Filesystem

- Part of the OS that deals with file management
- Result of the integration of storage resources under a single hierarchy
- Unix created history by blurring the distinction between files and I/O devices
- The newer versions of Unix also blur the distinction between files and processes, serial ports, IPC channels, and shared memory segments
- Typical UNIX system contains thousands of files which can be broadly classified as

  System files contain important information about the computer

  System programs are files containing commands for common tasks – one command to each file, such as `ls`

  Application programs contain commands such as `vi`, LaTeX

  User data files contain information to be processed such as reports and correspondence
- Unix maps a number of different types of objects into filesystem namespace
  - Advantage: Consistent programming interface
  - Disadvantages: Complexity
- Main components of filesystem
  1. Namespace: Naming objects and arranging them in a hierarchy
  2. API: Set of system calls for navigating and manipulating nodes
  3. Security model: Protect/hide/share objects
  4. Implementation: Code that ties model to disk
- Interface on modern systems
  - Defined as an abstract kernel-level interface
  - Portions are handled by different drivers within the kernel
  - No clear architectural boundaries
    - Device files make programs communicate with drivers instead of kernel
    - Handled by basic filesystem driver
  - More than one type of disk-based filesystem
    - System may require higher reliability or easier fault recovery
    - Extensions like access control lists not available in traditional filesystems
    - Accommodating other systems media such as DOS floppies and CD-ROMs

Pathnames

- Filesystem as a single hierarchy starting at `/`
- List of directories to be traversed
- Absolute and relative pathnames
- Arbitrarily deep filesystem
  - A filename cannot be more than 255 characters long
  - A pathname cannot be more than 1023 characters long

  - Longer pathname can be accessed by `cd`ing to an intermediate directory

- Filenames in Unix
  - Can use any character, except `/` and null
  - As a guideline, try to avoid names that
    - Start with a `+` or `-`
    - Contain spaces, `$`, `%`, `!`, or control characters (including backspace and carriage return)
  - Filename extensions
    - UNIX does not attach specific meaning to any form of file name
      - The commands do not have to end in `.exe` or `.com`
    - Some programs look at the extensions to determine if they are usable (such as `.c` for C programs, used by the `make` utility)
    - More often than not, the filename extensions are given for human understanding than for system programs
    - Common filename extensions in UNIX are given in Table 1

Table 1: Common filename extensions in UNIX

Extension	File type	Used by
`.a`	Library archive	Linker
`.c`	C source program	C compiler
`.C`	C++ source program	C++ compiler
`.f`	Fortran source program	Fortran compiler
`.gz`	Compressed file	Gnu zip programs
`.h`	C/C++ header file	C/C++ source
`.l`	Source for lexical analyzer	`lex`
`.o`	Compiled object file	Linker
`.s`	Assembly source file	Assembler
`.so`	Shared object	Run-time dynamically linked library
`.tex`	LaTeX and TeX source	LaTeX and TeX
`,v`	Revision control	RCS
`.y`	Source for `yacc`	`yacc`
`.Z`	Compressed file	Compress and uncompress tools
`.z`	Packed file	Pack and unpack tools

Mounting and unmounting filesystems

- File tree (complete filesystem hierarchy) has filesystems or partitions attached to it
- There can also be filesystems other than disk partitions such as network file server, kernel components, and memory-based disk emulators
- Filesystems attached to the tree by using `mount` command
  - `mount` attaches a file system to the hierarchy at the *mount point*
  - If mount point has any contents prior to mount operation, they are hidden until the file system is unmounted
    - Recommended to declare mount points as empty directories
- Lists of filesystems is customarily kept in the files `/etc/fstab`, `/etc/vfstab`, or `/etc/checklist`

  - `vfstab` describes defaults for each file system
    * Serves as layout documentation for file system
    * Information used to check filesystems using `fsck` and mounted automatically at boot time
    * Enables shorter `mount` command such as `mount /usr`

- Filesystems are detached from hierarchy by the `umount` command
  - Filesystem must be currently mounted and must not be busy
  - No open files or processes with current working directory in the filesystem
  - If filesystem contains programs, they should not be running
  - Busy filesystems can be unmounted by using the option `-f`
    * On Solaris, first run the command `lockfs -h` to *hard lock* the filesystem before issuing `umount -f`
  - The reason for a filesystem being busy can be determined by using the command `fuser -c`
    * Prints the id of the process as well as a letter code to give the reason
    * Use the `ps` command to check on the processes
    * An alternative to `fuser` is provided by `lsof`
- The default filesystem type is kept in the file `/etc/default/fs`
- The mount table is kept in the file `/etc/mnttab`
  - `mount` adds an entry to mount table
  - `umount` removes the corresponding entry from mount table

File tree organization – Deconstructing filesystem

- Not very well organized – flexibility vs standardization
- Composed of chunks called filesystems
  - Filesystem consists of one directory and its subdirectories
- Root filesystem
  - Contains the root directory and a minimal set of subdirectories
  - The `kernel` resides in the root filesystem
  - `/bin`
    * Contains binary executables for different user commands
    * Commands needed for minimum system operability
    * `/bin` is sometimes a link to `/usr/bin`
    * Other directories containing the user commands are `/usr/bin` and `/usr/ucb`
  - `/dev`
    * Contains device entries for terminals, disks, modems, etc.
    * Device types indicated by the name of the file
      · `dsk` – Disk accessed in block mode
      · `rdsk` – Disk accessed in raw mode
      · `mt` – Magnetic tape accessed in block mode
      · `rmt` – Magnetic tape accessed in raw mode
      · `term` – Terminal on serial line
      · `pts` or `ptc` – Pseudo terminal

- `/etc` and `/sbin`
  - System configuration files and executables, administrative files, boot scripts
  - Executable binaries for most system administration commands
  - `/etc/default`, if it exists, may contain default parameter values for various commands
- `/home`
  - Users' home directories
  - `/u` or `/users` in some systems
- `/lost+found`
  - Directory for lost files
  - Files may be lost due to disk error or improper system shutdown
    - Refer to disk locations marked as used but not listed in any directory
    - A non-empty inode not listed in any directory
  - The program `fsck`, normally run at boot time, finds these files
  - Every disk partition has a `lost+found` directory
- `/mnt`
  - Mount directory for temporary mounts
- `/proc`
  - Images of all running processes
  - Allows processes to be manipulable using Unix file access system calls
  - Files correspond to active processes (entries in the kernel process table)
  - There may be additional files (on Linux) containing information on system configuration: use of interrupts and I/O ports, and allocation of DMA channel and CPU
- `/tcb`
  - Trusted Computer Base
  - Directory tree for security-related database files on some systems offering enhanced security features
  - Configuration files related to the TCB are stored under `/etc/auth`
- `/tmp`
  - Temporary files that disappear between reboots
  - Available to all users as a scratch directory
- `/usr`
  - Contains subdirectories for locally generated programs, executables for user and administrative commands, shared libraries, and other parts of Unix OS
  - Also may contain application programs
  - `/usr/adm`
    - Administrative directory
    - Accounting files, records of resource usage
    - Recent versions of Unix have this directory changed to and linked to `/var/adm`
  - `/usr/bin`
    - Executable files, including shellscripts
    - Executables for X window system are stored in `/usr/bin/X11`
  - `/usr/games`
    - Games and diversions (old collection; not fun any more)
    - Some sites may not even have this one
  - `/usr/include`
    - Header files for C programs

    - Useful to define the program's interface to standard system libraries
    - Directory `/usr/include/sys` contains include files for operating system
  - `/usr/ucb`
    - Berkeley utilities and programs
  - `/usr/lib`
    - Support files for standard Unix applications
    - Standard C libraries for math and I/O
    - Names of the form `libx.a` where `x` is one or more characters related to the library's contents
    - Also may contain configuration files for some Unix services
  - `/usr/local`
    - Local software
    - Subdivided into another hierarchy
    - `/usr/local/adm`
    - `/usr/local/bin`
    - `/usr/local/etc`
    - `/usr/local/lib`
    - `/usr/local/sbin`
    - `/usr/local/src`
  - `/usr/man`
    - On-line manual pages
    - Divided into subdirectories for the various sections of the manual
    - Contains several `manx` and `catx` directories where `x` denotes the number 1 through 8, or the letters l or n
    - `catx` directories may be eliminated to save space
    - Significance of the numbers is given by the following table

1	User commands
2	System calls
3	Subroutines
4	Devices (Special files and hardware)
5	File formats and configuration files
6	Games and demos
7	Miscellaneous: characters sets, filesystem types, etc
8	System administration and maintenance
l	Local
n	New

  - `/usr/share`
    - Shared data
    - Static data files, such as manual pages, font directories, files for `spell`
    - `/usr/share/man`
      Shared manual pages
- `/var`
  - Varying data, including spooling and other volatile directories
  - `/var/spool`
    - Spooling directories for printers, mail, UUCP
    - `cron` utility also keeps the files here
  - `/var/tmp`
    - Temporary space where the files do not disappear between reboots

File types

- File types are hard-coded and cannot be changed for any file
- Only seven types of files defined for Unix

1. Regular files
   - Most common types of files
   - May contain ASCII characters, binary data, executable program binaries, program input or output
   - Both sequential and random access may need to be supported
   - Regular files can be further classified as
     (a) Text files (ASCII)
       * Lines of text
       * Lines may be terminated by carriage return
       * File itself has an end-of-file character
     (b) Binary files
       * Not readily readable
       * Has internal structure depending upon the type of file (executable or archive)
2. Directories
   - Binary file containing a list of files contained in it (including other directories)
   - May contain any kind of files, in any combination
   - `.` and `..` refer to directory itself and its parent directory
   - Created by `mkdir` and deleted by `rmdir`, if empty
   - Non-empty directories can be deleted by `rm -r`
3. Character-special files and Block-special files (types 3 and 4)
   - Allow Unix applications to communicate with the hardware and peripherals
   - Reside in the `/dev` directory
   - Character-special files
     * Allow the device drivers to perform their own I/O buffering
     * Used for unbuffered data transfer to and from a device
     * Generally have names beginning with `r` (for *raw*), such as `/dev/rsd0a`
   - Block-special devices
     * Expect the kernel to perform buffering for them
     * Used for devices that handle I/O in large chunks, known as blocks
     * Generally have names without the `r`, such as `/dev/sd0a`
   - Possible to have more than one instance of each type of device
     * Device files characterized by major and minor device number
     * Major device number
       Tells the kernel the driver corresponding to the file
     * Minor device number
       Tells the kernel about the specific instance of the device
       Tape drivers may use the minor device number to select the density for writing the tape
4. Hard links (not a separate type)
   - Additional name (alias) for a file
   - Associates two or more filenames with the same inode
   - Indistinguishable from the file it is linked to
   - Share the same disk data blocks while functioning as independent directory entries
   - May not span disk partitions as inode numbers are only unique within a given device

   - Unix maintains a count of the number of links that point to the same file and does not release the data blocks until the last link has been deleted
   - Created with `ln` and removed with `rm`

5. Symbolic links
   - Also known as *soft* link
   - Pointer files that name another file elsewhere on the file system
   - Reference by name; distinct from the file being pointed to
   - Points to a Unix pathname, not to an actual disk location
   - May even refer to non-existent files, or form a loop
   - May contain absolute or relative path
   - Created with `ln -s` and removed with `rm`
   - Problem of using `..` in the symbolic link
6. FIFO special file, or "named pipe" (ATT)
   - Characterized by transient data
   - Allow communications between two unrelated processes running on the same host
   - Once data is read from a pipe, it cannot be read again
   - Data is read in the order in which it was written to the pipe, and the system allows no deviation from that order
7. Unix domain sockets (BSD)
   - Connections between processes for communications
   - Part of the TCP/IP networking functionality
   - Communication end point, tied to a particular port, to which processes may attach
   - Socket `/dev/printer` is used to send messages to the line printer spooling daemon `lpd`
   - Visible to other processes as directory entries but cannot be read from or written into by processes not involved in the connection
   - Created with the `socket` system call, and removed with `rm` or `unlink` command (if not in use)

File attributes

- Mode bits
  - Nine permission bits to specify rwx permissions
  - Three bits to specify the operation of executable programs
- Twelve mode bits stored with four file-type information bits
- `setuid` and `setgid` bits
  - Bits with octal value 4000 and 2000
  - Modify effective uid and gid for a running program
  - Capitalized if program does not have x bit set
  - `setgid` bit on a directory causes the newly created files in directory to take group ownership of directory
- Sticky bit
  - Bit with octal value 1000
  - Obsolete for executables today
  - Makes public directories somewhat private

- Permission bits
  - Meaning for file and directory

Access	File	Directory
read	View contents	Search contents (using `ls`)
write	Change contents	Change contents (add or delete files)
execute	Run the executable	Allowed to get into the directory

- Permission changed with `chmod`, `chown`, `chgrp`, `umask`