COS 222

Lecture 34:

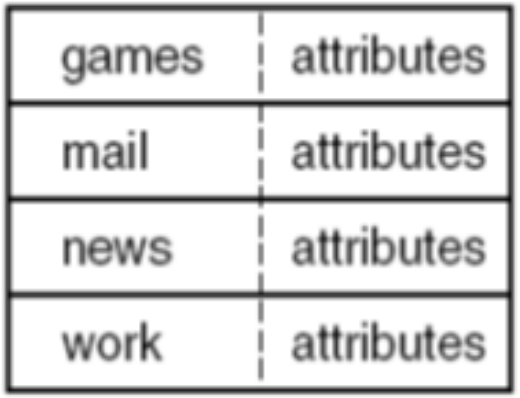
Directories

Directories

- To keep track of files, file systems normally have **directories** or **folders**, which, in many systems, are themselves files.
- In this Lecture we will discuss directories, their organization, their properties, and the operations that can be performed on them.

Simple Directories

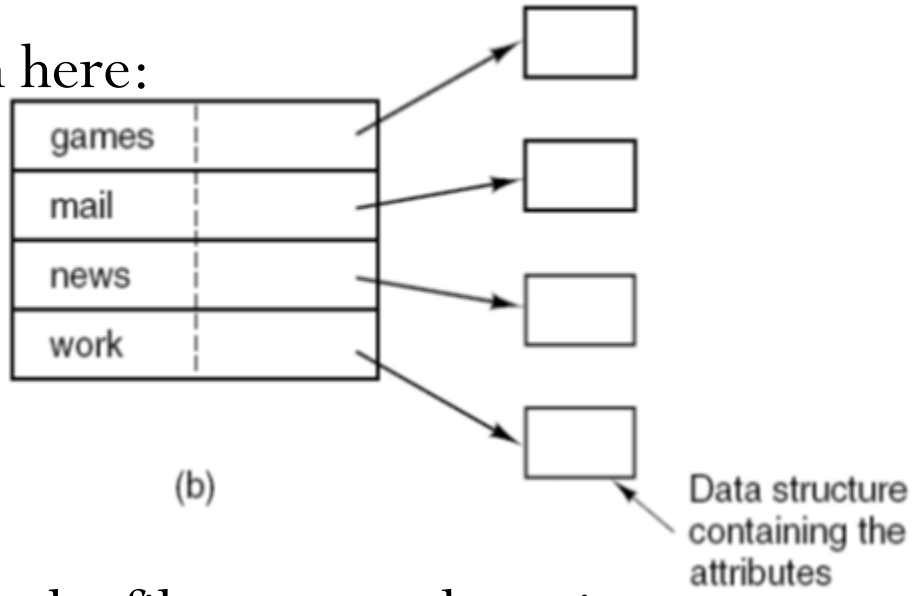
- A directory typically contains a number of entries, one per file.
- One possibility is shown here:



- In which each entry contains the file name, the file attributes, and the disk addresses where the data are stored.

Simple Directories

- Another possibility is shown here:



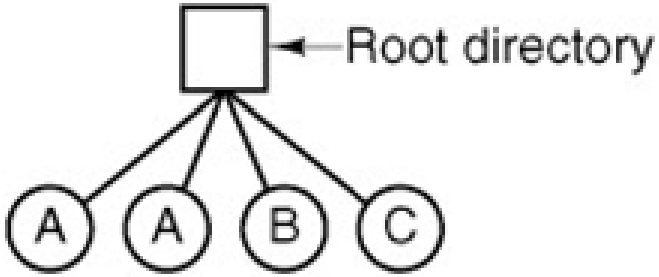
- Here a directory entry holds the file name and a pointer to another data structure where the attributes and disk addresses are found.
- Both of these systems are commonly used.

Simple Directories

- When a file is opened,
  - the operating system searches its directory until it finds the name of the file to be opened.
  - It then extracts the attributes and disk addresses, either directly from the directory entry or from the data structure pointed to, and puts them in a table in main memory.
  - All subsequent references to the file use the information in main memory.

Simple Directories

- The number of directories varies from system to system.
- The simplest form of directory system is a single directory containing all files for all users, as illustrated below



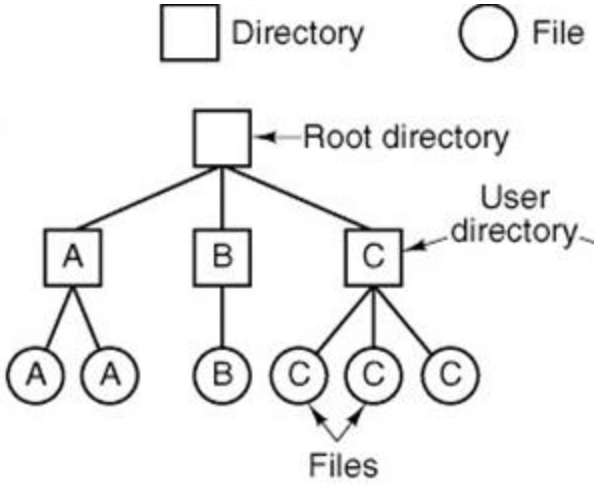
- On early personal computers, this single-directory system was common, in part because there was only one user.

Simple Directories

- The problem with having only one directory in a system with multiple users is that different users may accidentally use the same names for their files.
  - For example, if user *A* creates a file called *mailbox*, and then later user *B* also creates a file called *mailbox*, *B*'s file will overwrite *A*'s file.
- Consequently, this scheme is not used on multiuser systems any more,
  - but could be used on a small embedded system, for example, a handheld personal digital assistant or a cellular telephone.

Simple Directories

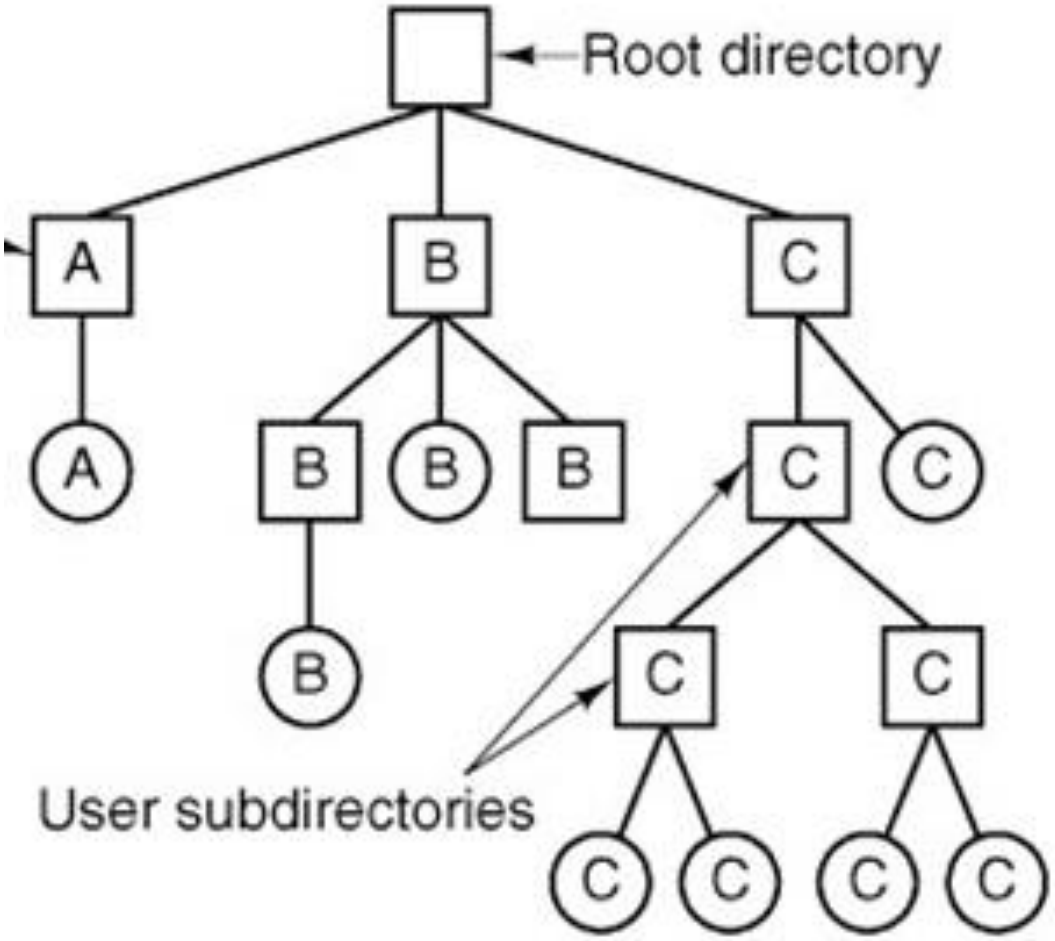
- To avoid conflicts caused by different users choosing the same file name for their own files,
  - the obvious next step up is giving each user a private directory.
  - In that way, names chosen by one user do not interfere with names chosen by a different user and there is no problem caused by the same name occurring in two or more directories



Hierarchical Directory Systems

- The two-level hierarchy eliminates file name conflicts between users.
- But another problem is that users with many files may want to group them in smaller subgroups.
  - E.g Music, Movies, Work….
- With this approach, each user can have as many directories as are needed so that files can be grouped together in natural ways.

Hierarchical Directory Systems



Hierarchical Directory Systems

- The ability to create an arbitrary number of subdirectories provides a powerful structuring tool for users to organize their work.
- For this reason nearly all modern PC and server file systems are organized this way.

Path Names

- When the file system is organized as a directory tree, some way is needed for specifying file names.
- Two different methods are commonly used.
  - In the first method, each file is given an **absolute path name** consisting of the path from the root directory to the file.
    - Absolute path names always start at the root directory and are unique
  - The other kind of name is the **relative path name.**
    - This is used in conjunction with the concept of the **working directory.**
    - A user can designate one directory as the current working directory, in which case all path names not beginning at the root directory are taken relative to the working directory.

Path Names

- Each process has its own working directory, so when a process changes its working directory and later exits, no other processes are affected and no traces of the change are left behind in the file system.
  - In this way it is always perfectly safe for a process to change its working directory whenever that is convenient.
- On the other hand, if a *library procedure* changes the working directory
  - and does not change back to where it was when it is finished, the rest of the program may not work
    - since its assumption about where it is may now suddenly be invalid.
  - For this reason, library procedures rarely change the working directory, and when they must, they always change it back again before returning.

Path Names

- Most operating systems that support a hierarchical directory system have two special entries in every directory,
  - "." and
  - ".."
- Dot refers to the current directory; dotdot refers to its parent.

Directory Operations

- The system calls for managing directories exhibit more variation from system to system than system calls for files.
- ***Create***.
  - A directory is created.
  - It is empty except for dot and dotdot, which are put there automatically by the system.
- ***Delete***.
  - A directory is deleted.
  - Only an empty directory can be deleted.
  - A directory containing only dot and dotdot is considered empty as these cannot usually be deleted.

Directory Operations

- ***Opendir***.
  - Directories can be read. For example, to list all the files in a directory,
    - a listing program opens the directory to read out the names of all the files it contains.
- ***Closedir***.
  - When a directory has been read, it should be closed to free up internal table space.
- ***Readdir***.
  - This call returns the next entry in an open directory.

Directory Operations

- ***Rename***.
  - In many respects, directories are just like files and can be renamed the same way files can be.
- ***Link***.
  - Linking is a technique that allows a file to appear in more than one directory.
  - This system call specifies an existing file and a path name, and creates a link from the existing file to the name specified by the path.
  - In this way, the same file may appear in multiple directories.
  - A link of this kind, which increments the counter in the file's i-node, is sometimes called a **hard link**.

Directory Operations

- ***Unlink***.
  - A directory entry is removed.
  - If the file being unlinked is only present in one directory (the normal case), it is removed from the file system.
  - If it is present in multiple directories, only the path name specified is removed. The others remain.