Linux Filesystem Structure *v1.0*

Daniel Quinlan, quinlan@bucknell.edu

FSSTND (Filesystem Standard) Group

ABSTRACT

The open and distributed process in which the Linux operating system has developed fosters rapid growth of the operating system, applications, and integrated distributions. This decentralized process, however, has created a need for standardizing the structure of the filesystem. This standard aims to define locations and specifications for files and directories in Linux systems. This will allow users, developers, and distributors to assemble parts of the system from various sources that will work together as smoothly as if they had been developed under a monolithic development process. It will also make general documentation less difficult, system administration more consistent, and development of second and third party packages easier.

Introduction

Status of the Standard

This proposed standard is being distributed to members of the Linux community in order to solicit their reactions to the series of ideas, concepts, and recommendations included within it. While the entire content of this standard may not be identical to what every individual desires, it has proven to be a good start towards solving many problems.

The guidelines in this standard are subject to change. Use of information contained in this document is at your own risk.

Organization of the Standard

This standard is divided into 6 parts:

- General, including a statement of scope, problems, objectives, and conformance requirements. (Section 1)
- The filesystem: a statement of some guiding principles. (Section 2)
- The root directory. (Section 3)
- The /usr directory. (Section 4)
- The /var directory. (Section 5)
- Issues and Additional Rationale. (Section 6)

Conventions in this Document

Typographical Conventions:

Courier font is used within directory and filename tables. Any filename that lacks a directory prefix ("/") and is not part of a table is enclosed in single quotes.

Variable directory names (and email addresses) are preceded by "<" and ended with ">". Variable substrings of directory names and filenames are indicated by "*".

Note: the conventions used within this document will be extended and clarified extensively in the next revision.

Brief glossary of terms:

distribution: a pre-packaged release of Linux. Examples include the MCC-Interim release, the Debian release, and the Slackware release.

filesystem: 1. a single formatted partition. 2. the entire directory hierarchy.

package: a group of files assembled for a specific function.

program: any binary or shell script that is executed within a shell.

root directory: the base of the directory tree and filesystem.

1. General

1.1. Scope

This standard defines a standard filesystem structure for Linux systems, including the placement of directories, files, and the contents of some system files.

The filesystem standard has been designed to be used by Linux distribution developers, package developers, and system implementors. However, it is intended to be a reference and not a tutorial on how to manage a Linux filesystem or directory hierarchy.

These are some of the fundamental problems that originally motivated this standardization effort:

- There was no single well accepted Linux directory structure. Instead, there were many different ones, each being incompatible with each other.
- In the most widely used filesystem hierarchies, the directories were not well structured and differed gratuitously from more modern directory structure "standards" (such as System V, SunOS, BSD, and others).
- The filesystem was unfamiliar and discomforting to experienced UNIX users and administrators who have experience on more mainstream UNIX systems.
- The lack of regularity was also confusing for newcomers to Linux, especially those coming from a non-UNIX background.
- Any incompatibilities between primary Linux distributions and other software packages were typically solved by methods of a less than appealing nature.
- Overall, symbolic links were used much too often within the filesystem to fix problems. (However, there are times when symbolic links need to be used to ensure backward compatibility or to allow specific systems to have an individual filesystem structure.)

It is to be expected that differences in opinion are to arise in any standardization effort such as this one. However, these differences should be overshadowed by the need for consensus and common practice within the Linux community.

This filesystem standard was primarily developed within the FSSTND mailing list and previously, the FSSTND channel of the Linux-activists mailing list. Input and comments were received from a great number of Linux developers, noted Linux programmers, system administrators, and users. Those volunteers who have contributed extensively to this standard are listed at the end of this document. This standard represents the consensus view of those and other contributors.

This standard seeks to address these problems by describing a well designed filesystem structure that the Linux community may voluntarily follow. Although this standard is more comprehensive and complete than any other previous attempt at standardization, it will probably never be truly finished. The needs of the Linux community will continually change in relation to emerging technology. It is also possible that better solutions to filesystem problems could be discovered or that these solutions will no longer be the best possible solutions. Because of these reasons, the FSSTND group plans to release supplementary drafts in addition to periodic updates to this document.

If you have any comments related to this standard, they are welcomed by the FSSTND group. Suggestions for changes to this document should be directed to the FSSTND mailing list, the FSSTND coordinator, or if you prefer, any of the listed contributors. Typographical or grammatical comments should be directed to the FSSTND coordinator.

Please do not send mail to the mailing list without first contacting the FSSTND coordinator or a listed contributor. Improper messages will not be well received on the mailing list.

Questions about how to interpret items in this document may occasionally arise. If you have need for a clarification, please contact the FSSTND coordinator. Since this standard represents the consensus of many participants, it is important to make certain that any interpretation also represents their collective opinion. Due to this reason, it may not be possible to provide an immediate response unless the inquiry has been the subject of previous discussion.

The FSSTND coordinator is Daniel Quinlan <quinlan@bucknell.edu>.

1.2. Specific Problems

Naturally, while defining a Linux filesystem structure, there were some specific problems that we sought to correct. Here are some of the major and well accepted ones:

- The primary binary directories, /bin and /usr/bin, do not have well defined divisions between them. As a result, the distribution of the binaries over these two directories varies greatly between the Linux distributions.
- Including both binaries and configuration files in /etc makes it more confusing and harder to maintain for inexperienced users or system administrators with especially large systems.
- The division between what is a site-wide configuration file and what is a machine-local configuration file is difficult to establish. This makes determining whether a configuration file is stored in /etc or /usr/etc difficult.
- The current implementation of /usr cannot be mounted read-only because it contains variable files and directories that need to be written to.
- In a networked environment it is desirable to serve software to workstations via a NFS mounted filesystem. Such filesystems need to be mounted read-only so that accidents or malice on one workstation cannot damage the files on the server. This requires identification and separation of files that a machine must write to and of files that are specific to a single machine
- Linux is not well prepared for a network installation including the possibility of a read-only /usr partition and diskless workstations.

While these are some of the major problems we addressed, there were numerous additional problems that needed to be solved. This standard attempts to address many of those other problems, but there may be something that was overlooked. If you wish to bring something to our attention, please note there are some things that have been discussed at length, but were not included in this standard.

1.3. Objectives

In trying to solve the above problems, several objectives were identified that needed to be realized in addition to the more technical matters. These goals comprise the correction of outstanding problems as well as the validation of this standard.

- Solve the above problems while also limiting the possible transition difficulties resulting from moving away from the former de-facto standards.
- Gain approval of distributors, developers, and other important people in the Linux community, as well as their suggestions.
- Provide a standard that all of the Linux community would choose to follow because it will solve the above problems as well as provide the most sensible structure for Linux's filesystem.

Some of these objectives have already been fully or partially met due to the limited distribution of advance drafts to any developer who requested one.

1.4. History and Progress

The original message that motivated this effort to restructure the Linux filesystem was written by Olaf Kirsh <okir@monad.swb.de> on August 2, 1993, to the NORMAL channel of the Linux activists mailing list.

Soon thereafter, it was decided that the best possible way to accomplish such a restructuring of the Linux filesystem would be to create a mailing list for the purpose of trying to develop a consensus standard.

After a comprehensive discussion, with surprisingly few flames, a preliminary draft was written. Then, with the help of several dedicated people, the draft was finished and that resulting draft submitted to the FSSTND channel for more discussion. The first draft was submitted to the channel on September 18, 1993, by Daniel Quinlan.

As the discussion continued and early drafts of FSSTND recommendations were developed further, contacts were established with accessible Linux distributors who then offered their input and support to our effort. Many Linux developers already agree that what we are doing is a good thing. These are some of the developers who are trying to follow the FSSTND standard [alphabetical order]:

- Debian Linux
  Ian A. Murdock <imurdock@shell.portal.com>
- LILO
  Werner Almesberger <almesber@nessie.cs.id.ethz.ch>
- Linux/PRO
  Fred N. van Kempen <waltje@metallica.uwalt.nl.mugnet.org>
- Slackware Linux
  Patrick J. Volkerding <volkerdi@mhd1.moorhead.msus.edu>
- TAMU Linux
  Dave Safford <dave.safford@net.tamu.edu>
- util-linux packages
  Rik Faith <faith@cs.unc.edu>

1.5. Conformance with this Document

Only items which are explicitly labeled as "recommended" or "optional" are not required for a system to be conformant. Any item that is recommended is not absolutely necessary on a local level, but is required for a distribution or program to be considered conformant. Some specifications are optional, but if distributors follow these sections it is likely to be of benefit to the Linux community.

There are three levels of compliance with this standard. A fully conforming distribution shall meet all recommendations and requirements in each section of this document. A fully conforming application or program shall meet all applicable recommendations and requirements. Sections or statements that are marked optional are not required for full conformance. Partial conformance indicates that an distribution conforms with a significant subset of this document. Partial conformance is only intended to apply for distributions and not separate programs. The phrase "a significant subset" is admittedly subjective, and in borderline cases, the concerned party should contact the FSSTND coordinator. It is anticipated that some variation will be tolerated in borderline cases.

Compatibility with this standard indicates that every file in the distribution can be found at the point at which it would be found on a fully compliant distribution, though this is not necessarily the file's primary location.

In order to qualify as partially FSSTND conformant or FSSTND compatible, however, a distribution must provide a list of all places at which it and the FSSTND document differ, and a brief explanation of the reason for this difference. This list shall also be provided to the FSSTND mailing list or the FSSTND coordinator.

Note: this will all be clarified in the next revision of this document. There are no plans to make conformance any more difficult.

2. The Filesystem

The UNIX filesystem is characterized by:

- A hierarchical structure
- Consistent treatment of file data
- Protection of file data

This standard on the Linux filesystem follows the same basic principles that most UNIX filesystems follow. Note that this standard does not attempt to conform in every possible respect to any particular UNIX system's implementation. However, many aspects of this standard are based on ideas found in UNIX and other UNIX-like systems.

This is after careful consideration of other factors, including:

- Common (good) practices in the Linux community
- The existence of other filesystem structures
- Applicable standards

The directory hierarchy is separated into unsharable data, "/" (root), and sharable data, "/usr". These two filesystems are divided according to the following two data types: static data and variable data. Static data includes binaries, libraries, documentation, and anything that does not change without system administrator intervention. Variable data is anything that does change without the system administrator's intervention.

It is possible to define two orthogonal categories of file data: sharable-unsharable and variable-static. Each hierarchy, / and the traditional /usr, contains both static and variable data. The /usr hierarchy is further defined as being sharable data and / as unsharable data. For reasons mentioned above and below, we will redefine the contents of /usr to be <u>only</u> shareable and static data, containing no variable data.

Throughout this document, and in any well planned filesystem, an understanding of these basic principles will help guide the structure and lend it additional consistency.

The distinction between sharable and unsharable data is needed for several reasons:

- In a networked environment (i.e. more than one host at a site), there is a good deal of shareable data that can be shared between different machines to save space and ease the task of maintenance.
- In a networked environment, certain filesystems contain information specific to a single machine. Therefore these filesystems cannot be shared (with NFS).
- The current implementation of /usr cannot be mounted read-only because it contains variable files and directories that need to be written to. This is a factor that must be addressed when /usr is shared on a network or mounted read-only because of other considerations (primarily safety).

The "sharable" factor can be extended in two directions:

- A /usr mounted (read-only) through the network (using NFS).
- A /usr mounted from read-only media. (You can think of that CD-ROM drive as a networked, using postal mail, filesystem that you are sharing with other Linux systems.)

The "static" versus "variable" factor dramatically affects the filesystem in 2 major ways:

- Since / contains both variable and static data, it needs to be mounted read-write.
- Since the traditional /usr contains both variable and static data, and since we want to mount it read-only (see above), it is necessary to provide a method to have /usr mounted read-only. This is done through the creation of a /var hierarchy that is mounted read-write (or is a part of another read-write partition, such as /), taking over much of the /usr partition's traditional functionality.

3. The root directory

This section describes the root directory structure. Enough data should be contained in the root partition to boot, restore, recover, and/or repair the system:

(1) To boot a system, enough must be present to mount /usr. This includes utilities, configuration, boot loader information, and other essential start-up data.

(2) To recover and/or repair a system, those utilities needed by an experienced user to diagnose and reconstruct a damaged system should be present on root.

(3) To restore a system, those utilities needed to restore a system from backups (on floppy, tape, etc.) should be present on root.

The primary concern used to balance these desires (placing many things in root) is the goal of keeping root as small as reasonably possible. It is desirable to keep root small in terms of number of directories, files, and disk space for several reasons:

(1) The root is often mounted from very small media. For example, most Linux users install and do recovery by mounting root off a RAM disk which is copied from a single 1.44M or 1.2M floppy disk.

(2) Root has many system-specific configuration files in it, a kernel that is specific to the system, a different hostname, etc. This means that root isn't always sharable between networked systems. Keeping root small on networked systems minimizes the amount of space lost on servers to unsharable files. It also allows workstations with smaller local hard drives.

However, with diskless clients, this does not have to be entirely the case, unless each client has a different root image.

(3) While you may have a large root partition, and may be able to fill it to your heart's content, there will be people with smaller partitions. If you have more files installed, you may find incompatibilities with other systems using limited root partitions. If you are a developer then you may be sharing this problem with a large number of users.

(4) Disk errors on the root partition are a greater problem than errors on any other partition. A small root partition is less prone to corruption as the result of a system crash.

Since root is small and host-specific (due to the division between / and /usr), this scheme necessitates a writeable root. However, this does not necessitate a fully locally stored root. The root partition doesn't have to be locally stored just to be system specific (i.e., root mounted from a NFS root server.)

Software should never create or require special files or subdirectories in the root directory. This structure provides more than enough flexibility for any package. Any package that does occupy a directory under root suffers from sheer arrogance.

```
/ : the root directory
  |
  |- bin        : essential command binaries
  |- boot       : static files of the boot loader
  |- dev        : device files
  |- etc        : machine-local system configuration
  |- home       : user home directories
  |- lib        : shared libraries (libc.so.*, libm.so.*, and ld.so)
  |- mnt        : mount point of temporary partitions
  |- proc       : process information pseudo-filesystem
  |- root       : home directory for root
  |- sbin       : essential system binaries
  |- tmp        : temporary files
  |- usr        : second major permanent mount point
  |- var        : files that tend to grow or vary in size
  +- {kernel image}
```

Following this section, each directory is presented in a separate subdivision. /usr and /var are discussed in separate sections of this document.

The root directory normally contains the current kernel image. The kernel image name is locally configurable, but the name we suggest (that has been used in recent Linux kernel sources) is ‘vmlinux’. This may or may not be a (symbolic-)link to the actual file, possibly depending on the system distribution used. Additional information on kernel placement is located in the /boot section of the standard.

3.1. /bin : essential command binaries (for use by all users)

There should be no subdirectories within /bin.

The commands (static data) that are needed in single user mode by the super-user (root) are stored in /bin. However, the commands in /bin are for use by <u>both</u> root and other users. On the same note, the /bin directory should not contain anything that is to be used only by root for system administration.

All root-only binaries such as standard daemons, ‘init’, ‘getty’, ‘update’, et al. (previously found in /etc), shall now be placed in /sbin or /usr/sbin depending on the necessity of the command. For further discussion of the definition of what is essential in the filesystem, please read section 6, "Issues and Additional Rationale".

Command binaries that are not essential enough to place into /bin should be placed into /usr/bin, instead. Items that are only needed by non-root users (‘mail’, ‘chsh’, ‘passwd’, et al.) are not vital to system operation and should not be placed in the root partition.

REQUIRED files for /bin:

• general commands:

The following commands have been added because they are essential. A few have been added because of their traditional placement in /bin.

```
{ arch, cat, chgrp, chmod, chown, cp, date, dd, df, dmesg, echo, ed,
  false, kill, ln, loadkeys, login, ls, mkdir, mknod, more, mount,
  mv, ps, pwd, rm, rmdir, setserial, sh, stty, su, sync, true,
```

`umount, uname` }

If /bin/sh is BASH, then /bin/sh should be a symbolic or hard link to /bin/bash since bash behaves differently when called as ‘sh’ or ‘bash’. The same goes for ‘pdksh’, which is often the /bin/sh on install disks and such (link from /bin/sh to /bin/ksh). I personally prefer the advantage of using a symbolic link in these cases, because it allows users to easily see that /bin/sh is not a true Bourne shell.

Both ‘[’ and ‘test’ are built into BASH, pdksh, zsh, recent Korn shells, and essentially every Bourne shell replacement there is for Linux. They should be placed into /usr/bin. (They must be included as separate binaries with any Linux system attempting to comply with the POSIX.2 standard.)

/bin/arch should produce the same output as "uname -m", specifically "i386" or "i486" for Intel or Intel-compatible systems.

• restoration commands:

These commands have been added to make restoration of a system possible (provided that / is intact).

{ `tar, gzip, gunzip (link to gzip), zcat (link to gzip)` }

If system backups are made with a package other than ‘gzip’ and ‘tar’, then that administrator should include the minimal necessary restoration components in the root partition. For instance, many systems should include ‘cpio’ as it is the next most commonly used backup utility after tar. Conversely, if no restoration from the root partition is ever expected, then these binaries may be omitted (e.g., a ROM chip root, mounting /usr through NFS). If restoration of a system is planned through the network, then ftp or tftp (along with everything necessary to get a ftp connection) should be available on the root partition.

• networking commands:

These are deemed the only necessary networking binaries that both root and users will want or need to execute other than the ones in /usr/bin or /usr/local/bin.

{ `domainname, hostname, netstat, ping` }

3.2. /boot : static files of the boot loader

This directory contains everything for boot except configuration files and the map installer. In the simplest sense, /boot is for anything which is used before the kernel execs /sbin/init. This includes saved master boot sectors, sector map files, and anything else that is not directly edited by hand. The boot loader program should be placed into /sbin and configuration files for boot loaders into /etc.

More specifically, for LILO:

```
Old location                 New location
------------------------     -----------------
/etc/lilo/config.defines     /etc/lilo.defines
/etc/lilo/config             /etc/lilo.conf
/etc/lilo/disktab            /etc/disktab
/etc/lilo/lilo               /sbin/lilo
/etc/lilo/boot.NNNN          /boot/boot.NNNN
/etc/lilo/part.NNNN          /boot/part.NNNN
/etc/lilo/map                /boot/map
/etc/lilo/*.b                /boot/*.b
```

"*.b" are the first and second stage boot loader, plus all those chain loaders. ‘QuickInst’ (if it is included) should be placed into /usr/sbin. (The ‘activate’ command is left out of this scheme because its future is uncertain at this time.) Use of LILO is not required for conformance to this standard, but it is recommended that it be used by Linux distributions as the default Linux boot loader.

Extra kernel images may be stored in /boot. The main kernel can either be placed in / or in /boot according to preference of the administrator. If placed in /, the kernel may also possibly be a symlink to a kernel image in /boot. Note that the standard location for the kernel is still in /.

3.3. /dev : Device files

/dev usually also contains a file, ‘MAKEDEV’, a shell script designed to create devices as needed. It also often contains a ‘MAKEDEV.local’ for any local-only devices.

‘MAKEDEV’ should have provisions for creating any device special file listed in the major/minor numbers list, not just those that a particular distribution installs.

Symbolic links in /dev should not be distributed with Linux systems, as the local setup will often differ from that on the distributor’s development machine. Also, if a distribution install script configures the symlinks at install time, these symlinks will often not get updated if local changes are made in hardware. When used responsibly at a local level, however, they can be put to good use.

This standard incorporates the Linux Device List which is maintained by Rick Miller <rick@ee.uwm.edu>, the Linux Device Registrar. This is the current standard that we strongly recommend using. It is available by anonymous ftp at sunsite.unc.edu as "/pub/Linux/docs/hardware/DEVICES".

3.4. /etc : Machine-local system configuration

No binaries should go directly into /etc. Binaries that would have, in the past, been found in /etc should now be placed in /sbin. This includes such files as init, getty, and update. Binaries such as hostname that are used by users as well as root should not be placed in /sbin, but in /bin.

```
/etc : Machine-local system configuration
     |
     |- X11         : Configuration for the X Window system
     |- keytables   : Keyboard translation tables
     +- skel        : Skeleton user configuration
```

/etc/X11 is the recommended location for all X11 machine-local configuration. This is necessary to allow for local control and read-only mounting of /usr. Files that should be in here include ‘Xconfig’, ‘xdm-config’, with possible subdirectories for window managers, and other files that are considered to be configuration such as ‘Xmodmap’, ‘Xserverrc’, ‘Xinitrc’, ‘Xresources’, etc.

/etc/keytables is the location for all keyboard translation tables used by loadkeys(1). This is required on the root partition so that all keyboard types can be supported (even in emergencies). It is possible that unneeded keyboard tables may be stored off of the root partition to save space. The recommended location in that case is /usr/lib/keytables.

/etc/skel is the location for so-called "skeleton" user files that are given by default to new users when receiving an account. This directory often contains subdirectories for different user groups (e.g., /etc/skel/staff or /etc/skel/users). Note that some people prefer to place a "skel" subdirectory in each directory of user accounts. Either of the two approaches is acceptable at this time.

This is a listing of the general files that are placed directly into /etc. It is not exhaustive. Also, some of the required files are not technically essential. For instance, if a system doesn't have printing capability, then the 'printcap' file is not needed.

REQUIRED files for /etc:

• general files:

  These files are needed on most Linux systems.

  { `adjtime, csh.login, fdprm, fstab, gettydefs, group, inittab, issue, ld.so.conf, magic, motd, mtab, mtools, passwd, profile, securetty, shells, termcap, ttytype, utmp` }

  The LILO configuration file, 'lilo.conf', is required if LILO is used as the system's boot manager.

• networking files:

  These files should be installed on most Linux systems.

  { `exports, ftpusers, gateways, hosts, host.conf, hosts.equiv, hosts.lpd, inetd.conf, networks, printcap, protocols, resolv.conf, rpc, services` }

There are two models for setup of the "rc" command scripts which are invoked by init(8) at boot time, the /etc/rc.* (BSD model) and the /etc/rc.d/* (System V model). Either model is acceptable at this time. This is a local issue and the implementation that is used should be determined by the system administrator or the developer.

Note: True System V systems (and the Linux "bootsys" package) use seperate rc.d directories per run level. The Linux "SysVinit" package uses a (well accepted) mixture of the System V and BSD implementations.

The 'wtmp' log file belongs in /var/adm because it can grow in size without bound.

Systems that use the shadow password suite will have additional configuration files in /etc (/etc/shadow and others) and /usr/sbin ('useradd', 'usermod', et cetera).

Systems that use "getty_ps" will have a few additional configuration files in /etc and /etc/default. Since getty_ps is the only program which makes use of /etc/default, it is recommended that /etc/getty_ps be used instead.

3.5. /home : User home directories (optional)

/home is a fairly standard concept, but it is clearly a site-specific filesystem. The setup will differ from machine to machine. This is only a suggested placement for user home directories, but we recommend that all Linux distributions use this as the default home directory.

On small systems, each user's directory is typically one of the many subdirectories of /home such as /home/smith, /home/linus, /home/operator, etc.

On large systems (especially when the /home directories are mounted across a number of machines using NFS) it is a good idea to subdivide user home directories. Subdivision can be accomplished by using subdirectories such as /home/staff, /home/guests, /home/students, etc.

Different people prefer to place user accounts in a variety of places. Therefore, no programming should rely on this location. If you want to find out a user's home directory, you should use the getpwent() library function rather than relying on /etc/passwd because user information may be stored remotely.

3.6. /lib : Shared libraries (needed to run dynamically linked binaries)

Only the shared library images necessary to boot the system should be placed in /lib. The shared library images are "/lib/libc.so.*", "/lib/libm.so.*", and "/lib/ld.so" (and not the actual ".a" files).

Shared libraries that are only necessary for binaries in /usr (such as any X Window binaries) do not belong in /lib. Only the shared libraries required to run binaries in /bin and /sbin should be here.

For compatibility reasons, /lib/cpp needs to exist as a reference to the C preprocessor installed on the system. The usual placement of this binary is /usr/lib/gcc-lib/<target>/<version>/cpp. /lib/cpp can either point at this binary, or at any other reference to this binary which exists in the filesystem. (For example, /usr/bin/cpp is also often used.)

3.7. /mnt : Mount point for temporarily mounted filesystems

This is the location where the system administrator may temporarily mount filesystems as needed. The setup of this directory is a local issue and should not affect the manner in which any program is run.

This directory should probably not be used by installation programs due to the site-specific nature of it. A temporary directory not in use by the system should be used instead.

3.8. /proc : Proc based process system

The procps filesystem is already becoming the de-facto standard Linux method for handling process information rather than /dev/kmem and other methods. This is strongly recommended as it should become the standard for the storage and retrieval of process information as well as other kernel and memory information.

3.9. /root: home directory for root (optional)

/ is traditionally the home directory of the root account on UNIX systems. /root is used on many Linux systems and on some UNIX systems. The root home directory is determined by developer or local preference. /, /root, and /home/root are all acceptable locations as far as this standard is concerned.

If the home directory for root is stored on a different partition than the root partition, it is necessary to make certain that 'login' will default to / if root's home directory can not be located.

Note: It is preferred practice not to use the root account for mundane things such as mail and news, but solely for systems administration. For this reason, subdirectories such as "Mail" and "News" should not appear in the root account's home directory. Mail is usually forwarded to a more appropriate account.

3.10. /sbin : System binaries (binaries once kept in /etc)

Utilities used for system administration (and other root-only commands) are stored in /sbin, /usr/sbin, and /usr/local/sbin. /sbin typically contains files essential for the booting phase of starting the system up. Anything executed after /usr is known to be mounted (when there are no problems) should be placed into /usr/sbin. Local-only system administration stuff should now be placed into /usr/local/sbin.

Deciding what things go into "sbin" directories is simple: If a user will need to run it, then it should go somewhere else. If it will only be run by system administrators or as root from system management scripts, then it should go in /sbin (or in /usr/sbin or /usr/local/sbin it the item is not vital to system operation).

Files such as 'chfn' which users only occasionally use should still be placed in /usr/bin. 'ping', although it is absolutely necessary for root (network recovery and diagnosis) is often used by users and should live in /bin for that reason.

Ordinary users should not have to place any of the "sbin" directories in their path.

Users should have execute permission for everything in /sbin that can be executed by non-root users. The division between /bin and /sbin was not created for security reasons or to prevent users from seeing the OS, but to provide a good division between binaries that everyone uses and the ones that are primarily used for administration tasks. There is no inherit security advantage in making /sbin off-limits for users.

REQUIRED files for /sbin:

- general commands:
  { `clock, getty, init, update, mkswap, swapon, swapoff` }
- shutdown commands:
  { `halt, reboot, shutdown` }
- filesystem commands:
  { `fdisk, fsck, fsck.*, tunefs (Ext2 only), mkfs, mkfs.*` }
  "*" = (ext, ext2, minix, msdos, xiafs)
- LILO commands:
  { `lilo` }
- networking commands:
  { `arp, ifconfig, ifsetup, route` }

OPTIONAL files for /sbin:

- static binaries:

  Static ln and static sync are useful when things go wrong. The primary use of sln (to repair hosed symlinks in /lib after a poorly orchestrated upgrade) is no longer a major factor now that the 'ldconfig' program exists and can act as a guiding hand in upgrading the dynamic libraries. Static sync is useful in some emergency situations. Note that these need not be statically compiled versions of the standard 'ln' and 'sync', but may be.

  The 'ldconfig' binary is optional for /sbin as well, since if you use it properly when upgrading the shared libraries in /lib, you won't need it in root.

{ `ldconfig, sln, ssync` }

• miscellaneous:

Due to the fact that some keyboards come up with such a high repeat rate as to be unusable, ‘kbdrate’ is necessary on some systems.

{ `kbdrate` }

3.11. /tmp : temporary files

/tmp is used for temporary files, preferably on a fast device (a memory based filesystem, for instance).

The "persistence" of the data that is stored in /tmp is different from that which is stored in /var/tmp. /tmp is usually cleaned out at boot time (or at relatively frequent intervals). Because of this, data stored in /tmp should not be expected to remain for any long period of time on the system... it is frequently deleted.

Programs should use /tmp or /var/tmp (which was originally /usr/tmp) according to the expected persistence of the data, but should not rely on any particular persistence for any tmp directories.

The precise arrangement of /tmp and /var/tmp is a local issue. If there are distinct /var/tmp and /tmp directories, then the persistence of /var/tmp files should be at least as long as for /tmp, but beyond that, a site administrator or distributor can arrange this any way they want, over RAM disk, symlinks, or whatever.

4. The /usr directory

/usr is the second major division of the filesystem. /usr is sharable data. That means that /usr should be sharable between various machines running Linux. Because it is sharable between machines, any information that is machine-local must be stored elsewhere, hence /var enters the picture.

Any large package (such as TeX and GNU Emacs) should not use a subdirectory of /usr. Instead, there should be a subdirectory inside /usr/lib, or /usr/local/lib (if it was installed totally locally).

```
/usr : Second major mount point (permanent)
     |
     |- X386        : X Window system on x86 platforms
     |- bin         : Most user commands
     |- dict        : Word lists
     |- doc         : Miscellaneous documentation
     |- etc         : Site-wide system configuration
     |- g++-include : GNU C++ include files
     |- games       : Games and educational binaries
     |- include     : Header files included by C programs
     |- info        : GNU Info system's primary directory
     |- lib         : Libraries
     |- local       : Local directory (empty after main installation)
     |- man         : Online manuals
     |- sbin        : Non-vital system administration binaries
     |- share       : Architecture-independent data
     +- src         : Source code
```

The following list of directory symbolic links need to be added. This only needs to be done until compatibility with the /var scheme can be assumed to exist.

```
/usr/adm -> /var/adm
/usr/preserve -> /var/preserve
/usr/spool -> /var/spool
/usr/tmp -> /var/tmp
/var/spool/locks -> /var/lock
```

The above symlinks should become unnecessary as packages that use the old locations are changed to support the new ones in /var.

The GNU Emacs lock file directory, if Emacs is installed, should be a symlink pointing to /var/lock/emacs if you want to be able to mount /usr read-only. It is usually found in /usr/emacs, /usr/lib/emacs, or /usr/local/lib/emacs (preferably not the first). Further information on this is located in section 5.3, "/var/lock".

4.1. /usr/X386 : X Window system on x86 platforms

This hierarchy is reserved for the use of XFree86 X11 releases.

```
/usr/X386 : XFree86 installation directory
          |
          |- bin
          |- doc
          |- include
          |- lib
          +- man
```

To simplify matters and make XFree86 more compatible with other X11 packages, our recommendation is to place the following symbolic links on your system:

```
    /usr/bin/X11 -> /usr/X386/bin
    /usr/lib/X11 -> /usr/X386/lib/X11
    /usr/include/X11 -> /usr/X386/include/X11
```

4.2. /usr/bin : Most user commands

```
/usr/bin : Binaries that are not needed in single-user mode.
         |
         |- mh           : commands for the MH mail handling system
         +- X11          : symlink to /usr/X386/bin
```

This is the primary directory of executable commands on the system.

4.3. /usr/dict : Word lists

REQUIRED files for /usr/dict:

{ words }

Traditionally this directory contains only the English ‘words’ file, which is used by look(1) and various spelling programs. ‘words’ may use either American or British spelling. Sites that require both may link ‘words’ to /usr/dict/american-english or /usr/dict/british-english.

Word lists for other languages may be added using the English name for that language, e.g., /usr/dict/french, /usr/dict/danish, etc. These should, if possible, use an ISO 8859 character set which is appropriate for the language in question; if possible the Latin1 (ISO 8859-1) character set should be used (this is often not possible).

The rationale behind having only word lists here is that they are the only files used by all spell checkers. For example, ispell(1) uses a complicated format for its "hashed dictionaries" that is only useful to ispell and should thus go in /usr/lib/ispell.

4.4. /usr/etc : site-wide system configuration

Storing configuration in /usr/etc for the software found in /usr/bin and /usr/sbin is a problem. It makes the read-only mounting of /usr through CD-ROM or NFS delivery very difficult at best.

One possible solution that we considered was to completely eliminate /usr/etc and specify that all configuration be stored in /etc. A problem with this approach is that it does not properly anticipate the possibility that many sites may want to have some configuration files that are not machine-local.

We eventually decided that /etc should be the only directory that is actually referenced by programs (that is, everything should look for configuration in /etc and not in /usr/etc). Any configuration files that need to be site-wide and are not needed before /usr is mounted (or in an emergency situation) should then be placed in /usr/etc. Then, specific files (in /etc) on specific machines may or may not be symbolically linked to appropriate configuration files located in /usr/etc. This also means that /usr/etc is technically an optional directory in the strictest sense, but we still expect all Linux systems to incorporate it.

It is not recommended for /usr/etc to contain symbolic links that point to files in /etc. This is unnecessary and would interfere with local control over machines that share a /usr directory.

4.5. /usr/include: Directory for standard include files.

This is where all of the system's general-use include files should be placed.

```
/usr/include: include files.
            |
            |- X11       : Symlink to /usr/X386/include/X11
            |- arpa      : ARPAnet defined protocol definitions
            |- asm       : Symlink to /usr/src/linux/include/asm
            |- bsd       : BSD compatibility include files
            |- gnu       : GNU include files
            |- linux     : Symlink to /usr/src/linux/include/linux
            |- net       : Generic network-related definitions
            |- netax25   : +AX25 (ARRL AX.25) specific definition
            |- netinet   : TCP/IP specific networking stuff
            |- netipx    : +IPX (Novell IPX/SPX) specific definitions
            |- protocols : Protocol definitions (mostly INET-based)
            |- readline  : The GNU readline library
            |- rpc       : Sun Microsystems RPC definitions
            |- rpcsvc    : Sun Microsystems RPC service definitions
            +- sys       : System generation include files
```

Note that not all of these subdirectories are required.

The "arpa" subdirectory contains "ARPAnet" defined protocol definitions, TCP/IP conversion functions, definitions for "ftp", "telnet" prototypes, and more.

The "net" subdirectory contains generic network-related definitions. It usually defines the system kernel interface, protocol family details, etc.

The "netinet" subdirectory contains TCP/IP specific networking stuff INET (DARPA Internet, also known as TCP/IP) specific definitions.

"ARRL AX.25" is better known as packet radio and "Novell IPX/SPX" is better known as Novell NetWare.

4.6. /usr/lib : Libraries for programming and packages

```
/usr/lib : Libraries for programming and packages
         |
         |- X11         : Symbolic link to /usr/X386/lib/X11
         |- emacs       : Support files for the GNU Emacs editor
         |- groff       : Libraries/directories for GNU groff
         |- gcc-lib     : System specific files/directories for `gcc'
         |- mh          : Libraries for the MH mail handling system
         |- mf          : Meta-Font data
         |- news        : Cnews/INN
         |- smail       : Smail
         |- terminfo    : Directories for terminfo database
         |- tex         : TeX (and LaTeX) data libraries
         |- uucp        : Commands for uucp
         +- zoneinfo    : Timezone information and configuration
```

/usr/lib traditionally includes object libraries, compiler program binaries, and other static data. It has also included some programs such as ‘sendmail’ and ‘makewhatis’ in addition to the C compiler internal binaries (which are located under /usr/lib/gcc-lib).

Since ‘makewhatis’ is not referenced by other programs, it has been relocated into /usr/sbin without other considerations. The ‘catman’ binary, also located in /usr/sbin, replaces ‘makewhatis’ on many Linux systems.

The ‘sendmail’ binary, however, is referenced by many programs. /usr/lib/sendmail should be replaced by a symbolic link which points to either the ‘smail’ binary (placed in /usr/bin) or the ‘sendmail’ binary proper (placed in /usr/sbin). The symbolic link is included_only for backwards compatibility. For increased compatibility between the systems that use ‘smail’ and those that use ‘sendmail’, a symlink should be placed at /usr/sbin/sendmail to /usr/bin/smail if a system is using ‘smail’. /usr/sbin/sendmail should become the semi-official location of sendmail-compatible mail user agents.

This arrangement also conforms to the new standard ‘sendmail’ location as defined in sendmail 8.6.x and 4.4BSD. Note that this placement demands that /usr/sbin and /usr/sbin/sendmail must be readable and executable by normal users.

‘smail’ is stored in /usr/bin because both normal users and administrators will often want to use it from the command line.

Any program or package which requires special data (non-variable) should store it in /usr/lib, or in /usr/local/lib (if it is installed locally). It is recommended that a subdirectory be used in /usr/lib for this purpose.

Note: No host-specific data for the X Window system should be stored in /usr/lib/X11 (which is really /usr/X386/lib). Host-specific configuration should be stored in /etc/X11 and host-specific variable data should be stored in /var/X11.

4.7. /usr/local : Local directory

```
/usr/local : Local directory
           |
           |- bin       : Local only binaries
           |- doc       : Local Documentation
           |- etc       : Configuration for local only binaries
           |- games     : Locally installed games
           |- lib       : Libraries for /usr/local
           |- info      : Local info pages
           |- man       : Man page hierarchy for /usr/local
           |- sbin      : Local only system administration
           +- src       : Local source code
```

The /usr/local directory is used by the system administrator to install software that needs to be safe from being overwritten when the system software is updated. It is used to store anything that is sharable among a group of machines, but not found in /usr.

This directory should always be empty after first installing Linux. No exceptions to this rule should be made other than the listed directory stubs.

Locally installed software should be placed within /usr/local rather than /usr unless it is being installed to replace or upgrade software in /usr or it is felt that the installed software is "important enough" to place in /usr or in /. Note that software placed in / or /usr will often be overwritten with complete system upgrades (unless proper backups are made). For this reason, local software should not be placed outside of /usr/local without good reason.

4.8. /usr/man : Manual pages

```
/usr/man : Manual page hierarchy
         |
         |- man1          : User programs
         |- man2          : System calls
         |- man3          : Library functions and subroutines
         |- man4          : Devices
         |- man5          : File formats
         |- man6          : Games
         |- man7          : Miscellaneous
         |- man8          : System Administration
         +- man9          : Kernel internal variables and functions
```

The cat page sections (cat[1-9]) containing formatted manual page entries are also found within subdirectories of /usr/man, but are not required nor should they be distributed in lieu of nroff source manual pages.

Local and X Windows manual pages (if present) should be stored in /usr/local/man and /usr/X386/man, respectively. These directories have a similar structure to /usr/man (man[1-9], cat[1-9], empty subdirectories being omitted).

The MH mail handling system manual pages should have "mh" affixed to all manual page filenames. X11 manual pages usually have "x" affixed to all X11 manual pages.

As Linux is more extensively used in countries where English is not the primary language, translations of manual pages are often developed. There is the impending problem that these manual pages will have to be stored somewhere else. Some German Linux distributions have already created a manual page system that is placed in /usr/man with the suffix "g". This is likely cause problems in the long run as other languages appear, particularly other languages which begin with the same letter, such as Greek. There is also the additional problem that manual pages will be formatted in many different character code formats. Even within the ISO 8859 character code standards, each separate implementation is incompatible with each other implementation.

Therefore, all non-English manual pages sections should be stored in subdirectories within /usr/man named according to the language that the the contained manual pages are written in (lowercase characters). Thus, for German manual pages:

/usr/man/german/man[1-9] and possibly /usr/man/german/cat[1-9]

Then, German-speaking Linux users can add /usr/man/german to their manual search path before /usr/man so that German manual pages are referenced first.

The best scheme, often found on HP-UX systems, is a manual pager that is based on NLS (native language support) or an NLS-like system which obeys a $LANG environment variable. This support is currently not available for Linux systems, meaning that this section is slightly presumptive. Under this scheme, if a German manual page is not located for a given command then the English version may be referenced automatically. This structure is designed with the intention that it may make a smooth transition from being supported through the $MANPATH environment variable to support through a $LANG environment variable, making the $MANPATH language-independent. Such software should also possess the ability to handle various character code standards, but conjecture on how to implement this handling is not feasible at this time.

This setup will be needed as the number of foreign (non-English) manual pages increases. German is the language mentioned here since it is the only non-English manual page system distributed with any Linux system at this time. Other languages will probably follow and they should try to conform to this scheme as well (if it proves to be adequate).

The practice of placing non-English manual pages in subdirectories of /usr/man should be followed as well for other manual page hierarchies, such as /usr/local/man and /usr/X11/man. (This portion of the FSSTND also applies later in section 5.3 on the optional /var/catman structure.)

Note: Using the language's own name for itself ("deutsch") rather than the English ("german") was considered, but this was met with disapproval from many people, including those who do not speak English as a first language. The reasons include: simplicity, the difficulty in displaying many languages' names in ASCII characters, and the fact that everyone should be able to recognize their language name in English.

A description of each section follows:

- man1: User programs
  Manual pages that describe publicly accessible commands are contained in this chapter. Most program documentation that a user will need to use is located here.
- man2: System calls
  This section describes all of the system calls which are entries to the Linux kernel (operating system). This section can be very useful to programmers, but users have little need of the items in section 2.
- man3: Library functions and subroutines
  Section 3 describes user-level library routines. This is another chapter that is only really of interest to programmers.

- man4: Special files
  Section 4 describes the special files, related driver functions, and networking support available in the system. Typically, the device files found in /dev.
- man5: File formats
  The formats for many nonintuitive data files are documented in the section 5. This includes various include files, program output files, and system files.
- man6: Games
  This chapter documents games, demos, and generally trivial programs. Different people have various notions about how essential this is.
- man7: Miscellaneous
  Manual pages that are difficult to classify are designated as being section 7. The troff and other text processing macro packages are found here.
- man8: System administration
  Documentation for programs used by system administrators for system operation and maintenance are documented here. Some of these programs are also occasionally useful for normal users.
- man9: Kernel internal variables and functions
  This is used on Linux systems to document the kernel source code.

4.9. /usr/sbin : Non-essential standard system binaries

This directory contains any non-essential binaries used exclusively by the system administrator. System administration programs that are required for system repair, system recovery, mounting /usr, or other essential functions should be placed in /sbin instead.

Typically, /usr/sbin contains networking daemons, any non-essential administration tools, and binaries for non-critical server programs. These server programs are used when entering the System V states known as "run level 2" (multi-user state) and "run level 3" (networked state) or the BSD state known as "multi-user mode". At this point the system is making services available to users (e.g., printer support) and to other machines (e.g., NFS exports).

Local system administration programs should be placed in /usr/local/sbin.

4.10. /usr/share : Architecture-independent data

Any specifications for /usr/share will be included in a supplementary drafts to the main FSSTND standard. Note that it is the consensus opinion of FSSTND that /usr/share is not needed on the majority of Linux systems. At this time, confining ourselves by providing an extensive definition of this directory would be a bad idea.

Please refer to section 6.2 for more detailed description of /usr/share.

4.11. /usr/src : Source code

```
/usr/src : Source code
          |
          +- linux : Source code for Linus’ kernel
```

Any non-local source code should be placed in this subdirectory. The only source code that should always be placed in a specific location is the kernel source (when present or linked in part to the /usr/include structure). Subdirectories may be used here if desired.

The source code for the kernel should always be in place or at least the include files from the kernel source. Those files are located in these directories:

```
/usr/src/linux/include/asm
/usr/src/linux/include/linux
```

/usr/include usually contains links to 'asm' and 'linux' in the source directory. Since they are needed by the C compiler, at least those include files should always be distributed with installations. They should also be distributed in the /usr/src/linux directory so there are no problems when system administrators upgrade their kernel version for the first time.

5. The /var directory

```
/var : Directories of files that tend to grow or vary in size
     |
     |- X11       : Variable files/directories for the X Window System
     |- adm       : System logging and accounting files
     |- catman    : Locally-formatted manual pages
     |- lock      : Lock files
     |- named     : DNS files (for 'named'), networking only
     |- preserve  : Saved text edited by 'vi' after crash or hang-up
     |- spool     : Directories for queue work to be performed later
     |- tmp       : Temporary files, used to keep /tmp small.
     +- yp        : Network Information Service (NIS) database files
```

The directory for variable data files. This includes spools, administrative files, logging files, transient files, and temporary files.

A good reason to use /var is to make it possible to mount /usr read-only. Everything that once went into /usr that is written to on a temporary basis, now goes into /var. The aforementioned symbolic links, also mentioned below in the issues and rationale section, should be added to /usr for compatibility. This is very helpful if you are mounting /usr through NFS, if you want a read-only /usr for safety reasons, or your /usr is mounted from read-only media such as a CD-ROM.

5.1. /var/X11 : Variable data for the X Window System

```
/var/X11 : Host-specific data relevant to X
         |
         +- xdm         : X display manager files
```

The standard program used to provide a per-machine login environment is XDM. XDM binaries should still be placed in the previous location in /usr/X386/lib/X11/xdm (e.g., 'chooser').

5.2. /var/adm : System logging and accounting files

All system logging should be written to this subdirectory and not to /var/log. 'wtmp' and 'lastlog' should be stored here.

/var/adm is also the appropriate location for all distribution packaging support. Until there is a common Linux packaging format, package data should be stored in "distribution specific" subdirectories of this directory.

5.3. /var/catman : Locally-formatted manual pages (optional)

This directory provides a standard location for sites that provide a read-only /usr partition, but wish to allow caching of locally-formatted man pages. Sites that mount /usr as writeable (e.g., single-user installations) may choose not to use /var/catman and write formatted man pages into the cat[1-9] directories in /usr directly. Most sites should use one of the following options instead:

- Preformat all manual pages in /usr with the 'catman' program.
- Allow no caching of formatted man pages, and require nroff to be run each time a man page is brought up.

- Allow local caching of formatted man pages in /var/catman.

The structure of /var/catman needs to reflect both the fact of multiple man page hierarchies and the possibility of multiple language support. Given a formatted man page that would normally appear in /usr/<path1>, the cached formatted version should go in /var/catman/<path2>, where <path2> is <path1> without the 'man' component. Thus /usr/man/cat1/ls.1 ends up in /var/catman/cat1/ls.1, and /usr/X386/man/german/cat3/foo.3 in /var/catman/X386/german/cat3/foo.3.

Note: current versions of the 'man' program do <u>not</u> support this proposed standard. Until such support is available, sites will have to choose between preformatting cat pages in /usr (or mounting /usr as writeable), and symlinking the various cat[1-9] directories to the appropriate subdirectory of /var/catman.

Local policy will dictate whether man pages written to /var/catman are eventually transferred to /usr, and whether formatted man pages in /usr are expired if they are not accessed for a period of time.

5.4. /var/lock : Lock files

Lock files should be stored within the /var/lock directory structure.

```
/var/lock : Lock files
          |
          +- emacs      : Emacs lock files
```

To preserve the ability to mount /usr read-only, no lock files or lock file systems should be stored on the /usr partition.

Device lock files, such as the serial device lock files which are found in /usr/spool/locks or /usr/spool/uucp, should be stored in /var/lock. The naming convention which should be followed is 'LCK..{device}'. For instance, if you are using a program which uses /dev/cua0, then it should create a lock file, '/var/lock/LCK..cua0'. The format used for Linux device lock files should be the HDB style. The HDB style is to write the locking process ID in ASCII, padded with leading zeroes to ten characters, followed by a newline.

Then, anything wishing to use /dev/cua0 can observe the lock file and act accordingly.

Programs with large lock file systems should use a subdirectory of /var/lock. For instance, GNU Emacs normally stores Emacs lock files in /usr/lib/emacs/lock — this poses a serious problem if you are trying to mount /usr read-only. Emacs lock files should instead be placed in /var/lock/emacs.

If you are a developer/maintainer of an application which uses lock files in any manner, it is a good idea to contact the FSSTND mailing list or me to discuss the arrangement of your lock files.

At this time, lock files which are stored in /etc, such as pid lock files, should remain in /etc.

5.5. /var/named : DNS and 'named' files

This is only needed for systems running a DNS name server (networking protocol for name servers).

Note: The directory placement of DNS may possibly change due to the existence of DNS servers which are not based on named.

5.6. /var/spool : Spool directories

/var/spool is traditionally used for machine-local data being spooled to or from UNIX subsystems. For example, print jobs are spooled here for delivery to the lpd system, out-bound mail is spooled for delivery to remote systems, and uucp files are spooled for writing to UUCP neighbors. In-bound mail and news are spooled here for delivery to users, and at and cron jobs are spooled for delayed execution by the cron daemon.

```
/var/spool : Spool directories
           |
           |- at        : at jobs
           |- cron      : Cron jobs
           |- lpd       : Printer spool directory
           |- mail      : Directory for user mailboxes
           |- mqueue    : Outgoing mail queue
           |- news      : News spool directory
           +- uucp      : Spool directory for uucp
```

UUCP lock files should be placed in /var/lock. See the above section on /var/lock.

5.5.1. /var/spool/lpd

```
/var/spool/lpd : Printer spool directory
               |
               |- {printer}      : Spools for a specific printer
               +- {printer}/lock : Lock file for a specific printer
```

The lock file for lpd, ‘lpd.lock’, should be placed in /var/spool/lpd. The lock file for each printer should be placed in the spool directory for that specific printer.

5.7. /var/tmp : temporary files, used to keep /tmp small

Files in /var/tmp are stored for an unspecified duration (please remember that system temporary directories are not guaranteed to hold data for any particular duration).

Data stored in /var/tmp is typically cleaned out "in a site-specific manner", but usually at less frequent intervals than /tmp. More information on temporary directories is in the section of the standard devoted to /tmp (above).

/usr/tmp is usually symlinked to /var/tmp for compatibility reasons.

5.8. /var/yp : Network Information Service (NIS) database files

The Network Information Service (NIS) was formerly known as the Sun Yellow Pages (YP). The functionality and directory placement of the two is the same, but the name "Yellow Pages" is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

6. Issues and Additional Rationale

This section discusses several areas that may require further explanation.

6.1. What is Essential?

The answer is: essential to clean, create, prepare, check, find and mount other filesystems (possibly on remote machines). There are other definitions, but this is a general definition that most people will at least incorporate into their own.

6.2. Networking

Networking presented an interesting dilemma. Some people wanted to separate networking binaries and configuration from other binaries and configuration. However, we disagree. We feel that networking is not a "package", but an integral part of most UNIX (and UNIX-like) machines. Because of this networking should not be placed into a single directory, but systematically placed in the appropriate directories.

- `/bin`: anything a user will want to use that is also considered vital
  `{hostname, netstat, ping, ...}`
- `/sbin`: anything only root needs and is considered vital
  `{arp, ifconfig, ifsetup, route}`
- `/usr/bin`: any binaries a user will want to use, but are not vital
  `{finger, rcp, rlogin, telnet, ...}`
- `/usr/sbin`: any root only networking binaries that are not vital
  `{ftpd, inetd, lpd, telnetd, ...}`

While this may seem confusing at first (and it does take a moment to digest), it does make sense. If you can only mount root for some reason and you need access to networking to repair your system, you don't need the files to be off in /usr/etc (as they often are). Files that are needed to mount /usr in normal (and emergency situations) are placed on the root subtree and any others are placed in /usr in order to keep the size of root small.

Configuration files for networking belong in /etc.

6.3. Architecture-independent Structures

The structure, /usr/share, typically contains architecture-independent files such as man-pages, timezone, terminfo information, et al. As of this time, there are no different architectures for Linux, but with the passage of time we should see Linux include other architectures and other UNIX-like systems.

One note: no program should ever reference anything in /usr/share. For instance, a manual page program should never directly look in /usr/share/man/man1/ls.1, but it should refer to /usr/man/man1/ls.1 at all times. Anything in /usr/share will be "pointed to" by the use of symlinks from other areas in the filesystem, such as /usr/man, /usr/lib/{something}, etc.

The specifications for /usr/share are still be worked out.

6.4. Symbolic Links

There are a wide range of uses for symbolic links (symlinks) in every filesystem. While symlinks are not encouraged for default setup (found after installing Linux) in a standard such as this, they are often used with good purpose on different systems. The point is that symlinks should be there to keep everything where everyone else expects find it.

Be prepared to accept that certain directories, even those contained on the root directory, are still going to be symlinks. For instance, on some systems /home will not be on the root, but symlinked to a /var directory, or to somewhere else. /home could also have its own physical partition, of course, and be mounted on its own.

Similarly, because /usr might be on a central file server mounted via NFS, /usr/local could be symlinked to /var/local. Like /usr/lib/emacs/lock, this change can be justified by recalling the main purpose for having /var: to separate directories of files that vary on different systems and machines from those that may be shared.

Sometimes systems will also link /tmp to /var/{something} if the root partition becomes too small (or starts out too small). There are more examples of "good" uses of symbolic links, but the entire issue boils down to two things: packages should be able to find things where they expect them (within reason) and symbolic links can be used to solve the problem in many cases. However, problems also can arise from using too many symbolic links. These problems include over-reliance on symbolic links to solve problems, confusion resulting from overuse of symbolic links, and the aesthetic preferences of different people.

6.5. Statically linked binaries

Linux is currently running on a wide variety of systems, some single user with small disks, some as servers in large networked environments. Because of this variety, this standard sets no rule regarding what binaries are static or dynamic with the following two exceptions. Both ‘ln’ and ‘sync’ should exist in /bin; any statically linked versions may be placed in /sbin, or replace those in /bin.

Large Linux systems may wish to include other statically linked binaries (sh, init, mkfs, fsck, tunefs, mount, umount, swapon, swapoff, getty, login, etc.). The developers and/or system administrators are free to statically/dynamically link these and other binaries as they see fit, as long as the location of the binaries doesn’t change.

Networked systems (especially those of the future which may lack floppy drives), may want to make ifconfig, route, hostname, and other networking utilities static as well. This is usually not needed.

The FSSTND mailing list

The FSSTND mailing list is located at linux-fsstnd@ucsd.edu. This list was originally located on the linux-activists@Niksula.hut.fi "Mail-Net" as the FSSTND channel. (To subscribe to the list send mail to listserv@ucsd.edu with the body of "ADD linux-fsstnd".)

Thanks to Network Operations at the University of California at San Diego who allowed us to use their excellent mailing list server.

Acknowledgments

Credit for this text should be given to the FSSTND activists, developers, system administrators, and users whose input was essential to this standard. I also wish to thank each of the contributors who helped me to write, compile, and compose this, a consensus standard.

I also wish to give real credit to those Linux developers who have seen that giving Linux a common filesystem layout is something that will further the development of the Linux operating system. I also wish to note the bravery and perseverance of those Linux developers who started following this standard before it was completed.

Original contributors

Drew Eckhardt	<drew@kinglear.cs.colorado.edu>
Ian Jackson	<ijackson@nyx.cs.du.edu>
Ian McCloghrie	<ian@ucsd.edu>
Daniel Quinlan	<quinlan@bucknell.edu>
Mike Sangrey	<mike@sojurn.lns.pa.us>
David H. Silber	<dhs@glowworm.firefly.com>
Theodore Ts'o	<tytso@athena.mit.edu>
Stephen Tweedie	<sct@dcs.ed.ac.uk>

Additional contributors

Brandon S. Allbery	<bsa@kf8nh.wariat.org>
Rik Faith	<faith@cs.unc.edu>
Stephen Harris	<hsw1@papa.attmail.co>
Fred N. van Kempen	<waltje@uwalt.nl.mugnet.org>
John A. Martin	<jmartin@csc.com>
Chris Metcalf	<metcalf@lcs.mit.edu>
Ian Murdock	<imurdock@shell.portal.com>
David C. Niemi	<niemidc@oasis.gtegsc.com>