File System Interface

- Ch13.1: File Concept
- Ch13.2: Access Methods, File System Operations
- Ch13.3: Directory Structure, File Sharing, Hard Links, Soft Links

What is a File?

- Textbook (Ch13,pg530): A named collection of related information that is recorded on secondary storage.
- The information is related in a logical sense that
  - It is used by a particular program
  - It is a particular program
- Secondary storage is usually a disk drive (including SSDs), a tape drive, or any other non-volatile device.
- In some systems “files” are not always files e.g. In UNIX devices are “files” and so are some operating system data structures (e.g. /proc in Linux)

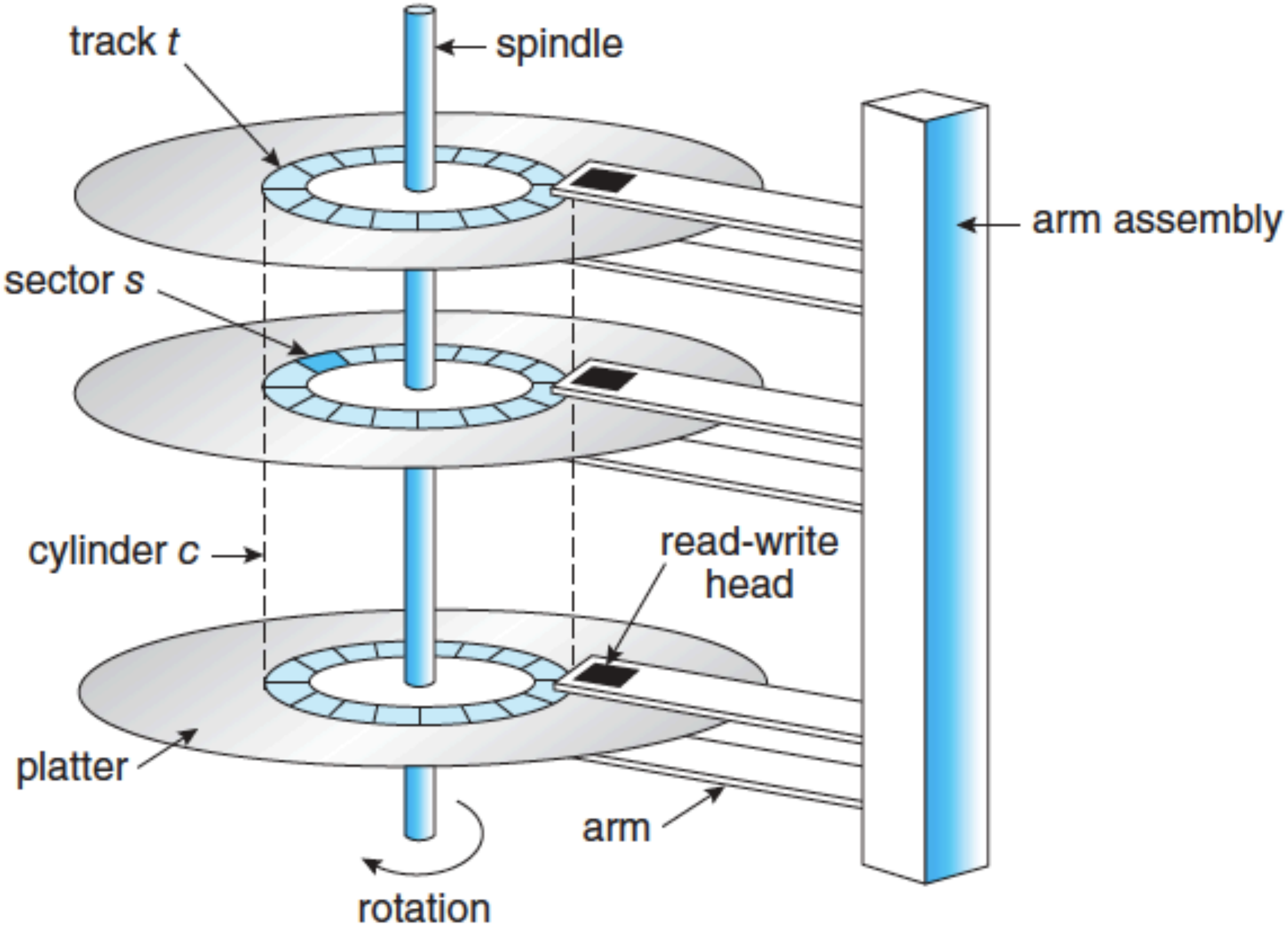
```
ls -l /proc
```

File Systems

A file system needs to satisfy these general requirements.

- **We need some way of storing information**
  - Independently from a running program, so it can be used at later time
  - Permanent / Persistence
  - Non-volatile storage, so it can be shared with other programs or users
- **An infinite variety of data is to be stored**
  - The more information the OS knows about the data the more it can facilitate use of the data. E.g. binary files, textual files
- **Naming the data**
  - The data needs to be stored and retrieved easily. We need a way to name the data.
  - We must then be able to locate the data using its name.
  - User can easily find, examine, modify

Hard Disk Drive (HDD)



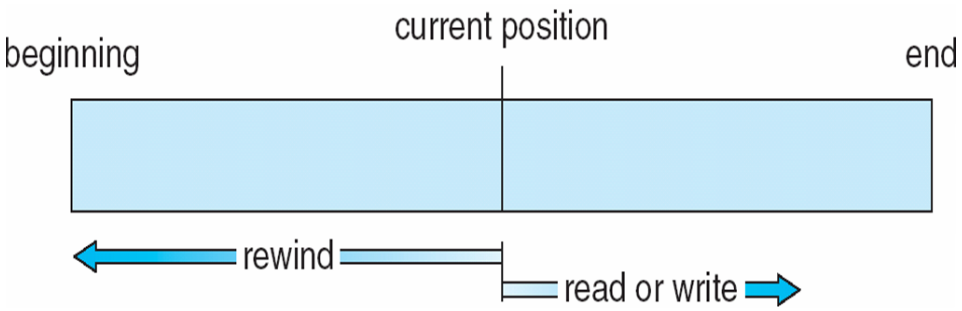
Moving Head Disk Mechanism (Ch11.1 Mass-Storage Structure)

How Disks Work?

- **Overhead**: the time the CPU takes to start a disk operation
- **Positioning Time**: the time to initiate a disk transfer of 1 byte to memory
  - **Seek time**: the time to position the head over the correct cylinder
  - **Rotational time**: the time for the correct sector to rotate under the head
- **Transfer Rate**: once a transfer is initiated, the rate of I/O transfer (bandwidth)

Access Methods

- Sequential Access
  - Information in the file is processed in order, one record after the other.
    - read_next(): reads the next portion of the file and automatically advances a file pointer
    - write_next(): appends to the end of the file and advances to the end of the newly written material



- Direct Access
  - File is made up of fixed-length logical records that allow programs to read and write records rapidly in no particular order.
  - File is viewed as a numbered sequence of blocks or record
    - read(n) / write(n): read / write block n, where n is the block number

File System Operations

On most systems these commands need security authorisation to perform and they work on the file as a whole.

Create

- Space in the file system must be found for the file
- Need to specify the name, the file type
- Create a file descriptor for the file including name, location on disk, and all file attributes
- Specify the size of the file
- Creation needs to do something to the associated device - an entry for the new file must be made in a directory.
- Some systems allow transitory files to not be recorded permanently in secondary storage.

File System Operations

Copy

- Creating a new file and then reading from the old and writing to the new.
- Most file systems preserve attributes (including last modified times) when a copy is made. This way a file can be last modified before it was created.
- The file information needs to point to the original data.

Change attributes

- We will see the different sorts of attributes shortly. Some of these should be changeable, others should be secured.

```
(base) [mfchiang@ my_solution]$ stat source/one
16777220 12888855230 -rw-r--r-- 1 mfchiang staff 0 2 "Sep 10 18:33:00 2020" "Sep 10 18:32:47 2020"
"Sep 10 18:32:47 2020" "Aug 25 10:48:36 2020" 4096 8 0 source/one
(base) [mfchiang@ my_solution]$ stat source/two
16777220 12888855256 -rw-r--r-- 1 mfchiang staff 0 2 "Sep 10 18:33:00 2020" "Sep 10 18:32:52 2020"
"Sep 10 18:32:52 2020" "Aug 25 10:49:06 2020" 4096 8 0 source/two
```

File System Operations

Delete: Remove the file

- Search the directory for the named file
- Having found the associated directory entry, we release all file space, so that it can be reused by other files, and erase or mark as free

Move

- Moving a file can be performed in different ways depending on the before and after locations
- If both locations are on the same device, change information of the file
- Otherwise, the data have to be copied and then the original deleted (i.e., copy + delete the original)

Seek: Repositioning within a file.

- The current-file-position pointer of the open file is repositioned to a given value.
- Repositioning within a file need not involve any actual I/O.

File System Operations Read

- Operations work on the files contents, Read and Write.
- We need to know where the information is on the device.
- Must specify what data to read, how much, and where to put it.
- Sequential Access
  - Data is retrieved in the same order it is stored
  - Keep a read pointer to the location in the file where the next read is to occur
  - Update the read pointer whenever a read occurs
- Direct (or Random) Access
  - Easy on a disk device
  - The read specifies exactly where it wants to get the data from
  - It could be a byte offset, or a record number

File System Operations
Write

- Very similar to read but commonly requires the allocation of extra space.

- Sequential Access
  - Keep a write pointer to the location in the file where the next write is to occur
  - Update the write pointer whenever a write occurs
- Direct (random) Access
  - The program “seeks” to the new position and write
  - Write can create holes
    - Allocate all of the intervening space and fill it with some null value
    - Mark the intervening space in the directory as not allocated (a.k.a. a *sparse* file)

Design Decisions

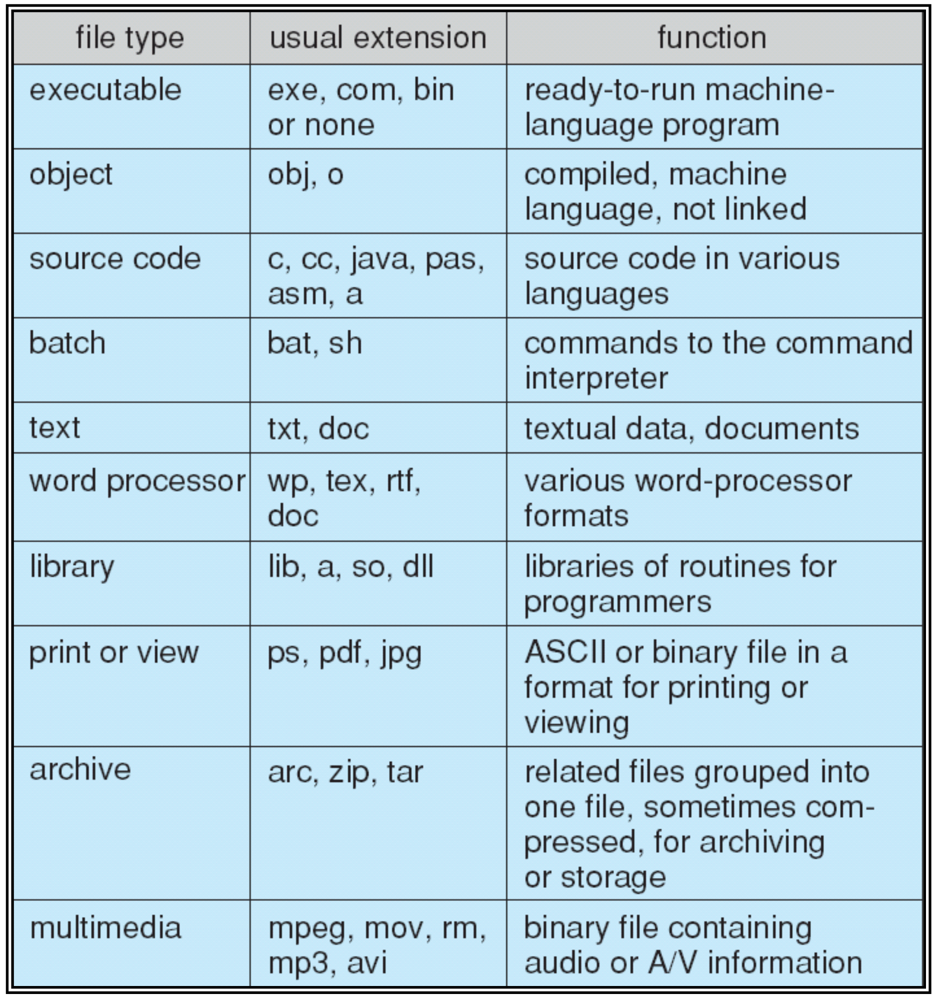
- Files need to contain vastly different types of information.
- Some of this information is tightly structured with lines, records etc.
- Should the file system allow flexibility in how it deals with differently structured files?
  - At the bottom level the file system is working with discrete structures (**sectors** and **blocks**) of a definite size.
  - The most common solution is to treat files as a sequence of bytes and any other structure is enforced by the programs using the files.
  - The work has to be done somewhere.
  - Some operating systems provide more facilities than others for dealing with a variety of file types.

File Attributes

- Information about the files.
- Standard ones
  - File name – the full name includes the directories to traverse to find this file. Many systems use a byte to indicate the file name length and so are limited to 255 characters. There are usually limitations on the characters you can use in filenames.
  - Identifile: usually a number, identifies the file within the file system
  - Location – where is the file stored, some pointer to the device (or server) and the positions on the device
  - Size – either in bytes, blocks, number of records etc
  - Owner information – usually the owner can do anything to a file
  - Other access information – who should be allowed to do what
  - Dates and Times – of creation, access, modification
  - Type – needed for systems that support different types of files

File Type

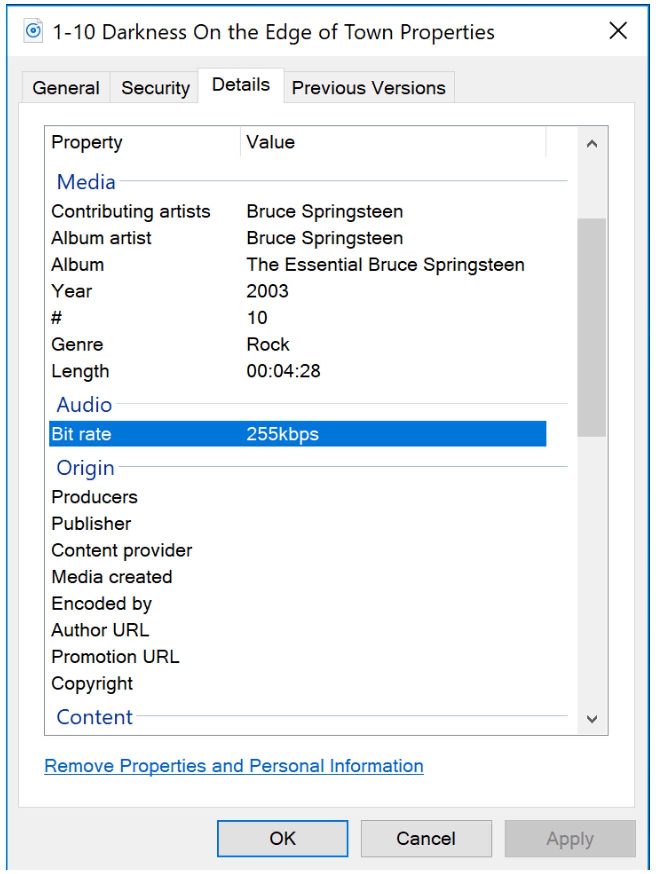
- The more the system knows about file types the more it can perform appropriate tasks.
    - Executable binaries can be loaded and executed.
    - Text files can be indexed.
    - Pictures can have thumbnails generated from them.
    - Files can automatically be opened by corresponding programs.
    - Also the system can stop the user doing something stupid like printing an mp3 file.
- All operating systems "know" about executable binary files.
- OS specific structure – information for the loader about necessary libraries and where different parts should be loaded and where the first instruction is.



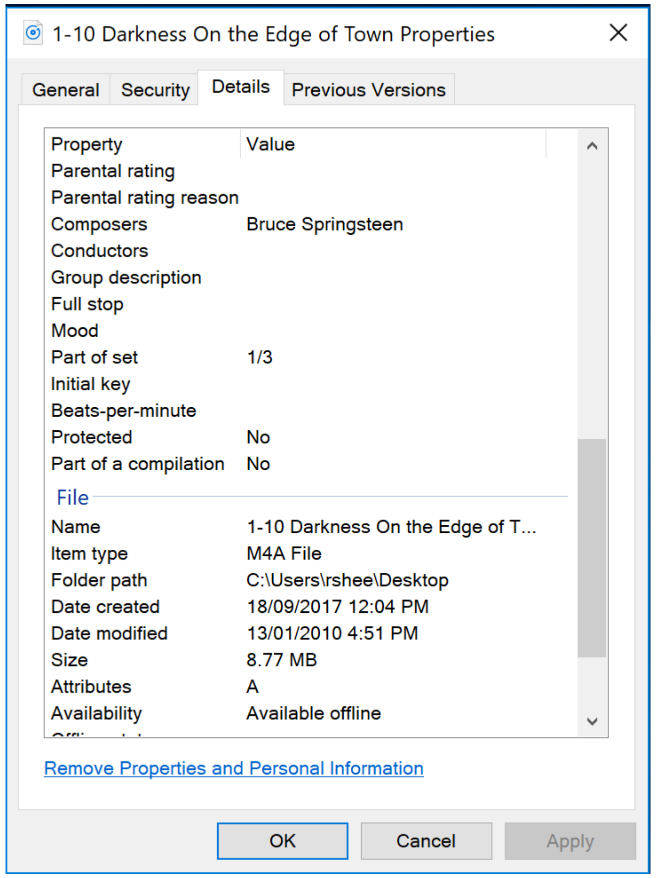
file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

Dealing with File Types

- Windows deals with different file types using a simple extension on the file name.
- The extensions are connected to programs and commands in the system registry.
- There is nothing to stop a user changing an extension (except a warning message).
- UNIX uses magic numbers on the front of the file data.
- If the file is executed the magic number can be used to invoke an interpreter for example.
- For "universal" file types many OSs can extract information from files.


1-10 Darkness On the Edge of Town Properties
General
Security
Details
Previous Versions
Remove Properties and Personal Information
OK
Cancel
Apply

Property	Value
Media	
Contributing artists	Bruce Springsteen
Album artist	Bruce Springsteen
Album	The Essential Bruce Springsteen
Year	2003
#	10
Genre	Rock
Length	00:04:28
Audio	
Bit rate	255kbps
Origin	
Producers	
Publisher	
Content provider	
Media created	
Encoded by	
Author URL	
Promotion URL	
Copyright	
Content	


1-10 Darkness On the Edge of Town Properties
General
Security
Details
Previous Versions
Remove Properties and Personal Information
OK
Cancel
Apply

Property	Value
Parental rating	
Parental rating reason	
Composers	Bruce Springsteen
Conductors	
Group description	
Full stop	
Mood	
Part of set	1/3
Initial key	
Beats-per-minute	
Protected	No
Part of a compilation	No
File	
Name	1-10 Darkness On the Edge of T...
Item type	M4A File
Folder path	C:\Users\rshee\Desktop
Date created	18/09/2017 12:04 PM
Date modified	13/01/2010 4:51 PM
Size	8.77 MB
Attributes	A
Availability	Available offline

Try using the **`file`** command on Linux (or Mac).

And **`od`** e.g.

```
od -a -N 32 README.rtf
```

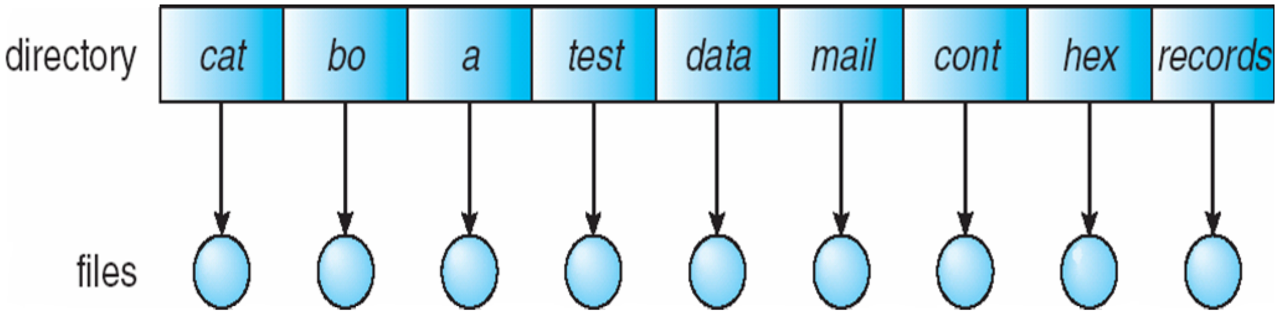
Directory Structure

- Data about files and other information about storage on a device: metadata
- Usually a disk device has one or more directories to store metadata.

- These directories can be arranged in different ways:

Single Level – All files are contained in the same directory

- Advantages: Working well with simple, small systems, especially with small disk devices (floppies)
- Disadvantages: finding files as the number of files grows (some implementations use a B-tree).
- To be workable it requires very long filenames (up to 255 characters).

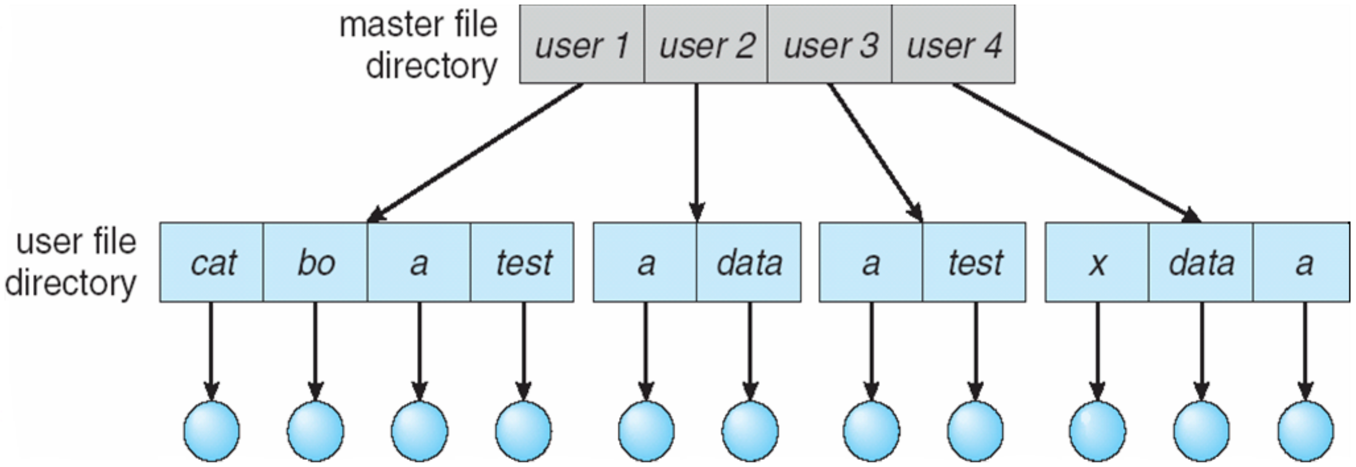


Multiple Levels

Two Level – Create separate directory for each user.

- Top level (MFD): one entry per user on a multi-user system
- Second level (UFD): a single level system to each user

- Creating a user file directory is usually only allowed for administrators.
- When users log in they are placed within their own directories. Any files mentioned are in that directory.
- Can use full pathnames to refer to other user’s files (if permissions allow it).
- Advantages: Allows same file name for different user. Efficient searching.
- Disadvantages: No grouping capability

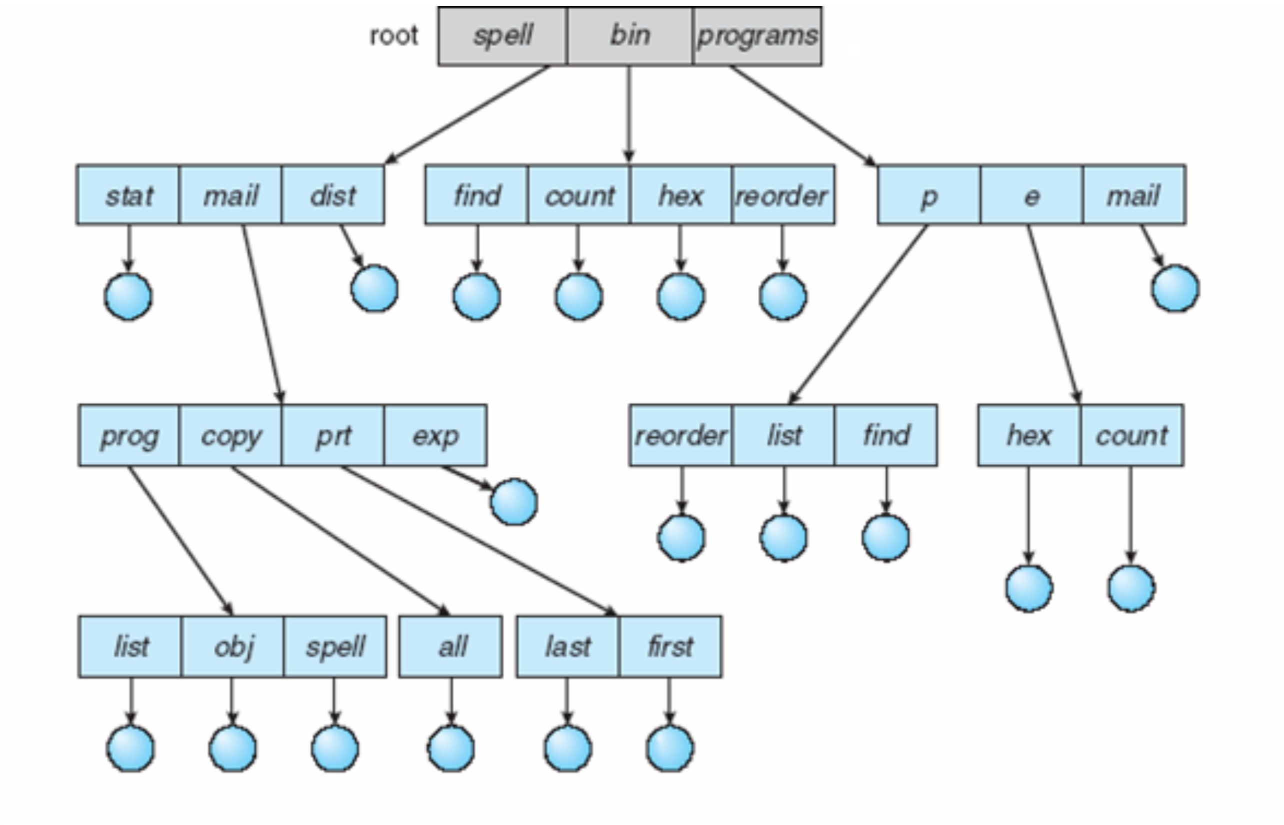


Tree-Structured Directories

Tree – as many directories as required. This facilitates the organisation of collections of files.

- It allows a user to create subdirectories to organize files.
- Directories are special files.

- Advantages:
  - Efficient searching
  - Grouping Capability



Sharing Files/Directories

- We commonly want the same data to be accessible from different places in the file hierarchy. e.g. Two people might be working on a project and they both want the project to be in their local directories.
- This can be accomplished in several different ways:
- If the data is read only we could just make an extra copy.
- We could make two copies of the file record information (not the data).
    - There is a problem with consistency.
- There can be one *true* file entry in one directory. Other entries have some reference to this entry.
    - UNIX **symbolic links** and Windows shortcuts.
- There can be a separate table with file information. Then all directory entries for the same file just point to the corresponding information in this table.
    - UNIX and NTFS **hard links**.

Hard Links

UNIX

ln *ExistingFilename NewFilename*

- Each directory entry stores a pointer to the file’s inode (more on those soon) which holds the real information about the file.

```
$ ls -al
total 8
drwxr-xr-x  5 mfchiang  staff  160 Sep 20 19:35 ..
-rw-r--r--  1 mfchiang  staff    2 Sep 20 19:44 a
```

The reference count of file “a” is 1

```
drwxr-xr-x  3 mfchiang  staff   96 Sep 20 19:44 .
$ ln a a2
```

Create a hard link from file“a2” to text file content in file “a”

```
$ ls -al
total 16
drwxr-xr-x  4 mfchiang  staff  128 Sep 20 19:44 .
drwxr-xr-x  5 mfchiang  staff  160 Sep 20 19:35 ..
-rw-r--r--  2 mfchiang  staff    2 Sep 20 19:44 a
-rw-r--r--  2 mfchiang  staff    2 Sep 20 19:44 a2
```

The reference counts of file “a” and “a2” increase from 1 to 2

```
$ rm a
$ cat a2
```

Content of “a2” still exists even the original text file“a” is deleted

```
a
```

Symbolic Links

- The file called “link2a” is actually just a text file with the contents “a”.
- OS (Unix) knows to treat it differently from other text files because of the “l” in the attributes on the left hand side.
- If the original is moved then UNIX can’t do anything about it. You get unable to open errors.

```
(base) [mfchiang@MacBook lec15]$ ls -al
total 16
drwxr-xr-x  4 mfchiang  staff  128 Sep 20 19:35 .
drwxr-xr-x  5 mfchiang  staff  160 Sep 20 19:35 ..
-rw-r--r--  1 mfchiang  staff    2 Sep 20 19:32 a
(base) [mfchiang@MacBook lec15]$ cat a
a

(base) [mfchiang@MacBook lec15]$ ln -s a link2a
(base) [mfchiang@MacBook lec15]$ ls -al
total 16
drwxr-xr-x  5 mfchiang  staff  160 Sep 20 19:35 .
drwxr-xr-x  5 mfchiang  staff  160 Sep 20 19:35 ..
-rw-r--r--  1 mfchiang  staff    2 Sep 20 19:32 a
lrwxr-xr-x  1 mfchiang  staff    1 Sep 20 19:35 link2a -> a
(base) [mfchiang@MacBook lec15]$ cat link2a
a
(base) [mfchiang@MacBook lec15]$ rm a
(base) [mfchiang@MacBook lec15]$ cat link2a
cat: link2a: No such file or directory
```

Before Next Time

- Read from the textbook
  - Ch14.4: Allocation Methods
  - Ch14.5: Free-Space Management
  - Ch14.6: Efficiency