Chapter 2 Linux Filesystem Tree Layout - Notes

2.3 Learning Objectives:

- Explain why Linux requires the organization of one big filesystem tree, and what the major considerations are for how it is done.
- Explain the role played by the Filesystem Hierarchy Standard.
- Describe what must be available at boot in the root (`/`) directory, and what can be available only once the system has started.
- Explore each of the main subdirectory trees, explain their purposes, and examine their contents.

2.4 One Big Filesystem

Linux: consists of one big filesystem tree (similar to all UNIX-based operating systems) -> usually diagrammed as inverted tree with **root directory**, `/` , at top of tree.

May be more than one (or even many) distinct file systems **mounted** at various points within this one large logical filesystem -> appear as subdirectories. Distinct filesystems usually on different partitions, which can also be on any number of different devices, including on network.

Applications: do not care about what physical device files reside on, just looks like one big filesystem.

In past, different UNIX-like OS organized tree in various ways + even various Linux distributions had differences -> resulted in difficulty and frustration writing applications/accomplishing system administration tasks on more than one kind of system

Result -> Linux ecosystem worked hard establish standardized procedures to minimize pain.

2.5 Data Distinctions

Taxonomy for what kind of info has to be read/written when talking about how files/data organized in one big directory tree. Two kinds of distinctions:

1. **Shareable vs non-shareable**: Shareable data can be shared between different hosts, non-shareable data must be specific to particular host. Eg. home directories -> shareable, device lock files -> non-shareable.
2. **Variable vs. static**: static data -> binaries, libraries, documentation + anything that doesn't change without system administrator assistance. Variable data -> anything that may change without systems administrator's help

These logical distinctions embodied as different kinds of information residing in various directories (or partitions and filesystems).

2.6 FHS Linux Standard Directory Tree

Filesystem Hierarchy Standard (FHS): administered by The Linux Foundation (originally by the Free Standards Group), specifies main directories that must be present and their purposes.

Specifies standard layout -> simplifies predictions of file locations.

Most Linux distributions respect FHS, but none follow exactly. Last official version -> old enough, does not take into account some new developments.

Distributions -> like to experiment, and some experiments become generally accepted.

FHS document for more information.

2.7 Main Directory Layout

Linux distributors -> spend lot of time making sure filesystem layout is coherent and evolves correctly over time. List of main directories normally found under `/`:

Directory	In FHS?	Purpose
`/`	Yes	Primary directory of entire file system hierarchy
`/bin`	Yes	Essential executable programs that much be available in **single user mode**
`/boot`	Yes	Files needed to boot system, such as kernel, **initrd** or **initramfs** images, boot configuration files, bootloader programs
`/dev`	Yes	**Device nodes**, used to interact with hardware and software devices
`/etc`	Yes	System wide configuration files
`/home`	Yes	Use **home directories** including personal settings, files, etc
`/lib`	Yes	Libraries required by executable binaries in `/bin` and `/sbin`
`/lib64`	No	64-bit libraries required by executable binaries in `/bin` and `/sbin`, for systems which can run both 32-bit/64-bit programs
`/media`	Yes	Mount points for removable media eg. CDs, DVDs, USB sticks etc
`/mnt`	Yes	Temporarily mounted filesystems
`/opt`	Yes	Optional application software packages
`/proc`	Yes	Virtual pseudo-filesystem giving information about the system and processes running on it. Can be used to alter system parameters
`/sys`	No	Same as `/proc`. Similar to **device tree** and is part of **Unified Device Model**
`/root`	Yes	Home directory for the **root** user
`/sbin`	Yes	Essential system binaries
`/srv`	Yes	Site-specific data served up by system. Seldom used.
`/tmp`	Yes	Temporary files; on many distributions lost by reboot and may be ramdisk in memory
`/usr`	Yes	Multi-user applications, utilities, data: theoretically read-only
`/var`	Yes	Variable data that changes during system operation

May be additional distribution-specific directories found under root directory:

- `/misc`, for miscellaneous data
- `/tfpboot`, for booting using **tftp**. If files in directory, related to diskless workstation booting

Note: does not violate FHS to have other directories, does violate to have components in directories *other* than those dictated by standard.

2.8 The Root (`/`) Directory and Filesystems

There may be multiple partitions/filesystems joined together while entire filesystem can be viewed as one big tree.

Partition and filesystem that contains root directory itself: rather special, often in special dedicated partition. Other components mounted later (`/home` , `/var` , `/opt`)

Root partition: must contain all essential files required to boot system and then mount all other filesystems. Needs utilities, configuration files, boot loader info, other essential startup data. It must be able to:

- Boot system
- Restore system from system backups on external media, eg. tapes, other removable media, NAS etc
- Recover/repair system; experienced maintainer must have tools to diagnose + reconstruct damaged system

FHS: "no application or package should create new subdirectories of the root directory".

2.9 `/bin`

Very important:

- Contains executable programs + scripts needed by both system administrators/unprivileged users, required when no other filesystems have yet been mounted, eg. when booting into **single user** or recovery mode
- May also contain executables which are used indirectly by scripts
- May *not* include any subdirectories

Ubuntu
lsbin

Required programs in `/bin/` :

cat, chgrp, chmod, chown, cp, date, dd, df, dmesg, echo, false, hostname, kill, ln, login, ls, mkdir, mknod, more, mount, mv, ps, pwd, rm, rmdir, sed, sh, stty, su, sync, true, umount, uname [may include **test**]

Optionally may include: **csh, ed, tar, cpio, gunzip, zcat, netstat, ping**

Nonessential command binaries -> `/usr/bin` if not good enough to be placed in `/bin` (eg. programs required only by non-root users)

Note: some recent distributions -> no separation between `/bin` and `/usr/bin` (also `/sbin` and `/usr/sbin`), just one directory with symbolic links (to preserve two directory view). Believe time-honored concept of enabling possibility of placing `/usr` on separate partition mounted after boot -> obsolete.

2.10 `/boot`

Must contain essential files for booting system in `/boot` directory and its subdirectories. Two files, absolutely essential:

- **vmlinuz**: compressed Linux kernel
- **initramfs**: **initial RAM filesystem**, mounted before real root filesystem becomes available

May have longer names, which depend on kernel version (exact form depends on Linux distribution). **initramfs** may be called **initrd**, **initial RAM disk** (older method but name survives).

Stores data used before kernel begins executing user-mode programs. May include: saved master boot sectors, sector map files, other data not directly edited by hand. Exact contents vary by distribution + time.

Before, essential files often placed directly in `/` instead of separate `/boot` directory (following traditional UNIX practices) -> now considered obsolete.

RHEL
lsboot

2.11 Other Files and Directories in `/boot`

Multiple kernel versions available in `/boot`, with four files available for each version (Choice between kernels make by using GRUB at boot time).

Two other files besides **vmlinuz** and **initramfs**:

- **config**: configuration file used when compiling kernel, here just for bookkeeping and reference when debugging
- **System.map**: kernel **symbol table**, very useful for debugging. Gives hexadecimal addresses of all kernel symbols

Neither required for booting or running system.

Distributions may place other files/directories in `/boot`, eg. saved master boot sectors, other data not hand-edited.

2.12 `/dev`

Contains **special device files** (**device nodes**) -> represent devices built into or connected to system. Files essential for proper system function.

Device files represent **character** (byte-stream) + **block I/O** devices.

Network devices -> no device nodes in Linux, instead referenced by name, eg. **eth1**, **wlan0**.

lsdev

2.13 `/etc`

Contains machine-local configuration files and some startup scripts; *no* executable binary programs. Files and directories include:

csh.login, exports, fstab, ftpusers, gateways, gettydefs, group, host.conf, hosts.allow, hosts.deny, hosts.equiv, hosts.lpd, inetd.conf, inittab, issue, ld.so.conf, motd, mtab, mtools.conf, networks, passwd, printcap, profile, protocols, resolv.conf, rpc, securetty, services, shells, syslog.conf.

Configuration files and directories added to `/etc` by distrbutions. Eg. Red Hat: number of other directories (eg. `/etc/sysconfig`, where number of system configuration files live).

Other important subdirectories: `/etc/skel` (contains **skeleton** files used to populate newly created home directories), `/etc/init.d` (contains start up + shut down scripts when using System V initialization)

2.14 `/home`

User directories conventionally placed under `/home` on Linux systems (eg. `/home/coop`, `/home/student`, etc). All personal configuration, data, executable programs contained in `/home` hierarchy. May also contain subdirectories for various groups/associations of users (eg. `/home/students`, `/home/staff`, `/home/aliens`, etc).

Other UNIX-like OS -> concept of `/home` exists, but subtly different. Eg. Solaris: user directories created in `/export/home`, then **automount** facility eventually mount in `/home`. Why -> usual situation: home directory may be anywhere on corporate network

(probably on NFS server), and home directory mounted automatically upon use.

Linux has same **automount** facilities, but many users not aware + on self-contained systems, concept of NFS mounts -> not apply.

User can always substitute environment variable **$HOME** for their root directory, or shorthand ~; so, following commands equivalent:
`shell $ ls -l $HOME/public_html $ ls -l ~/public_html`

One exception: home directory for **root** user in Linux systems *always* found under `/root`. Some older UNIX systems may use `/` instead, causing clutter.

lshome

2.15 `/lib` and `/lib64`

Should contain only libraries needed to execute binaries in `/bin` and `/sbin`. Libraries particularly important for booting system + executing commands within root filesystem.

Kernel modules (often device/filesystem drivers) located under `/lib/modules/<kernel-version-number>`.

PAM (**P**luggable **A**uthentication **M**odules) files stored in /lib/security.

Systems supporting both 32-bit/64-bit binaries must keep both libraries on system. On Red Hat-based systems, separate directories for 32-bit (`/lib`) and 64-bit (`/lib64`) libraries.

2.16 `/media`

Typically used to mount filesystems on removable media, eg. CDs, DVDs, USB drives, floppies (heh).

Modern Linux systems: mount media dynamically upon insertion. **udev** creates directories under `/media` + mounts removable filesystems there (names set with **udev** rules specified in configuration files). Directories used as mount points under `/media` -> disappear upon unmounting + removal.

If more than one partition and filesystem on media -> more than one entry on `/media`. On many Linux distributions, file manager (eg. Nautilus) pops up upon media mounting.

Note: removable media pop up under `/run/media/[username]...` for some newer distributions (eg. SUSE, RHEL 7). `/run` discussed later.

2.17 `/mnt`

Provided so system administrator can temporarily mount filesystem when needed. Common use: network filesystems:

- **NFS**
- **Samba**
- **CIFS**
- **AFS**

Old systems used `/mnt` for files now mounted under `/media` (or `/run/media`) in modern systems

Generally, should *not* be used by installation programs. Another temporary directory not currently used serves better.

2.18 `/opt`

Designed for software packages that wish to keep all/most files in one isolated place, rather than scatter across system in directories shared by other software. Eg. package name **dolphy_app**: all files reside in directories under `/opt/dolphy_app`, including `/opt/dolphy_app/bin` for binaries, `/opt/dolphy_app/man` for **man** pages.

Makes installing/uninstalling software relatively easy -> all files in one convenient isolated location in predictable + structured manner. Also easy for system administrators to determine nature of each file within package.

Note: also easy to install/uninstall with clear sense of manifests/locations without antisocial behavior if using packaging systems eg. **RPM**, **APT**.

In Linux, `/opt` directory often used by proprietary software providers, or those who like to avoid distribution variance complications. Eg. `/opt/skype`, `/opt/google` (containing `chrome`, `earth`, `talkplugin`).

Reserved for local system administrator use: `/opt/bin`, `/opt/doc`, `/opt/include`, `/opt/info`, `/opt/lib`, `/opt/man`. Packages must be able to function without programs being in these special directories (but may also provide files linked/copied to these reserved directories).

2.19 `/proc`

Mount point for **pseudo-filesystem**, where all info resides only in memory (not on disk). `/proc` directory empty on non-running system (like `/dev`).

kernel -> exposes some important data structures through `/proc` entries. Each active process on system -> has own subdirectory, gives detailed info about state of process, used resources, history.

`/proc` entries -> often termed **virtual files**, have interesting qualities. Most listed as zero bytes in size, but contain large amount of info when viewed.

Most virtual files time/date settings -> current time/date, reflects constant change. Info in file *only* obtained when viewed, not constantly/periodically updated.

Important pseudo-files provide up-to-date moment glimpse of system's hardware, eg. `/proc/interrupts`, `/proc/meminfo`, `/proc/mounts`, `/proc/partitions`

Also provide system configuration info + interface, eg. `/proc/filesystems`, `/proc/sys`.

Files containing info on similar topic grouped in virtual directories/sub-directories, eg. `/proc/scsi` -> info for all SCSI devices, **process** directories -> info about each running process on system

Entries in `/proc` examined throughout course -> detailed look for kernel configuration + system monitoring.

lsproc

2.20 `/sys`

Mount point for **sysfs pseudo-filesystem** where all info resides in memory, not on disk. Empty on non-running system (like `/dev`, `/proc`).

sysf -> used to gather info about system, modify behavior while running. Resembles `/proc`, but younger + adheres to strict standards about entries contained, eg. almost all pseudo-files in `/sys` contain only one line or value; no long entries like found in `/proc`.

Entries in `/sys` also examined throughout course.

lssys

2.21 `/root`

Home directory for root user (pronounced "slash-root")

Root account that owns `/root` -> *only* use for actions requiring superuser privilege. Use another account for non-privileged user actions.

lsroot

2.22 `/sbin`

Contains binaries essential for booting, restoring, recovering, repairing (in addition to binaries in `/bin`). Must also be able to mount other filesystems on `/usr`, /home, other locations if needed (once root filesystem known to be in good health during boot).

Included programs (if subsystems installed):

fdisk, fsck, getty, halt, ifconfig, init, mkfs, mkswap, reboot, route, swapon, swapoff, update.

Note: some recent distributions -> no separation between `/sbin` and `/usr/sbin` (also `/bin` and `/usr/bin`), just one directory with symbolic links (to preserve two directory view). Believe time-honored concept of enabling possibility of placing `/usr` on separate partition mounted after boot -> obsolete.

lssbin

2.23 `/srv`

Contains site-specific date, served by this system.

Main purpose of specifying this: users may find location of data files for particular service, services which require single tree for read-only data writable data, scripts (eg. **cgi** scripts) reasonable placed.

Methodology of `/srv` subdirectory naming: unspecified, currently no consensus. One method for structuring data: by protocol, eg. **ftp, rsync, www**, and **cvs**.

Often confusion about what is best to go in `/var` vs. `/srv`. Some system administrators rely heavily on `/srv`, others ignore.

2.24 `/tmp`

Used to store temporary files, can be accessed by any user of application. Files on `/tmp` cannot be depended to stay around long-term. Some distributions:

- Run automated **cron** jobs -> remove any file older than 10 days typically, unless purge scripts modified to exclude files. RHEL 6 policy.
- Remove contents of `/tmp` with every reboot. Ubuntu policy.
- Modern: utilize virtual filesystem, using `/tmp` only as mount point for **ram disk** using **tmpfs** filesystem. Default policy of recent Fedora systems. All info lost during system reboot; `/tmp` indeed temporary.

In last case, avoid creating large files on `/tmp`: occupy memory instead of disk, easy to hard/crash system through memory exhaustion. Guideline: avoid large files in `/tmp`, but plenty of applications violating policy. Possible to put them elsewhere (eg. by specifying environment variable), but many users not aware of how to configure + all users have access to `/tmp`.

Can cancel policy on systems using **systemd**, eg. Fedora, by:
`shell $ sudo systemctl mask tmp.mount`

followed by system reboot.

2.25 `/usr`

Can be thought of as **secondary hierarchy**. Used for files not needed for system booting. Need not reside in same partition as root directory, can be shared among hosts using the same system architecture across network.

Software packages -> should *not* create subdirectories directly under `/usr`. Some symbolic links to other locations may exist for compatibility purposes.

Typically read-only data. Contains binaries not needed in single user mode. Contains `/usr/local` (local binaries and such), `/usr/share/man` (**man** pages)

Note: some recent distributions -> no separation between `/bin` and `/usr/bin` (also `/sbin` and `/usr/sbin`), just one directory with symbolic links (to preserve two directory view). Believe time-honored concept of enabling possibility of placing `/usr` on separate partition mounted after boot -> obsolete.

Directories under `/usr`:

Directory	Purpose
`/usr/bin`	Non-essential binaries and scripts, not needed for single user mode. Generally, means user applications not needed to start system
`/usr/etc`	Non-essential configuration files (usually empty)
`/usr/games`	Game data
`/usr/include`	Header files used to compile applications
`/usr/lib`	Library files
`/usr/lib64`	Library files for 64-bit
`/usr/local`	Third-level hierarchy (for machine local files)
`/usr/sbin`	Non-essential system binaries
`/usr/share`	Read-only architecture-independent files
`/usr/src`	Source code and headers for the Linux kernel
`/usr/tmp`	Secondary temporary directory

2.26 `/var`

Contains **variable** (or **volatile**) data files that change frequently during system operation, including:

- Log files
- Spool directories, files for printing, mail queues, etc
- Adminsitrative data files
- Transient/temporary files, eg. cache contents

Cannot be mounted as read-only filesystem (obviously).

Often considered good idea to mount `/var/` as separate file system for security reasons. If directory filled up, should not lock up system.

`/var/log` directory -> most log files located, `/var/spool` directory -> local files for processes such as mail, printing, cron jobs stored while awaiting action.

Directory	Purpose
`/var/ftp`	Used for ftp server base
`/var/lib`	Persistent data modified by programs as they run
`/var/lock`	Lock files used to control simultaneous access to resources
`/var/log`	Log files
`/var/mail`	User mailboxes
`/var/run`	Information about the running system since the last boot
`/var/spool`	Tasks spooled or waiting to be processed, eg. print queues
`/var/tmp`	Temporary files to be preserved across system reboot, sometimes linked to `/tmp`
`/var/www`	Root for website hierarchies

lsvar

2.27 `/run`

Not formally accepted by FHS, although proposed. New directory tree mounted at `/run` in use for several years by major Linux distributions.

Purpose: store **transient** files: containing runtime information, need to be written early in system startup, do not need to be preserved when rebooting.

Implemented as empty mount point with **tmpfs** ram disk (like `/dev/shm`) mounted there at runtime. Pseudo-filesystem existing only in memory.

Some existing locations (`/var/run`, `/var/lock`) just symbolic links to directories under `/run`. Other locations, depending on distribution taste, may also just point to locations under `/run`.

lsrun

##

Back to top

Previous Chapter - Table of Contents - Next Chapter