CS 537 Notes, Section #25: Directories

Motivation

Users need a way of finding the files that they created on disk. One approach is just to have users remember descriptor indexes.

Of course, users want to use text names to refer to files. Special disk structures called *directories* are used to tell what descriptor indices correspond to what names.

A hard concept to understand at the beginning: naming is one of the (if not *the*) most important issues in systems design.

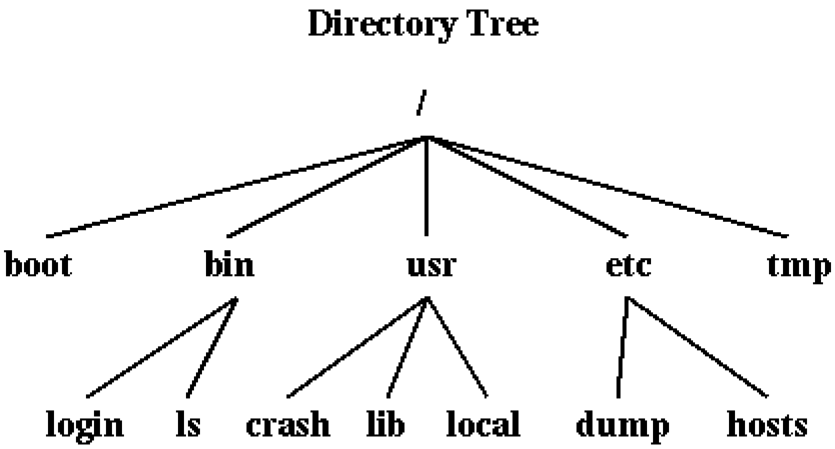
Approach #1: have a single directory for the whole disk. Use a special area of disk to hold the directory.

- Directory contains pairs.
- If one user uses a name, no-one else can.

Approach #2: have a separate directory for each user (TOPS-10 approach). This is still clumsy: names from different projects get confused.

Unix Directories

Unix approach: generalize the directory structure to a tree.



- Directories are stored on disk just like regular files (i.e. file descriptor with 13 pointers, etc.). User programs can read directories just like any other file (try it!). Only special system programs may write directories.

- Each directory contains pairs. The file pointed to by the index may be another directory. Hence, get hierarchical tree structure, name with /usr/local.
- There is one special directory, called the *root*. This directory has no name, and is the file pointed to by descriptor 2 (descriptors 0 and 1 have other special purposes).

It is very nice that directories and file descriptors are separate, and the directories are implemented just like files. This simplifies the implementation and management of the structure (can write "normal" programs to manipulate them as files).

Working directory: it is cumbersome constantly to have to specify the full path name for all files.

- In Unix, there is one directory per process, called the working directory, that the system remembers.
- When it gets a file name, it assumes that the file is in the working directory. "/" is an escape to allow full path names.
- Many systems allow more than one current directory. For example, check first in A, then in B, then in C. This set of directories is called the *search path* or *search list*. This is very convenient when working on large systems with many different programmers in different areas.
- For example, in Unix the shell will automatically check in several places for programs. However, this is built into the shell, not into Unix, so if any other program wants to do the same, it has to rebuild the facilities from scratch. Should be in the OS.
- This is yet another example of locality.