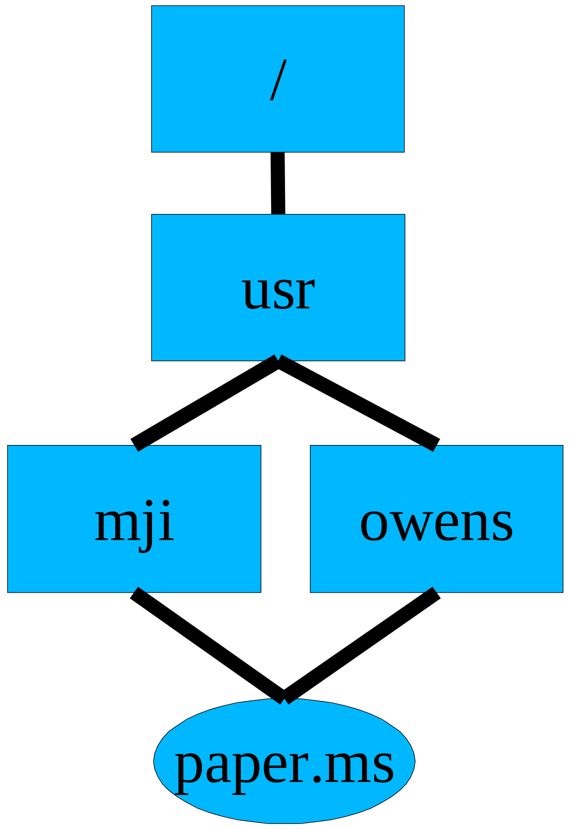


Tree Directories

- Directories are special files
  - Created with special system calls – mkdir()
  - Format understood, maintained by OS
- Current directory (".")
  - "Where I am now"
  - Start of ***relative*** pathname
    - ./stuff/foo.c aka stuff/foo.c
    - ../joe/foo.c aka /usr/joe/foo.c

DAG Directories

- Share files and directories between users
- Not mine, not yours: ***ours***
- Destroy when ***everybody*** deletes
- Unix "hard link"
  - For files (".. problem")



Soft links

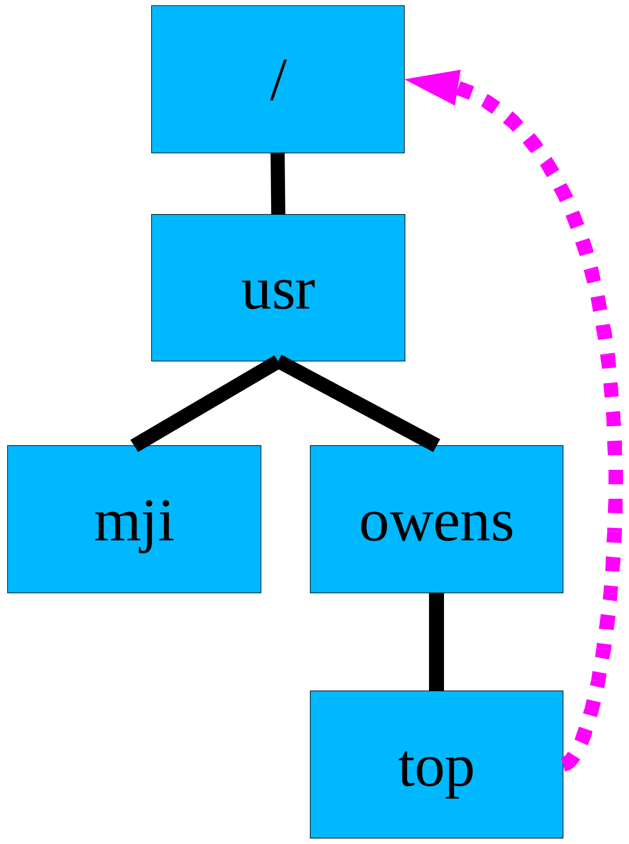
- Hard links “too hard”?
  - Level of indirection in file system
  - No “one true name” for a file
  - NIH syndrome?
- Soft link / symbolic link / “short cut”
  - Tiny file, special type
  - Contains ***name*** of another file
  - OS dereferences link when you open() it

Hard vs. Soft Links

- Hard links
  - Enable reference-counted sharing
  - No name is better than another
  - Dangerous to allow hard links to directories
- Soft links
  - Work across file system & machine boundaries
  - Easier to explain
  - “Dangling link” problem

Graph Directories

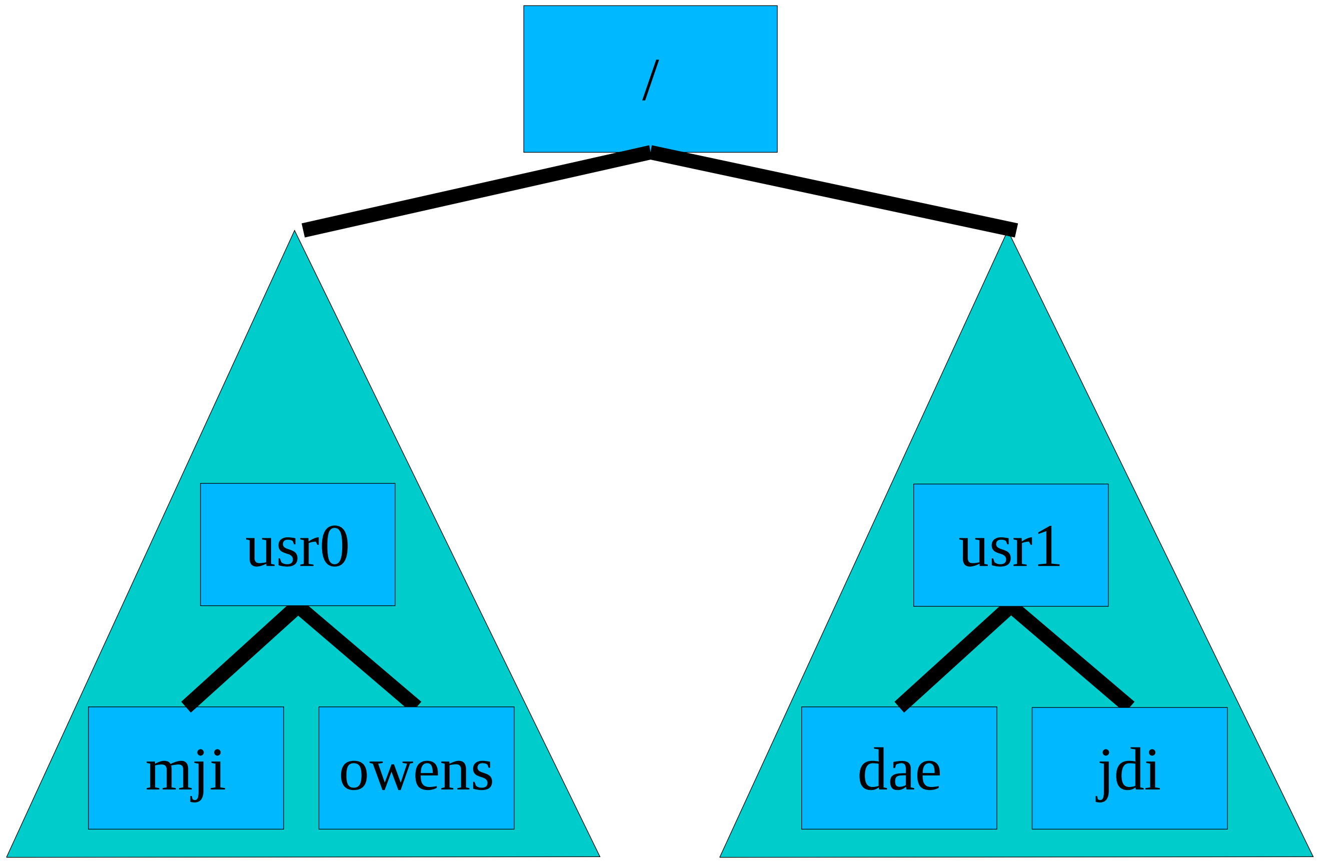
- “find” can be slow!
- Need ***real*** garbage collection
- Do we really need this?



Mounting

- Multiple disks on machine
- Multiple partitions on disk
- File system ***within*** a partition
  - Or, within a volume / logical volume / ...
- How to name files in “another” file system?
  - Wrong way
    - C:\temp vs. D:\temp
    - [1003,221]PROFILE.CMD vs. [1207,438]PROFILE.CMD

Mounting



Multiple Users

- Users want to share files
- What's a user?
  - Strings can be cumbersome
  - Integers are nicer
- User ID / “uid” (Unix), Security ID / “sid” (Win)
- What's a group?
  - A set of users
  - May have its own gid / sid

Protection

- Override bit (e.g., MS-DOG)
  - Bit says “don't delete this file”
    - Unless I clear the bit
- Per-file passwords
  - Annoying in a hurry
- Per-directory passwords
  - Still annoying

Protection

- Access modes
  - Read, Write, Execute, Append, Delete, List, Lock, ...
- Access Control List (ACL)
  - File stores list of (user, modes) tuples
  - Cumbersome to store, view, manage
- Capability system
  - User given list of (file, access keys) tuples
  - Revocation problem

Protection – typical

- File specifies ***owner, group***
  - Permissions for each
    - Read, write, ...
  - Permissions for “other” / “world”
    - Read, write, ...
- Unix
  - r, w, x = 4, 2, 1
  - rwxr-x—x = 0751 (octal)
  - 3 32-bit words specifies everything

Summary

- File
  - Abstraction of disk/tape storage
    - Records, not sectors
    - Type information
  - Naming
    - Complexity due to linking
  - Ownership, permissions
  - Semantics of multiple open()s
- More in 20.7, 20.8