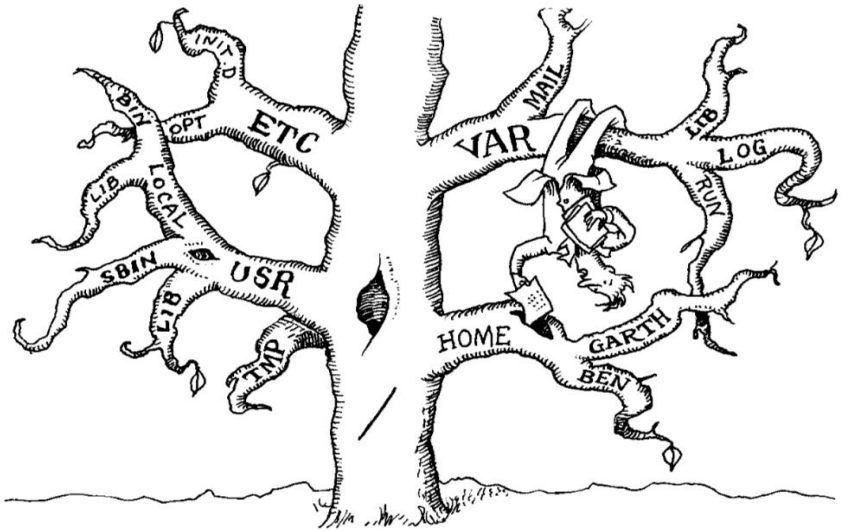
CS183

Instructor: Ali Davanian
(Slides were adopted from Brian Crites and Alireza Abdoli)



The Filesystem

The Filesystem

- Basic Purpose: to represent and organize the system’s storage resources
- Also: represents a ton of other systems as files (devices, busses, etc.)
- Comprised of four main components:
  - **Namespace**: a way to name things and organize them in a hierarchy
  - **API**: a set of system calls for navigating and manipulating objects
  - **Security Models**: schemes for protecting, hiding, and sharing things
  - **Implementation**: software to tie the logical model to the hardware

Pathnames

Absolute paths start from the root directly `/`

`/etc/apache2/httpd.conf`

Relative paths are invisibly prepended with your current working directory

`apache2/httpd.conf -> /etc/apache2/httpd.conf`

(the above assumes our current working directory is /etc)

The locations `.` and `..` are special files which refer to the current and parent directories, respectively

mount and automounting

- The command `mount` is used to attach a file system to your full file tree, and `unmount` is used to remove it
- `mount` can attach any file system (including ones already connected to the file tree somewhere else, remote drives, other disks, etc.) to the file tree via a mount point
- The mount point is (usually) an empty directory which will have its contents overwritten with the mounting file system
- File systems which should be automatically mounted at boot are listed at `/etc/fstab` and this file can be modified to add/remove boot file systems

```
#
# /etc/fstab
# Created by anaconda on Tue Apr  2 18:19:15 2019
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
UUID=52c8f406-e595-4027-8a70-c267be111c37 /                       ext4    defaults        1 1
UUID=e6b90fa6-d7c9-42f8-bf70-db29f8c1190c swap                    swap    defaults        0 0
```

- Use *lsblk* and *blkid* to find the UUID for the file system you want to automount
- Create a backup of the `/etc/fstab` file (example: `/etc/fstab.old`)
- Create an empty directory to use as a mount point
- Fill in the following fields in `/etc/fstab`
  - UUID, mount point, file system type, mount options (defaults usually), dump, fsck

Standard directories and their contents

Pathname	Contents
/bin	Core operating system commands
/boot	Boot loader, kernel, and files needed by the kernel
/compat	On FreeBSD, files and libraries for Linux binary compatibility
/dev	Device entries for disks, printers, pseudo-terminals, etc.
/etc	Critical startup and configuration files
/home	Default home directories for users
/lib	Libraries, shared libraries, and commands used by **/bin** and **/sbin**
/media	Mount points for filesystems on removable media
/mnt	Temporary mount points, mounts for removable media
/opt	Optional software packages (rarely used, for compatibility)
/proc	Information about all running processes
/root	Home directory of the superuser (sometimes just /)
/run	Rendezvous points for running programs (PIDs, sockets, etc.)
/sbin	Core operating system commands [a]
/srv	Files held for distribution through web or other servers
/sys	A plethora of different kernel interfaces (Linux)
/tmp	Temporary files that may disappear between reboots
/usr	Hierarchy of secondary files and commands
/usr/bin	Most commands and executable files
/usr/include	Header files for compiling C programs
/usr/lib	Libraries; also, support files for standard programs
/usr/local	Local software or configuration data; mirrors **/usr**
/usr/sbin	Less essential commands for administration and repair
/usr/share	Items that might be common to multiple systems
/usr/share/man	On-line manual pages
/usr/src	Source code for nonlocal software (not widely used)
/usr/tmp	More temporary space (preserved between reboots)
/var	System-specific data and a few configuration files
/var/adm	Varies: logs, setup records, strange administrative bits
/var/log	System log files
/var/run	Same function as **/run**; now often a symlink
/var/spool	Spooling (that is, storage) directories for printers, mail, etc.
/var/tmp	More temporary space (preserved between reboots)

a. The distinguishing characteristic of **/sbin** was originally that its contents were statically linked and so had fewer dependencies on other parts of the system. These days, all binaries are dynamically linked and there is no real difference between **/bin** and **/sbin**.

File Types

- There are seven typical types of files
  - Regular files
  - Directories
  - Character device files
  - Block device files
  - Local domain sockets
  - Named pipes (FIFOs)
  - Symbolic links
- Even when creating a new system that utilizes the file abstraction, you must make your file look like one of these types
- Use the `file` command to find which type a file is

File Types

- Regular Files
  - Series of bytes with no structure imposed by the filesystem
- Directories
  - Named references to other files with special entries . and .. automatically added (which cannot be removed)
- Hard Links
  - Hard links point to the actual node in the hard drive (not the filename)
  - More than one link can refer to a file at once, and the reference can have different names (but cannot cross file systems)
  - Hard links are created using `ln existingfile newlinkname`

File Types

- Symbolic Link
  - Reference to a file by name, and can be absolute or relative
  - Symbolic links are distinct from the files they point to, where as hard links are actual reference to a specific file
- Character and Block Device Files
  - Abstraction for communicating with system hardware and peripherals
  - Files system requests that refer to a character or block device file are passed to the associated device driver
  - Device driver is a process (usually a daemon) which takes care of running and interfacing with the device or peripheral
  - Device files have an associated major device number (specifies driver) and a minor device number (used by driver, usage varies by device)

File Types

- Local Domain Sockets
  - Connections between processes that allow them to communicate easily
  - These sockets are only available through the local host so are referred through the file system rather than a network port
- Named Pipes
  - Serves a similar purpose to local domain sockets
  - These files are a historical artifact, and local domain sockets are essentially a superset of named pipes

File-type encoding used by ls

File type	Symbol	Created by	Removed by
Regular file	–	editors, **cp**, etc.	**rm**
Directory	`d`	**mkdir**	**rmdir**, **rm -r**
Character device file	`c`	**mknod**	**rm**
Block device file	`b`	**mknod**	**rm**
Local domain socket	`s`	**socket** system call	**rm**
Named pipe	`p`	**mknod**	**rm**
Symbolic link	`l`	**ln -s**	**rm**

File Permissions

- Each file has 9 permission bits: 3 for the owner, 3 for the group, and 3 for all
- These are often set with octal (base 8) numbers
  - The topmost three bits (400, 200, 100) represent user permissions
  - The middle three bits (40, 20, 10) represent group permissions
  - The lowest three bits (4, 2, 1) represent all permissions
  - The high bit is the read bit, the middle bit is the write bit, the low bit is execute
- Only the most specific permissions apply when a user fits into multiple categories

Special Permission

- The octal values 4000 and 2000 are the setuid and setgid permission bits
- The octal value 1000 is used for the sticky bit, which is ignored on normal files
- The sticky bit, when set on a directory, means that the directory and files within it can only be removed by the file/directory owner or superuser
- Linux systems also have a number of “bonus flags” which can only be viewed and modified with `lsattr` and `chattr` respective
- These “bonus flags” can further modify the filesystem but are not available on all file systems, consult the `chattr` man page for more specifics
- Primarily this is only necessary to check if a file is acting irregularly

ls for special cases

- The `ls -l` command will show the following for the special permission bits
  - The setuid bit replaces the owners `x` bit with an `s`
  - The setgid bit replaces the groups `x` bit with an `s`
  - The sticky bit replaces the other's `x` bit with a `t`
  - If these bits are set but the execution bit is not, then it will be `S` or `T`
- The `ls` command lists the number of hard links after the permission bits, the file will only be removed from disk when there are no hard links
- For device files, the file size is replaced with the major and minor device numbers

Access Control Lists (ACL)

- More powerful but more complicated way of regulating access to files
- Two predominant “standards”:
  - POSIX ACL: Primarily extends the normal permissions to be specified for additional numbers of specific users and groups (supported by both Linux and FreeBSD)
  - NFSv4 ACL: Superset of POSIX ACL as well as Windows ACL to work across systems (supported by FreeBSD and indirectly via NFS daemon)
- ACLs should primarily be used when windows support or the level of flexibility needed is beyond UNIX permissions
- ACLs can cause unexpected interactions with ACL-unaware backup systems, network file systems, and other programs

POSIX ACLs

- Allows for the rwx permission bits to be set independently for any combination of users and groups
- Set and queried on files using the `setfacl` and `getfacl` commands, respectively (`ls` will show a `+` after permission bits to indicate an ACL exists)

Entries that can appear in POSIX ACLs

Format	Example	Sets permissions for
user::*perms*	`user::rw-`	The file's owner
user:*username*:*perms*	`user:trent:rw-`	A specific user
group::*perms*	`group::r-x`	The group that owns the file
group:*groupname*:*perms*	`group:staff:rw-`	A specific group
other::*perms*	`other::---`	All others
mask::*perms*	`mask::rwx`	All but owner and other[a]

a. Masks are somewhat tricky and are explained later in this section.

Questions?

Additional Resources

fsck man page

Understanding Symbolic Links

Hard Link and Symbolic Link??

An Advanced Socket Communication Tutorial

chattr man page

HOWTO: Use NFSv4 ACL

Understanding NFSv4 ACL’s