CS 140 - Summer 2008 - Handout #17: Directories

The Next Big Adventure

- Disk contains billions and billions of messy "things"
  - **How to organize (name) them?!?**
  - **Taxonomies! (In OS-speak: "Hierarchical name spaces")**
  - **Mechanics: directories that can point to directories**

  Silberschatz:
  - **7th ed: ch. 10 & 11;**
  - **6th ed: ch. 11 & 12.**

  Unix implementation paper

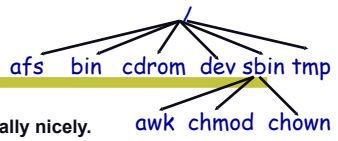
Directories

- Problem: users need a way to get back to file after turning off computer and going home.
- Approach 0: have user remember where on disk the file is.
  - **(e.g., social security numbers)**
- Yuck. People want human digestible names
  - **we use directories to map names to file blocks**
- Today: What is in a directory and why?

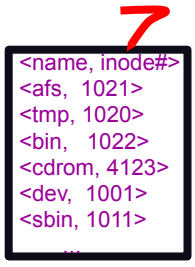
A brief history of time

- Approach 1: have a single directory for entire system.
  - **Put directory at known location on disk.**
  - **Directory contains <name, index> pairs.**
  - **If one user uses a name, no one else can.**
  - **Many older personal computers work this way.**
- Approach 2: have a single directory for each user
  - **Still clumsy. And `ls` on 10,000 files is a real pain.**
- Approach 3: hierarchical name spaces
  - **Allow directory to map names to files or other dirs.**
  - **File system forms a tree (or graph, if links allowed).**
  - **Large name spaces tend to be hierarchical (ip addresses, domain names, scoping in programming languages, etc.)**

Hierarchical Unix



- Used since CTSS (1960s)
  - **Unix picked up and used really nicely.**
- Directories stored on disk just like regular files
  - **inode contains special flag bit set.**
  - **User's can read just like any other file.**
  - **Only special programs can write (why?).**
  - **File pointed to by the index may be another directory.**
  - **Makes FS into hierarchical tree (what needed to make a DAG?)**

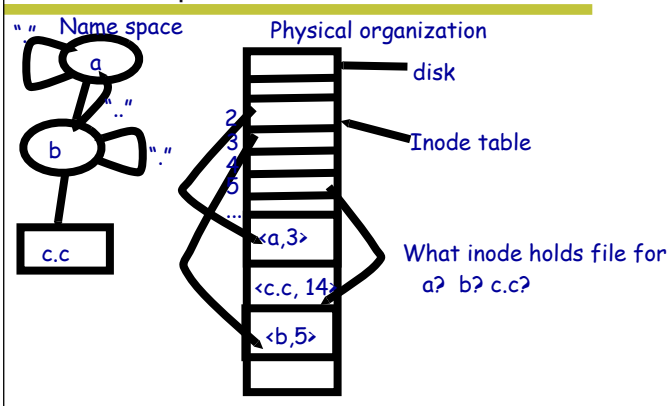


- Simple. Plus speeding up file ops = speeding up dir ops!

Naming magic

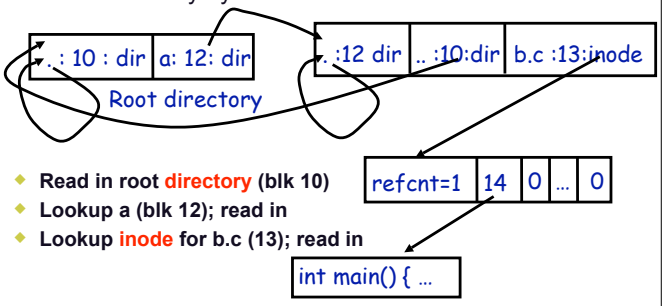
- Bootstrapping: Where do you start looking?
  - **Root directory.**
  - **inode #2 on the system.**
  - **0 and 1 used for other purposes.**
- Special names:
  - **Root directory: "/" (bootstrap name system for users)**
  - **Current directory: "."**
  - **Parent directory: ".." (otherwise how to go up??)**
  - **user's home directory: "~" (shell rather than OS in Unix)**
- Using the given names, only need two operations to navigate the entire name space:
  - **cd 'name': move into (change context to) directory "name"**
  - **ls : enumerate all names in current directory (context)**

Unix example: /a/b/c.c



Example: Unix file system

- Want to modify byte 4 in /a/b.c:



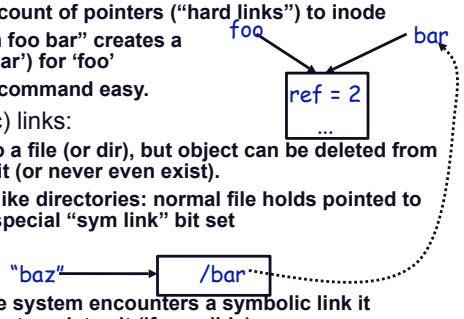
- **Read in root directory (blk 10)**
- **Lookup a (blk 12); read in**
- **Lookup inode for b.c (13); read in**
- **Use inode to find blk for byte 4 (blksize = 512, so offset = 0 gives blk 14); read in and modify**

Default context: working directory

- Cumbersome to constantly specify full path names
  - **In Unix, each process associated with a "current working directory".**
  - **File names that do not begin with "/" are assumed to be relative to the working directory, otherwise translation happens as before.**
- Shells track a default list of active contexts
  - **A "search path".**
  - **Given a search path { A, B, C } a shell will check in A, then check in B, then check in C**
  - **Can escape using explicit paths: "./foo"**
- Example of locality

Creating synonyms: Hard and soft links

- More than one dir entry can refer to a given file
  - **Unix stores count of pointers ("hard links") to inode**
  - **To make: "ln foo bar" creates a synonym ('bar') for 'foo'**
  - **Makes `mv` command easy.**
- Soft (symbolic) links:
  - **Also point to a file (or dir), but object can be deleted from underneath it (or never even exist).**
  - **Unix builds like directories: normal file holds pointed to name, with special "sym link" bit set**



  - **When the file system encounters a symbolic link it automatically translates it (if possible).**

File contents

- A file's inode holds a bunch of stuff (metadata):
  - **file size, access times, owner and group id, protection bits**
- example: "ls -il /home/alice/cs140/quiz"
  - **12445 -rw-r--r-- 1 alice cs140 2918 Oct 3 3:15 quiz**
- example:
  - **"ln /usr/alice/cs140/quiz1/Q1 quiz"**
  - **"ls -il /usr/alice/cs140/Q1"**
  - **12445 -rw-r--r-- 2 alice cs140 2918 Oct 3 3:15 Q1**