File Management

Tanenbaum, Chapter 4

COMP3231

Operating Systems

Thomas Sewell

(slides mostly by Kevin Elphinstone)

Outline

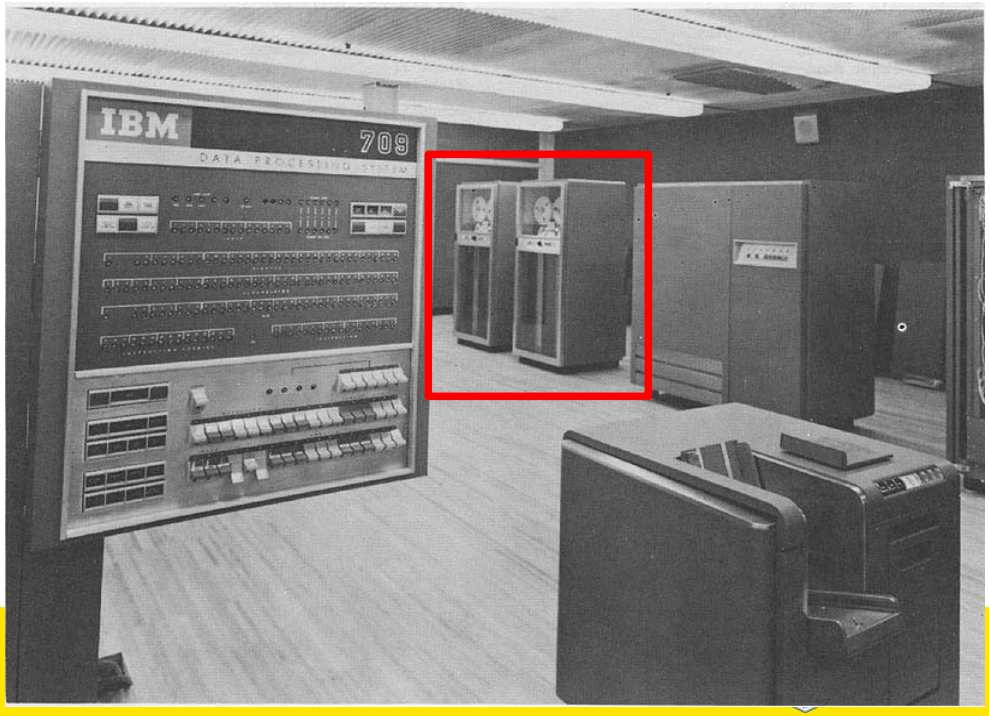
- Files and directories from the programmer (and user) perspective
- Later: Files and directories internals – the operating system perspective

A brief history of file systems

Early batch processing systems:

- No OS
- I/O from/to punch cards
- Tapes and drums for external storage, but no FS
- Rudimentary library support for reading/writing

IBM 709 [1958]



A brief history of file systems

The first file systems (60s)

- Single-level (same as everything in one directory)
- Files are contiguous chunks
  - Maximum size of the file must be known in advance
- You can edit a program and save it in a named file on the tape!

PDP-8 with DECTape [1965]

A brief history of file systems

- Time-sharing OSs
  - Required full-fledged file systems
- MULTICS
  - Multilevel directory structure (keep files that belong to different users separately)
  - Access control lists
  - Symbolic links

Honeywell 6180 running MULTICS [1976]

A brief history of file systems

- UNIX
  - Based on ideas from MULTICS
  - Simpler access control model
  - Everything is a file!

PDP-7

Overview of the FS abstraction

User's view	Under the hood
Uniform namespace	Heterogeneous collection of storage devices
Hierarchical structure	Flat address space (block numbers)
Arbitrarily-sized files	Fixed-size blocks
Symbolic file names	Numeric block addresses
File is one object	File blocks may be "fragmented" across device
Access control	Direct access to devices
Tools for • Formatting • Defragmentation • Backup • Consistency checking	

File System Interface

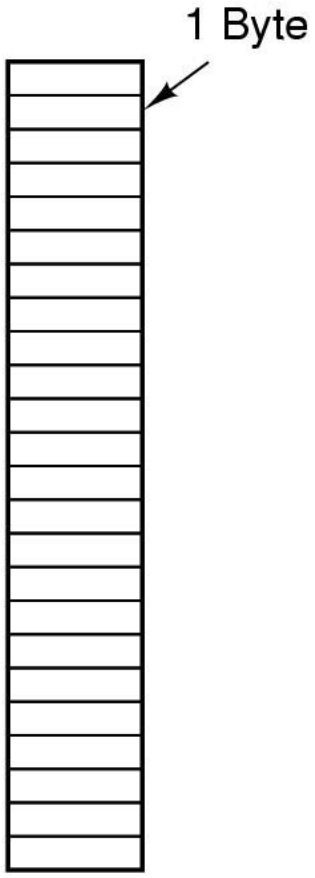
- We're going to discuss the user-visible file system interface
  - fopen("logs/log1.txt", "w");
  - open("logs/log1.txt", O_WRONLY | O_TRUNC);
  - open "logs" dir, find "log1.txt" entry, open that.
  - bytes read and written on a drive.
- You probably already have a good intuition for how these operations work.

File Names

- File system must provide a convenient naming scheme
  - Textual Names
  - May have restrictions
    - Only certain characters
      - E.g. no '/' characters
    - Limited length
    - Only certain format
      - E.g DOS, 8 + 3
  - Case (in)sensitive
  - Names may obey conventions (.c files for C files)
    - Interpreted by tools (e.g. UNIX)
    - Interpreted by operating system (e.g. Windows "con:")

File Structure

- Sequence of Bytes
  - OS considers a file to be unstructured
  - Applications can impose their own structure
  - Used by UNIX, Windows, most modern Oses
    - Mac OS experimented with structured files



(a)

File Types

- Regular files
- Directories
- Device Files
  - May be divided into
    - Character Devices – stream of bytes
    - Block Devices
- Streams/Pipes
- Some systems distinguish between regular file types
  - ASCII text files, binary files

File Access Types (Patterns)

- Sequential access
  - read all bytes/records from the beginning
  - cannot jump around, could rewind or back up
  - convenient when medium was magnetic tape
- Random access
  - bytes/records read in any order
  - essential for data base systems
  - read can be …
    - move file pointer (seek), then read or
      - lseek(location,…);read(…)
    - each read specifies the file pointer
      - read(location,…)

Possible File Attributes

Attribute	Meaning
Protection	Who can access the file and in what way
Password	Password needed to access the file
Creator	ID of the person who created the file
Owner	Current owner
Read-only flag	0 for read/write; 1 for read only
Hidden flag	0 for normal; 1 for do not display in listings
System flag	0 for normal files; 1 for system file
Archive flag	0 for has been backed up; 1 for needs to be backed up
ASCII/binary flag	0 for ASCII file; 1 for binary file
Random access flag	0 for sequential access only; 1 for random access
Temporary flag	0 for normal; 1 for delete file on process exit
Lock flags	0 for unlocked; nonzero for locked
Record length	Number of bytes in a record
Key position	Offset of the key within each record
Key length	Number of bytes in the key field
Creation time	Date and time the file was created
Time of last access	Date and time the file was last accessed
Time of last change	Date and time the file has last changed
Current size	Number of bytes in the file
Maximum size	Number of bytes the file may grow to

Typical File Operations

- Create
- Delete
- Open
- Close
- Read
- Write
- Append
- Seek
- Get attributes
- Set Attributes
- Rename

Example File System Calls (1/2)

```
/* Example file copy program. Error checking is minimal */

#include <sys/types.h>
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>

#define BUF_SIZE 4096
#define FILE_MODE 0700

int main (int argc, char *argv) {
    int in_fd, out_fd, rd_count, wt_count;
    char buffer[BUF_SIZE];

    if (argc != 3) { exit(1); }
```

Example File System Calls (2/2)

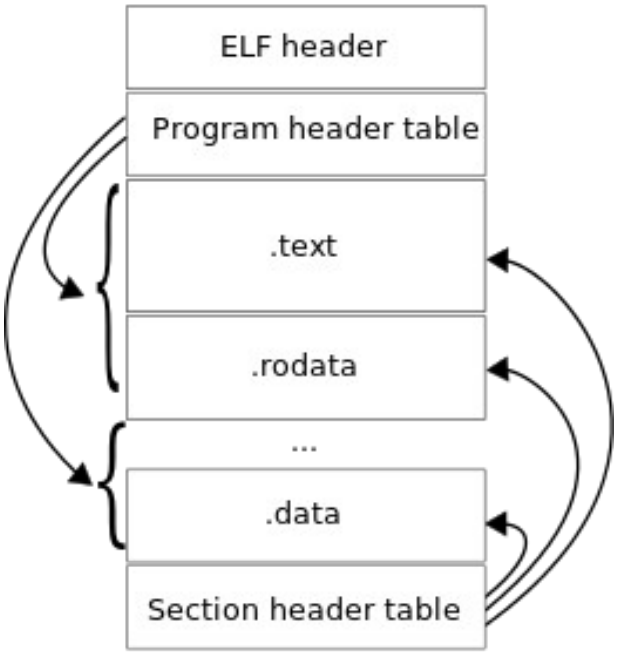
```
in_fd = open(argv[1], O_RDONLY);    /* Open source file. */
if (in_fd < 0) { exit(2); }         /* Exit on failure. */
out_fd = creat(argv[2], FILE_MODE);    /* Create dest file. */
if (out_fd < 0) { exit(3); }        /* Exit on failure. */

/* Copy loop */
while (TRUE) {
    rd_count = read(in_fd, buffer, BUF_SIZE);   /* Read chunk. */
    if (rd_count <= 0) { break; }        /* Exit on failure. */
    wt_count = write(out_fd, buffer, rd_count);  /* Write chunk */
    if (wt_count <= 0) { exit(4); }        /* Exit on failure. */
}

/* Close files. */
close(in_fd);
close(out_fd);
return 0;
}
```

File Organisation and Access
Programmer's Perspective

- Given an operating system supporting unstructured files that are a *stream-of-bytes*,

  how can one organise the contents of the files?

E.g. Executable Linkable Format (ELF)



File Organisation and Access
Programmer's Perspective

- Some possible access patterns:
  - Read the whole file
  - Read individual records from a file
    - record = sequence of bytes containing the record
  - Read records preceding or following the current one
  - Retrieve a set of records
  - Write a whole file sequentially
  - Insert/delete/update records in a file

Programmers are free to structure the file to suit the application.

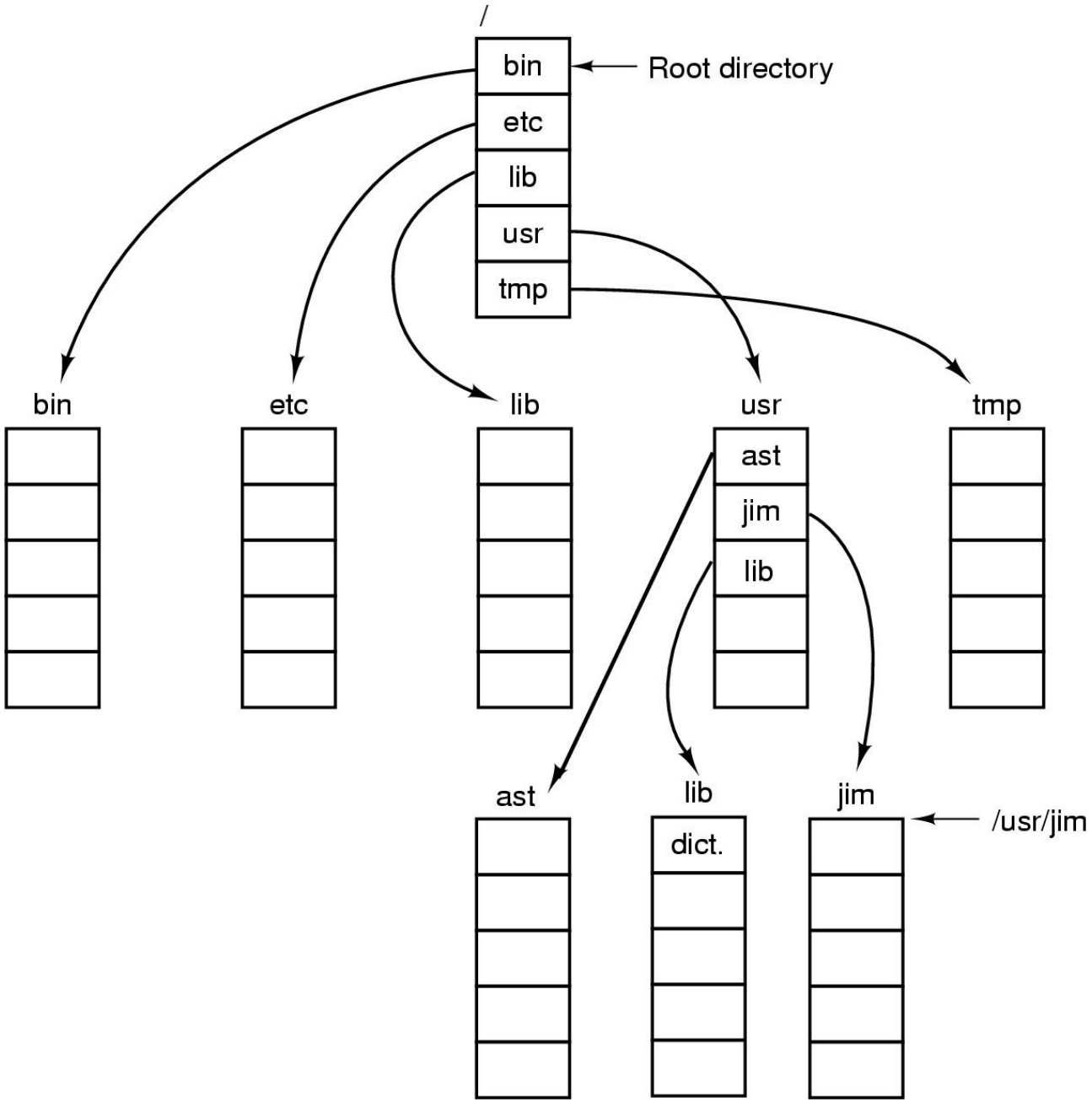
Criteria for File Organization

Things to consider when designing file layout

- Rapid access
  - Needed when accessing a single record
  - Not needed for batch mode
    - read from start to finish
- Ease of update
  - File on CD-ROM will not be updated, so this is not a concern
- Economy of storage
  - Should be minimum redundancy in the data
  - Redundancy can be used to speed access such as an index

File Directories

- Provide mapping between file names and the files themselves
- Contain information about files
  - Attributes
  - Location
  - Ownership
- Directory is itself a file owned by the operating system


/
bin
etc
lib
usr
tmp
Root directory
bin
etc
lib
usr
tmp
ast
jim
lib
ast
lib
jim
dict.
/usr/jim

Hierarchical (Tree-Structured) Directory

- Files can be located by following a path from the root, or master, directory down various branches
  - This is the *absolute* pathname for the file
- Can have several files with the same file name as long as they have unique path names

Current *Working Directory*

- Always specifying the absolute pathname for a file is tedious!
- Introduce the idea of a *working directory*
  - Files are referenced relative to the working directory
- Example: cwd = /home/tsewell

  .profile = /home/tsewell/.profile

Relative and Absolute Pathnames

Absolute path names:

- A path from the root of the system to the file.
- e.g. /home/tsewell/work/os_3231/lectures/26t2/w04b-files.pdf

Relative path names:

- A path from the current working directory.
- Note that (in UNIX) '.' and '..' refer to current and parent dir.
- e.g. suppose cwd = /home/tsewell
- ../../etc/passwd = /etc/passwd
- ../kevine = /home/kevine
- ./work/os_3231/lectures = /home/tsewell/work/os_3231/lectures

Typical Directory Operations

- Create
- Delete
- Opendir
- Closedir
- Readdir
- Rename
- Link
- Unlink

Nice properties of UNIX naming

- Simple, regular format
  - Names referring to different servers, objects, etc., have the same syntax.
    - Regular tools can be used where specialised tools would be otherwise be needed.
- Location independent
  - Objects can be distributed or migrated, and continue with the same names.

  Where is /home/tsewell/.profile?

  You only need to know the name!

An example of a bad naming convention

- From, Rob Pike and Peter Weinberger, “The Hideous Name”, Bell Labs TR

UCBVAX::SYS$DISK:[ROB.BIN]CAT_V.EXE;13

File Sharing

- In multiuser system, allow files to be shared among users
- Two issues
  - Access rights
  - Management of simultaneous access

Access Rights

- None
  - User may not know of the existence of the file
  - User is not allowed to read the directory that includes the file
- Knowledge
  - User can only determine that the file exists and who its owner is

Access Rights

- Execution
  - The user can load and execute a program but cannot copy it
- Reading
  - The user can read the file for any purpose, including copying and execution
- Appending
  - The user can add data to the file but cannot modify or delete any of the file's contents

Access Rights

- Updating
  - The user can modify, delete, and add to the file's data. This includes creating the file, rewriting it, and removing all or part of the data
- Changing protection
  - The user can change the access rights granted to other users
- Deletion
  - The user can delete the file

Access Rights

- Owners
  - The owner user has all rights previously listed
  - They may grant rights to others using the following classes of users:
    - Specific users
    - User groups
    - All users
      - For public files

Simultaneous Access

- Most OSes provide mechanisms for users to manage concurrent access to files
  - Example: flock(), lockf(), system calls
- Typically
  - User may lock entire file when it is to be updated
  - User may lock individual records (i.e. ranges) during an update
- Mutual exclusion and deadlock are issues for shared access

This Lecture

- Files and directories from the programmer (and user) perspective.
- Typical operations of the file-system interface.
- File and path names and hierarchical directory structure.
- File access rights.