Topic 14: File Naming — Directories

- *Naming*: How do users refer to their files? How does the OS find a file, given its name?
- The file record has to be stored on disk, so it will stay around even when the OS does not.
  - Unix: All the records (*inodes*) are stored in a fixed-size array on disk. An inode's index in the array is its *i-number*, and is used by the OS to refer to the file.
  - Special areas of the disk are used for this array:
    - Originally: the inode array was at one end (edge or center) of the disk.
    - Then: inode array was put mid-way across the disk.
    - Today: there is an inode array in each cylinder group.
  - The sizes of the inode arrays are determined when the disk is initialized.
    - Can not be changed, unless you reformat the disk!
- Users need a way of getting back to files that they leave around on disk. One approach is just to have users remember the i-numbers for their files…

- Of course, users want to use descriptive names to refer to files. Special disk structures called *directories* are used to tell what i-numbers correspond to what names. The conversion of a file name into the i-number for the file is known as *name lookup*.

- Approach #1:
  - Have a single directory for the whole disk. Use a special area of the disk to hold the directory.
    - Directory contains <name, index> pairs.
    - If one user uses a name, no one else can.

- Approach #2:
  - Have a separate directory for each user.
  - This is still clumsy: names from a user's different projects get confused.

- Approach #3: (Unix) Generalize the directory structure to a tree.
  - Directories are stored on disk just like regular files (i.e., inode with 15 pointers, etc.) except the inode has a special flag bit set. User programs can read directories just like any other file (try it!). Only special system programs may write directories.
  - Each directory contains <name, i-number> pairs in no particular order. Each entry is variable-sized, and contains the name length, the name (up to 255 characters), and the i-number. Entries do not cross block boundaries.
  - The entries are searched in a linear fashion until the desired name is found. This can be very slow if there are a lot of entries.
  - The i-number in an entry may refer to another directory, creating a directory graph (usually a tree).
  - Every directory has the following two entries:
    - ".": refers to itself.
    - "..": refers to its parent.
  - There is one special directory, called the *root*. This directory has no name, but always uses a particular inode. The ".." entry in the root directory points to itself.

- Sometimes, it is useful for a single file to have multiple names. There are two ways to do this in Unix:
  - *Hard link*: Multiple directory entries refer to a single file. Unix uses reference counts in the inodes to keep track of the directory entries. The file is not deleted until the last directory entry goes away (reference count is now zero).
    - What if the source of the link is a directory?
      - Directory hierarchy is a directed graph, not a tree.
      - Should be a directed acyclic graph (DAG), or there will be trouble.
      - This is enforced in Unix by allowing only the super-user to make hard links to directories.
    - Where does ".." point to in source?
  - *Symbolic link*: A file that contains another file name. Flag bit in the inode tells the OS to read the file name from symbolic link file and open that file, rather than the symbolic link file.
    - Symbolic link will be left dangling if the source file is renamed.
    - Requires two name lookups.
  - Hard links are not allowed across devices. Soft links are allowed.

- Sometimes, it is useful for a file to have no name. Programs sometimes use temporary files whose names are unimportant.
  - Unix: A program can create a file, open it, then delete it. The file is still open, but it has no name.
    - This may also happen if a process deletes a file that another process has open.
    - Other OS's may consider this an error.
  - Although the reference count on the file is 0, the OS must refrain from deleting the file until it is closed.
  - What happens if there is a crash?
    - During consistency check after a reboot, put all such files into the *lost+found* directory.

- Working directory (pwd, cwd): it is cumbersome to constantly have to specify the full path name for all files.
  - Unix: Each process has a current directory, called its *working directory*.
  - When the OS gets a file name, it assumes that the file is in the working directory, unless the name starts with "/". A name starting with "/" is called *absolute*, otherwise it is *relative*.

- Shells often implement search *paths*, a list of directories to be searched for the given file name.
  - Examples are *path* and *cdpath* variables in *csh* (try the **`printenv`** command).