Directories (Topic 10)

홍 성 수

서울대학교 공과대학 전기 공학부
Real-Time Operating Systems Laboratory

Directories (1)

- Readings for this topic:
  - Silberschatz/Galvin: Chapter 10.3, 11.4, 19.7
- Naming:
  - How do users refer to their files?
  - How does OS find a file, a given name?
- The file descriptor information has to be stored on disk, so it will stay around even when the OS doesn't.
  - In UNIX, all the descriptors are stored in a fixed-size array on disk. The descriptors also contain protection and accounting information.

Directories (2)

- Special areas of disk are used for this. Show picture.
  Originally: Descriptor array at one side of disk.
  Then: Descriptor array mid-way across disk.
  Today: Many small descriptor arrays spread across disk, so descriptors can be near to file data.
- The sizes of the descriptor arrays are determined when the disk is initialized, and can't be changed. In Unix, the descriptor is called an i-node, and its index in the array is called its i-number, Internally, the OS uses the i-number to refer to the file.

Directories (2)

- When a file is open, it descriptor is kept in main memory. When the file is closed, the descriptor is stored back to disk.
- Users need a way of getting back to files that they leave around on disk. One approach is just to have users remember descriptor indexes.
  Example: Social Security numbers.
- 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.

Directories (3)

- Approach #1: Have a single directory for the whole disk. Use a special area of disk to hold the directory.
  - Directory contains <name, index> pairs.
  - If one users uses a name, no-one else can.
  - Many personal computers work this way.
- Approach #2: have a separate directory for each user (TOPS-10 approach).
  - This is still clumsy: Name a user's different projects get confused.

Directories (4)

- Unix approach: Generalize the directory structure to a tree.
  - Directories are stored on disk just like regular files (i.e. file descriptor with 14 pointers, etc.) except file descriptor has special flag bit set. User programs can read directories just like any other file.

    Only special system programs may write directories.
  - Each directory contains <name, fd index> pairs in no particular order. The file pointed to by the index may be another directory. Hence, get hierarchical tree structure.

    Names have slashes separating the levels of the tree.

Directories (5)

  - There is one special directory. called the root. This directory has no name, and is the rile pointed to by descriptor 2 (descriptors 0 and 1 have other special purposes).
- Example: /a/b/c
  - Inode 2: Contains < "a", 5 >
  - Inode 5: Contains < "b", 7 >
  - Inode 7: Contains < "c",14 >
  - Inode 14: File c.

Directories (6)

- It is possible for more than one directory entry to refer to a single file ("hard links"). UNIX uses reference counts in the file descriptors to keep track of the directory entries, only delete file when last directory entry goes away.
- Other things kept in UNIX file descriptors:
  - File size.
  - Access times.
  - Owner and group id.
  - Protection bits.

Directories (7)

- Example: In /usr/class/cs240a/code/hw1 do a "ls –lig list.cc"
- 122945 –rw-rw-r–1 jbenett cs240a 2918 Oct 3 15:17 list.cc
- It is very nice that directories and file descriptors are separate, and that directories are implemented just like files. This simplifies the implementation and management of the structure (can write "normal" programs to manipulate them as files).
  - Drawback: Distributed environments.

Directories (8)

- Working directory: It is cumbersome constantly to have to specify the full path same for all files.
  - In Unix, there is one directory per process, called the working directory, which 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.

Directories (9)

  - 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.

    Another example is " ".
  - This is yet another example of locality.
- Symbolic links: A file whose contents are just another file name. Also stored on disk just like regular files, but with a special flag set in descriptor.