File Systems

- Requirements
  - It must be possible to store a very large amount of info
  - The info must survive the termination of the process using it
  - Multiple processes must be able to access the info concurrently
- Files
  - On disks or other media units
  - The info is **persistent** (i.e. info not affected by process creation or termination)
- File system
  - A method for storing and organising computer files and the data they contain to make it easy to find and access them.

Files

- File Naming
  - Conventions vary from system to system
  - Common: 8 characters, a dot and a file extension
    - E.g. fish.dat
  - May be case sensitive (Fish.dat , fish.dat)
  - File extension
    - MS-DOS: 1 – 3 chars; Unix: variable
    - Sometimes, just conventions; sometimes required (e.g. C compiler)
    - Examples: file.bak, file.c, file.dat file.doc, file.lib, file.man, file.obj, file.txt

File Structure and Types

- 3 main structures
  - Byte sequence- user programmes decide how to handle info
  - Record sequence – each record has some internal structure
  - Tree – each record has a **key** for accessing the info (the data is sorted)
- File types
  - Regular files
    - user info, either ASCII or binary files
    - ASCII files are printed as they are displayed, easy to share
    - Binary files usually have some internal structure, e.g. executable files, library files; would print “rubbish” if printed directly
  - directories

File Access and Attributes

- Access:
  - Sequential: start reading file at beginning and read data or records in order (OLD way)
  - Random Access Files (RAM): data or records can be read in any order
- Attributes:
  - Extra info on a file; different systems use different attributes
  - E.G. protection, password, creator, owner, flags (read-only, hidden, system, archive, binary, temporary), record length, key position, key length, creation time, time of last access, time of last change, current size, max size

File Operations

- Create, delete, open, close, read, write
- Append (only add data to end of file)
- Seek (repositions the pointer to the current position to a specific place in the file) G
- Get attributes (e.g. Unix), set attributes (e.g. flags)
- Rename

Directories

- Hierarchical file systems
  - Concept of **root** directory, with levels of **sub-directories** and then files
- Path names
  - **Absolute path name** consists of the path from the root directory to the file e.g. /usr/ast/mailbox (in unix)
  - **Relative path name** is when the path names are taken relative to the **current** (**working**) directory e.g. mailbox if the user is currently in /usr/ast
  - 2 special entries:
    - . (dot): refers to the current directory
    - .. (dot dot): refers to the parent directory
    - E.g. cp /usr/lib/dictionary . (unix)
- Operations:
  - (unix) Create, delete, opendir, closedir, readdir, rename, link, unlink)

File System Implementation

- Contiguous allocation
  - Each file is stored as a contiguous block of data
  - Simple to implement, good performance
  - No feasible unless max size is know at creation, disk fragmentation
- Linked list allocation
  - Link each block of data to the next one
  - No disk fragmentation, reading info is slow
- Improvements: link list allocation using an index, i-node

Disk Space Management

- 2 strategies
  - Store all file in consecutive bytes of disk space (not used)
  - Split file up into a number of block
  - Akin to memory management systems
- block size is an important consideration
- Need to keep track of free blocks
- Disk quotas
  - The system administrator assigns each user a maximum allotment of files and blocks and the OS makes sure that the users do not exceed their quotas
  - Soft limit: give warning; hard limit may never be exceeded

File System Reliability

- Very important
  - Often easier to replace computer than data
- Bad block management
  - System detects bad blocks and never uses them
- Backups
  - Dump file system every day
  - Incremental dumps
    - Make a complete dump periodically and make a daily dump of only those files that have been modified since the last full dump
    - Some systems use an archive bit

File System Issues

- Consistency
  - Check to see if files are OK after a crash
- Performance
  - Block or buffer cache – keeps data in memory for performance reasons
  - Like paging: FIFO, LRU
  - Need to write out info to disk

Types of File Systems

- Disk file systems
- Flash file systems
  - Designed for storing files on flash memory devices
- Database file systems
- Transactional file systems
  - A transaction can either be finished completely or reverted completely
- Network file systems
  - Is a file system that acts as a client for a remote access protocol, proving access to files on a server

Security

- Security: overall problem
- Protection mechanism: refers to the specific OS mechanisms used to safeguard info
- 2 aspects:
  - data loss
    - Act of God; h/w or s/w errors, human errors
  - Intruders
    - Casual prying, snooping, attempt to make money, commercial or military espionage
- Privacy
  - Protecting individuals from misuse of info about them

Security

- Some security flaws
  - Getting access to the password file
  - **Trojan horse attack**: modifying a normal programme to do nasty things in addition to its usual function and arranging for the victim to use the modified version
- Viruses and Works
  - **Worm**: complete programme that seeks to cause damage
  - **Virus**: is a programme fragment that is attached to a legitimate programme with the intention of infecting other programmes
  - Both attempt to spread themselves and can do severe damage
- Tiger or penetration teams

Design Principles for Security

- The system should be public
- The default should be no access
- Check for current authority
- Give each process the least privilege possible
- The protection mechanism should be simple, uniform and built into the lowest layers of the system
- The scheme chosen must be psychologically acceptable
- User authentication:
  - Passwords, questions, challenge-response, physical identification