Linux Filesystem Structure — Release *1.2*

Daniel Quinlan <Daniel.Quinlan@linux.org>

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. Yet, there exists a need for standardization of the Linux filesystem structure. This document aims to specify standard locations of files and directories in Linux systems. A standardized filesystem structure allows users, developers, and distributors to obtain system components from various sources that will work together as smoothly as if they had been developed under a centralized development process. It also eases system administration, development of second and third party packages, and the writing of implementation independent documentation.

March 28, 1995

Linux is not a trademark, and has no connection to UNIX.

UNIX is a trademark of the X/Open Company, Ltd.

HP-UX is a trademark of Hewlett-Packard.

Novell and Novell NetWare are trademarks of Novell.

SunOS, Sun Microsystems, Sun NIS, Sun RPC, and NFS are trademarks of Sun Microsystems, Inc.

System V and SVR4 are trademarks of AT&T.

X Window System is a trademark of X Consortium, Inc.

All other copyrights are owned by their owners, unless specifically noted otherwise. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark.

Copyright © 1994, 1995 Daniel Quinlan

Permission is granted to copy and distribute verbatim copies of this standard provided the copyright and this permission notice are preserved on all copies.

Permission is granted for FSSTND contributors and participants to copy and distribute modified versions of this standard under the conditions for verbatim copying for purposes of filesystem standardization activities only, and subject to those restrictions listed below.

The following restrictions apply to reproducing or transmitting the document in any form:

- All copies or portions thereof must identify the document's title and section, and must be accompanied by this entire notice in a prominent location.
- No portion of this document may be redistributed in any modified or abridged form without the prior approval of the FSSTND coordinator.

Any entities seeking permission to distribute any material derived from this document (other than verbatim copies) must contact the FSSTND coordinator for the appropriate license.

Preface

Status of the Standard

This is version 1.2 of the Linux Filesystem Structure (FSSTND).

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 Hierarchy. (Section 4)

— The /var Hierarchy. (Section 5)

— Issues and Additional Rationale. (Section 6)

Typographical Conventions

Courier font is used for the names of files and directories.

Components of filenames that vary are represented by a description of the contents enclosed in "<" and ">" characters, <thus>. Electronic mail addresses are also enclosed in "<" and ">" but are shown in the usual typeface.

Optional components of filenames are enclosed in "[" and "]" characters and may be combined with the "<" and ">" convention. For example, if a file existed which could be either be found either with or without an extension, it might be represented by <filename>[.<extension>].

Variable substrings of directory names and filenames are indicated by "*".

1. General

1.1 Scope

This document specifies a standard filesystem structure for Linux systems, including the location of files and directories, 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 primarily intended to be a reference and is 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 incompatible with one another.
- The most widely used filesystem hierarchies were not well structured and differed gratuitously from more modern directory structure "standards" (such as System V, BSD, SunOS, 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.)

Differences in opinion arise in any standardization effort. The need for consensus and common practice within the Linux community should overshadow these differences.

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 we hope the Linux community will 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 the problems we address will be discovered or that our solutions will no longer be the best possible solutions. For these reasons, the FSSTND group plans to release supplementary drafts in addition to periodic updates to this document.

Comments related to this standard are welcomed by the FSSTND group. Any comments or suggestions for changes should be directed to the FSSTND coordinator, or if you prefer, any of the listed contributors. Typographical or grammatical comments should be directed to the FSSTND coordinator.

There is also a FAQ, maintained by Ian McCloghrie, which answers some of the more commonly asked questions about this standard. If you want to implement the FSSTND or if you have some questions, please read the FSSTND FAQ first. This is available via anonymous FTP at tsx-11.mit.edu in `/pub/linux/docs/linux-standards/fsstnd/FSSTND-FAQ`.

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. For 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 <Daniel.Quinlan@linux.org>

1.2 Specific Problems

Naturally, while standardizing the Linux filesystem structure, there were some specific problems that we sought to correct. Here are some of the most obvious and major 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 between these two directories varies greatly between the Linux distributions.
- Including both binaries and configuration files in `/etc` makes this directory more confusing and more difficult to maintain, both for inexperienced users and for system administrators (especially those with large systems).
- The division between what is a site-wide configuration file and what is a machine-local configuration file is difficult to establish.
- Many common implementations of `/usr` cannot be mounted read-only because they contain variable files and directories that need to be written to.
- In a networked environment it is desirable to serve software to workstations via NFS. Such filesystems may 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.
- Common Linux filesystem structures were generally not well suited to networked installations which may require read-only components within the filesystem (primarily in the `/usr` hierarchy) or involve 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 problems above, several objectives were identified that needed to be achieved 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 problems listed above while limiting transitional difficulties while moving away from the former de-facto standards.
- Gain approval of distributors, developers, and other important people in the Linux community, as well as encouraging them to give us their suggestions.
- Provide a standard that the whole Linux community will choose to follow because it solves the problems above and provides the most sensible structure for the filesystems of Linux installations.

Some of these objectives have already been fully or partially met due to the limited distribution of an advance draft 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 the necessary restructuring of the Linux filesystem would be to create a mailing list for the purpose of developing a consensus standard.

After a comprehensive discussion, with surprisingly few flames, a preliminary draft was written. With the help of several dedicated people, the draft was finished and the 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 agreed that this standardization effort was worthwhile and supported it.

These are some of the developers who aim to follow the FSSTND standard, partially or completely, listed in alphabetical order:

- ATIM Linux/PRO
  Fred N. van Kempen *et al.* <waltje@infomagic.com>
- BOGUS Linux
  Rik Faith, Kevin E. Martin, and Doug L. Hoffman <linux-bogus@cs.unc.edu>
- Debian Linux
  Ian A. Murdock <imurdock@debian.org>
- LILO boot loader
  Werner Almesberger <almesber@nessie.cs.id.ethz.ch>
- MCC Interim Linux
  Owen Le Blanc <LeBlanc@mcc.ac.uk>
- Red Hat Software Linux (RHS Linux)
  Marc Ewing <marc@redhat.com>
- Slackware Linux
  Patrick J. Volkerding <volkerdi@mhd1.moorhead.msus.edu>
- TAMU Linux
  Dave Safford <dave.safford@net.tamu.edu>

- util-linux package
  Rik Faith <faith@cs.unc.edu>
- Yggdrasil Plug-and-Play Linux
  Adam J. Richter <adam@yggdrasil.com>

1.5 Conformance with this Document

This section defines the meanings of the terms "compliant" and "compatible" with respect to this standard, and of "partial" compliance and conformance.

An "implementation" here refers to a distribution, an installed system, a program, a package (or some similar piece of software or data), or some component thereof.

An implementation is fully compliant with this standard if every requirement in this standard is met. Every file or directory which is part of the implementation must be located as specified in this document. If the contents of a file are described here the actual contents must correspond to the description. The implementation must also attempt to find any files or directories (external to itself) primarily or exclusively in the location specified in this standard.

An implementation is fully compatible with this standard if every file or directory which it contains can be found by looking in the location specified here and will be found with the contents as specified here, even if that is not the primary or physical location of the file or directory in question. The implementation must, when it attempts to find any files or directories which are not part of it, do so in the location specified in this standard, though it may also attempt to find it in other (non-standard) locations.

An implementation is partially compliant or compatible if it complies with or is compatible with a significant subset of this document. Partial compliance or compatibility is only intended to apply to distributions and not to 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.

In order to qualify as partially FSSTND compliant or partially FSSTND compatible an implementation must provide a list of all places at which it and the FSSTND document differ in addition to a brief explanation of the reasoning for this difference. This list shall be provided with the implementation in question, and also made available to the FSSTND mailing list or the FSSTND coordinator.

The terms "must", "should", "contains", "is" and so forth should be read as requirements for compliance or compatibility.

Note that an implementation does not need to contain all the files and directories specified in this standard to be compliant or compatible. It is merely necessary for those files that it does contain to be located appropriately. For example, if the ext2 filesystem is not supported by a distribution, the ext2 tools need not be included, even though they are mentioned explicitly in the section on `/sbin`.

Furthermore, certain portions of this document are optional. In this case this will be stated explicitly, or indicated with the use of one or more of "may", "recommend", or "suggest". Items marked as optional have no bearing on the compliance or conformance of an implementation; they are suggestions meant to encourage common practice, but may be located anywhere at the implementor’s choice.

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 agree in every possible respect with 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 and sound practices in the Linux community
- The implementation of other filesystem structures
- Applicable standards

It is possible to define two orthogonal categorizations of files: shareable vs. unshareable and variable vs. static.

Shareable data is that which can be shared between several different machines; unshareable is that which must be local to a particular machine. For example, user home directories are shareable data, but device lock files are not.

Static data includes binaries, libraries, documentation, and anything that does not change without system administrator intervention; variable data is anything else that does change without system administrator intervention.

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 shareable and unshareable 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 data that can be shared between different machines to save space and ease the task of maintenance.
- In a networked environment, certain files contain information specific to a single machine. Therefore these filesystems cannot be shared (without taking special measures).
- The de-facto implementation of the filesystem did not allow the `/usr` hierarchy to be mounted read-only because it contained files and directories that need to be written to often. This is a factor that must be addressed when parts of `/usr` are shared on a network or mounted read-only because of other considerations such as security.

The "shareable" distinction can be used to support, for example:

- A `/usr` partition (or components of `/usr`) mounted (read-only) through the network (using NFS).
- A `/usr` partition (or components of `/usr`) mounted from read-only media. A CD-ROM can be considered a read-only filesystem shared with other Linux systems, using the mail system as a network.

The "static" versus "variable" distinction 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 may 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.

Summarizing chart with examples:

	shareable	unshareable
static	/usr /home	/etc /boot
variable	/var/spool/mail /var/spool/news	/var/run /var/lock

3. The Root Directory

This section describes the root directory structure. The contents of the root filesystem should be adequate to boot, restore, recover, and/or repair the system:

- To boot a system, enough must be present to mount `/usr` and other non-essential parts of the filesystem. This includes utilities, configuration, boot loader information, and other essential start-up data.
- To enable recovery and/or repair of a system, those utilities needed by an experienced maintainer to diagnose and reconstruct a damaged system should be present on the root filesystem.
- To restore a system, those utilities needed to restore from system backups (on floppy, tape, etc.) should be present on the root filesystem.

The primary concern used to balance these considerations, which favor placing many things on the root filesystem, is the goal of keeping root as small as reasonably possible. For several reasons, it is desirable to keep the root filesystem small:

- It is often mounted from very small media. For example, many Linux users install and recover systems by mounting root off a RAM disk, which is copied from a single 1.44M or 1.2M floppy disk.
- The root filesystem has many system-specific configuration files in it. Possible examples include a kernel that is specific to the system, a different hostname, etc. This means that the root filesystem isn't always shareable between networked systems. Keeping it small on networked systems minimizes the amount of space lost on servers to unshareable files. It also allows workstations with smaller local hard drives.
- While you may have the root filesystem on a large 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 root filesystems on smaller partitions. If you are a developer then you may be turning your assumption into a problem for a large number of users.
- Disk errors that corrupt data on the root filesystem are a greater problem than errors on any other partition. A small root filesystem is less prone to corruption as the result of a system crash.

This document as currently drafted requires a writable root filesystem (primarily due to `/etc/mtab`). 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 — for example, it might be mounted from an NFS server.

Software should never create or require special files or subdirectories in the root directory. The Linux filesystem structure provides more than enough flexibility for any package. Any package that does occupy a directory under the root of the filesystem 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
├── 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 hierarchy
└── var       Variable data
```

Each directory listed will be discussed in detail in a separate subsection below. `/usr` and `/var` each have their own major sections of this document.

The Linux kernel image should be located in either `/` or `/boot`. If it is located in `/`, we recommend using the name `vmlinux` or `vmlinuz` which have been used in recent Linux kernel source packages. Additional information on kernel placement can be found in the section regarding `/boot`, below.

3.1 /bin : Essential user command binaries (for use by all users)

`/bin` contains commands that may be used by both the system administrator and by users, but which are required in single user mode. It may also contain commands which are used indirectly by scripts.

All root-only binaries such as daemons, `init`, `getty`, `update`, etc. should be placed in `/sbin` or `/usr/sbin`, depending on whether they are essential. For further discussion of the definition of what is essential on the root filesystem, please read section 6, "Issues and Additional Rationale".

There should be no subdirectories within `/bin`.

Command binaries that are not essential enough to place into `/bin` should be placed in `/usr/bin`, instead. Items that are only used by non-root users (`mail`, `chsh`, etc.) are not essential enough to be placed into the root partition.

Required files for /bin:

- General commands:

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

  ```
  { arch, cat, chgrp, chmod, chown, cp, date, dd, df, dmesg, echo,
    ed, false, kill, ln, login, ls, mkdir, mknod, more, mount, mv, ps,
    pwd, rm, rmdir, sed, 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`. `pdksh`, which may be the `/bin/sh` on install disks, should likewise be arranged with /bin/sh being a symlink to /bin/ksh. The use of a symbolic link in these cases allows users to easily see that /bin/sh is not a true Bourne shell.

Since the de-facto standard location of the C-shell is /bin/csh, if and only if a C-shell or equivalent (such as tcsh) is available on the system, it should be available by the name /bin/csh. /bin/csh may be a symbolic link to /bin/tcsh or /usr/bin/tcsh.

The [and test commands are built into Bash, pdksh, zsh, and recent Korn shells — essentially every Bourne shell replacement there is for Linux. These commands 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 and 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 using programs other than gzip and tar, then the root partition should contain the minimal necessary restoration components. 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 (i.e., 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.

  Restoration commands may appear in either /bin or /usr/bin on different Linux systems.

- Networking commands:

  These are 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. Programs necessary to arrange for the boot loader to be able to boot a file (such as the lilo map installer) should be placed in /sbin. Configuration files for boot loaders should be placed in /etc.

As already stated above, the Linux kernel may either be placed in / or in /boot. It is recommended that the kernel image be given a more descriptive filename if placed within /boot.

3.3 /dev : Device files

This is the device directory. It should contain an entry for every device that the Linux kernel is configured to support.

`/dev` also contains a script named `MAKEDEV` which can create devices as needed. It may also contain a `MAKEDEV.local` for any local-only devices.

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

Symbolic links in `/dev` should not be distributed with Linux systems except as provided in the Linux device list. This is because local setups will often differ from that on the distributor's development machine. Also, if a distribution install script configures the symbolic links 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 by reference the Linux Device List which is maintained by H. Peter Anvin <Peter.Anvin@linux.org>, the Linux Device Registrar. All device special files should follow the standard in that document, which is available via anonymous ftp at ftp.yggdrasil.com in `/pub/device-list`.

3.4 /etc : Machine-local system configuration

`/etc` contains configuration files and directories, which are local to the current system.

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

`/etc` — Machine-local system configuration

```
├─ X11          Configuration for the X Window System
└─ skel         User skeleton configuration
```

`/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 may contain subdirectories for different user groups (e.g., `/etc/skel/staff` or `/etc/skel/users`).

`/etc/X11` is the recommended location for all X11 machine-local configuration. This directory is necessary to allow local control if `/usr` is mounted read only. Files that should be in this directory include `Xconfig` (and/or `XF86Config`) and `Xmodmap`.

Subdirectories of `/etc/X11` may include those for `xdm` and for any other programs (some window managers, for example) that need them. We recommend that window managers with only one configuration file which is a default `.*wmrc` file should name it `system.*wmrc` (unless there is a widely-accepted alternative name) and not use a subdirectory. Any window manager subdirectories should be identically named to the actual window manager binary.

`/etc/X11/xdm` holds the configuration files for `xdm`. These are most of the files normally found in `/usr/lib/X11/xdm`; see Section 5, `/var/lib/xdm`, for more information.

The following section is intended partly to illuminate the description of the contents of `/etc` with a number of examples; it is definitely not an exhaustive list.

Required files for /etc:

- General files:

  These files are needed on most Linux systems.

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

- 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 may be used, or a mixture of the two.

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

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 section describes only a suggested placement for user home directories; nevertheless we recommend that all Linux distributions use this as the default location for home directories.

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

On large systems (especially when the `/home` directories are shared amongst many machines using NFS) it is useful to subdivide user home directories. Subdivision may 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 program should rely on this location. If you want to find out a user's home directory, you should use the `getpwent(3)` library function rather than relying on `/etc/passwd` because user information may be stored remotely using systems such as NIS.

3.6 /lib : Essential shared libraries and kernel modules

The `/lib` directory contains those shared library images needed to boot the system and run the commands in the root filesystem.

`/lib` — essential shared libraries and kernel modules

```
└─ modules      Loadable kernel modules
```

This includes `/lib/libc.so.*`, `/lib/libm.so.*`, the shared dynamic linker `/lib/ld.so`, and other shared libraries required by binaries in `/bin` and `/sbin`.

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. The library `libm.so.*` may also be placed in `/usr/lib` if it is not required by anything in `/bin` or `/sbin`.

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.)

The specification for `/lib/modules` is forthcoming.

3.7 /mnt : Mount point for temporarily mounted filesystems

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

We recommend against the use of this directory by installation programs, and suggest that a suitable temporary directory not in use by the system should be used instead.

3.8 /proc : Kernel and process information virtual filesystem

The proc filesystem is becoming the de-facto standard Linux method for handling process and system information, rather than `/dev/kmem` and other similar methods. We strongly encourage this 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 account's home directory may be determined by developer or local preference. Obvious possibilities include `/`, `/root`, and `/home/root`.

If root's home directory is not stored on the root partition it will be necessary to make certain it will default to `/` if it can not be located.

Note: we recommend against using the root account for mundane things such as mail and news, but rather to use it solely for systems administration. For this reason, we recommend that subdirectories such as `Mail` and `News` not appear in the root account's home directory. We recommend that mail for root and postmaster be forwarded to a more appropriate user.

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 binaries essential for booting the system in addition to the binaries in `/bin`. Anything executed after `/usr` is known to be mounted (when there are no problems) should be placed into `/usr/sbin`. Local-only system administration binaries should be placed into `/usr/local/sbin`.

Deciding what things go in `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` if 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.

We recommend that users have read and execute permission for everything in `/sbin` except, perhaps, certain setuid and setgid programs. The division between `/bin` and `/sbin` was not created for security reasons or to prevent users from seeing the operating system, but to provide a good partition between binaries that everyone uses and ones that are primarily used for administration tasks. There is no inherent security advantage in making `/sbin` off-limits for users.

Required files for /sbin:

- General commands:

  `{ clock, getty, init, update, mkswap, swapon, swapoff, telinit }`

- Shutdown commands:

  `{ fastboot, fasthalt, halt, reboot, shutdown }`

  (Or any combination of the above, so long as `shutdown` is included.)

- Filesystem management commands:

  `{ fdisk, fsck, fsck.*, mkfs, mkfs.* }`

  `*` = one of `ext, ext2, minix, msdos, xia` and perhaps others

- Second extended filesystem commands (optional):

  `{ badblocks, dumpe2fs, e2fsck, mke2fs, mklost+found, tune2fs }`

- Boot-loader map installer:

{ lilo }

- Networking commands:

  { arp, ifconfig, route }

Optional files for /sbin:

- Static binaries:

  Static `ln` (`sln`) and static `sync` (`ssync`) are useful when things go wrong. The primary use of `sln` (to repair incorrect symlinks in `/lib` after a poorly orchestrated upgrade) is no longer a major concern now that the `ldconfig` program (usually located in `/usr/sbin`) 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` since a site may choose to run `ldconfig` at boot time, rather than only when upgrading the shared libraries. (It's not clear whether or not it is advantageous to run `ldconfig` on each boot.) Even so, some people like `ldconfig` around for the following (all too common) situation:

  (1) I've just removed `/lib/<file>`.

  (2) I can't find out the name of the library because `ls` is dynamically linked, I'm using a shell that doesn't have `ls` built-in, and I don't know about using "`echo *`" as a replacement.

  (3) I have a static `sln`, but I don't know what to call the link.

  { ldconfig, sln, ssync }

- Miscellaneous:

  So as to cope with the fact that some keyboards come up with such a high repeat rate as to be unusable, `kbdrate` may be installed in `/sbin` on some systems.

  Since the default action in the kernel for the Ctrl-Alt-Del key combination is an instant hard reboot, it is generally advisable to disable the behavior before mounting the root filesystem in read-write mode. Some `init` suites are able to disable Ctrl-Alt-Del, but others may require the `ctrlaltdel` program, which may be installed in `/sbin` on those systems.

  { ctrlaltdel, 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 of data which is stored in `/var/tmp`. `/tmp` may be cleaned out at boot time or at relatively frequent intervals. Therefore, data stored in `/tmp` should not be expected to remain for any long period.

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

System administrators may choose to link `/tmp` to another directory, such as `/var/tmp`; this is useful, for example, to conserve space on the root partition. If this is done, then the persistence of files in `/var/tmp` should be at least as long as for `/tmp`.

`/tmp` may be on a RAM disk. `/var/tmp` should never be located on a RAM device.

4. The /usr Hierarchy

`/usr` is the second major section of the filesystem. `/usr` is shareable, read-only data. That means that `/usr` should be shareable between various machines running Linux and should not be written to. Any information that is machine-local or varies with time is stored elsewhere.

No large package (such as TEX and GNU Emacs) should use a direct subdirectory of `/usr`. Instead, there should be a subdirectory within `/usr/lib` (or `/usr/local/lib` if it was installed completely locally) for the purpose. An exception is made for the X Window System because of considerable precedent and widely-accepted practice.

`/usr` — Second major mount point (permanent)

```
|
├── X11R6      X Window System, version 11 release 6
├── X386       X Window System, version 11 release 5 on x86 Platforms
├── bin        Most user commands
├── dict       Word lists
├── doc        Miscellaneous documentation
├── etc        Site-wide system configuration
├── games      Games and educational binaries
├── include    Header files included by C programs
├── info       GNU Info system's primary directory
├── lib        Libraries
├── local      Local hierarchy (empty after main installation)
├── man        Online manuals
├── sbin       Non-vital system administration binaries
├── share      Architecture-independent data
└── src        Source code
```

The following symbolic links to directories may be present. This possibility is based on the need to preserve compatibility with older systems until all implementations can be assumed to use the `/var` hierarchy.

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

Once a system no longer requires any one of the above symbolic links, the link may be removed, if desired. Notably, it takes little effort to completely remove `/usr/preserve` since only `ex` and `vi` use it.

4.1 /usr/X11R6 : X Window System, Version 11 Release 6

This hierarchy is reserved for the X Window System, version 11 release 6, and related files.

`/usr/X11R6` — X Window System (version 11 release 6)

```
|
|— bin
|— doc
|— include
|— lib
`— man
```

To simplify matters and make XFree86 more compatible with the X Window System on other systems, the following symbolic links should be present:

```
/usr/bin/X11 -> /usr/X11R6/bin
/usr/lib/X11 -> /usr/X11R6/lib/X11
/usr/include/X11 -> /usr/X11R6/include/X11
```

In general, software should not be installed or managed via the above symbolic links. They are intended for utilization by users only. The difficulty is related to the release version of the X Window System — in transitional periods, it is impossible to know what release of X11 is in use. For the same reason, there should not be a symbolic link from `/usr/X11` pointing to the current X Window System hierarchy.

4.2 /usr/X386 : X Window System, Version 11 Release 5, on x86 platforms

This hierarchy is generally identical to `/usr/X11R6`, except that the `/usr` symbolic links should be absent if `/usr/X11R6` is installed.

4.3 /usr/bin : Most user commands

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

`/usr/bin` — Binaries that are not needed in single-user mode

|— `mh`	Commands for the MH mail handling system
`— `X11`	Symlink to `/usr/X11R6/bin`

Because shell script interpreters (invoked with `#!<path>` on the first line of a shell script) cannot rely on a path, it is advantageous to standardize the location of them. The Bourne shell and C-shell interpreters are already fixed in `/bin`, but Perl, Python, and Tcl are often found in many different places. `/usr/bin/perl`, `/usr/bin/python`, and `/usr/bin/tcl` should reference the `perl`, `python`, and `tcl` shell interpreters, respectively. They may be symlinks to the physical location of the shell interpreters.

4.4 /usr/dict : Word lists

Recommended 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).

Other word lists, such as the `web2` "dictionary" should be included here, if present.

The rationale behind having only word lists here is that they are the only files common to all spell checkers.

4.5 /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 recommend that all Linux systems incorporate it.

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

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

This is where all of the system's general-use include files for the C and C++ programming languages should be placed.

/usr/include — Include files

├── X11	Symlink to /usr/X11R6/include/X11
├── arpa	ARPAnet defined protocol definitions
├── asm	Symlink to /usr/src/linux/include/asm-<arch>
├── bsd	BSD compatibility include files
├── g++	GNU C++ 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 definitions
├── netinet	TCP/IP specific definitions
├── 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

The arpa subdirectory contains protocol header definitions for the ARPAnet protocols, TCP/IP conversion functions, definitions for ftp, telnet prototypes, and similar material.

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

The netinet subdirectory contains INET (DARPA Internet, which is also known as TCP/IP) specific definitions.

ARRL AX.25 is better known as packet radio. The Novell IPX/SPX protocols are part of the Novell NetWare file services.

4.7 /usr/lib : Libraries for programming and packages

/usr/lib includes object libraries, compiler program binaries, and static data of various kinds — both executable code (for example, GCC's internal binaries are located under /usr/lib/gcc-lib) and other types of data.

`/usr/lib` — Libraries for programming and packages

├── `X11`	Symbolic link to `/usr/X11R6/lib/X11`
├── `emacs`	Static support files for the GNU Emacs editor
├── `games`	Static data files for `/usr/games`
├── `groff`	Libraries/directories for GNU groff
├── `gcc-lib`	System specific files/directories for GCC
├── `kbd`	Keyboard translation tables and related data
├── `mh`	Libraries for the MH mail handling system
├── `news`	Cnews/INN
├── `smail`	Smail
├── `terminfo`	Directories for terminfo database
├── `texmf`	TeX/MF (and LaTeX) data libraries
├── `uucp`	Commands for UUCP
└── `zoneinfo`	Timezone information and configuration

Historically, `/usr/lib` has also included some executable commands such as `sendmail` and `makewhatis`.

Since `makewhatis` is not referenced by other programs, there is no problem with moving it to a binary directory. Since users have good cause to use `makewhatis`, `/usr/bin` is where it belongs. The `catman` binary (which replaces the `makewhatis` script on many Linux systems) should also be placed in `/usr/bin`.

The `sendmail` binary is referenced by many programs by its historical name, `/usr/lib/sendmail`. This should be a symbolic link to the standard location for mail transfer agents with a sendmail-compatible command line interface, `/usr/sbin/sendmail`.

Systems using Smail should place Smail in `/usr/sbin/smail`, and `/usr/sbin/sendmail` should be a symbolic link to it.

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 executable by normal users.

Any program or package which contains or requires data that doesn't need to be modified should store that data in `/usr/lib` (or `/usr/local/lib`, if installed locally). It is recommended that a subdirectory be used in `/usr/lib` for this purpose.

Game data stored in `/usr/lib/games` should be purely static data. Any modifiable files, such as score files, game play logs, and so forth, should be placed in `/var/lib`. If necessary for compatibility with old BSD-style games, a symlink from `/usr/games/lib` to `/usr/lib/games` can be used.

Note: No host-specific data for the X Window System should be stored in `/usr/lib/X11` (which is really `/usr/X11R6/lib/X11`). Host-specific configuration files such as `Xconfig` or `XF86Config` should be stored in `/etc/X11`. This should include configuration data such as `system.twmrc` even if it is only made a symbolic link to a more global configuration file (perhaps in `/usr/etc/X11` or `/usr/X11R6/lib/X11`).

4.8 /usr/local : Local hierarchy

The `/usr/local` hierarchy is for use by the system administrator when installing software locally. It needs to be safe from being overwritten when the system software is updated. It may be used for programs and data that are shareable amongst a group of machines, but not found in `/usr`.

`/usr/local` — Local hierarchy

├── `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

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`.

Note that software placed in `/` or `/usr` may be overwritten by system upgrades (though we recommend that distributions do not overwrite data in `/etc` under these circumstances). For this reason, local software should not be placed outside of `/usr/local` without good reason.

4.9 /usr/man : Manual pages

This section details the organization for manual pages throughout the system, including `/usr/man`.

Manual pages are stored in `<mandir>/<locale>/man[1-9]`. An explanation of `<mandir>` and `<locale>` is given below.

`<mandir>/<locale>` — A 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 primary `<mandir>` of the system is `/usr/man`. `/usr/man` contains manual information for commands and data under the `/` and `/usr` filesystems. Obviously, there are no manual pages in `/` because they are not required at boot time nor are they required in emergencies.

Provisions must be made in the structure of /usr/man to support manual pages which are written in different (or multiple) languages. These provisions must take into account the storage and reference of these manual pages. Relevant factors include language (including geographical-based differences), and character code set.

This naming of language subdirectories of /usr/man is based on Appendix E of the POSIX 1003.1 standard which describes the locale identification string — the most well accepted method to describe a cultural environment. The <locale> string is:

```
<language>[_<territory>][.<character-set>][,<version>]
```

The <language> field shall be taken from ISO 639 (a code for the representation of names of languages). It shall be two characters wide and specified with lowercase letters only.

The <territory> field shall be the two-letter code of ISO 3166 (a specification of representations of countries), if possible. (Most people are familiar with the two-letter codes used for the country codes in email addresses.[1]) It shall be two characters wide and specified with uppercase letters only.

The <character-set> field should represent the standard describing the character set. If the <character-set> field is just a numeric specification, the number represents the number of the international standard describing the character set. It is recommended that this be a numeric representation if possible (ISO standards, especially), not include additional punctuation symbols, and that any letters be in lowercase.

A parameter specifying a <version> of the profile may be placed after the <character-set> field, delimited by a comma. This may be used to discriminate between different cultural needs; for instance, dictionary order versus a more systems-oriented collating order. This standard recommends not using the <version> field, unless it is necessary.

Systems which use a unique language and code set for all manual pages may omit the <locale> substring and store all manual pages in <mandir>. For example, systems which only have English manual pages coded with ASCII, may store manual pages (the man[1-9] directories) directly in /usr/man. (That is the traditional circumstance and arrangement, in fact.)

Countries for which there is a well accepted standard character code set may omit the <character-set> field, but it is strongly recommended that it be included, especially for countries with several "competing" standards.

Various examples:

Language	Territory	Character Set	Directory
English	—	ASCII	/usr/man/en
English	United Kingdom	ASCII	/usr/man/en_GB
English	United States	ASCII	/usr/man/en_US
French	Canada	ISO 8859-1	/usr/man/fr_CA
French	France	ISO 8859-1	/usr/man/fr_FR
German	Germany	ISO 646-DE	/usr/man/de_DE.646de
German	Germany	ISO 6937	/usr/man/de_DE.6937

1. A major exception to this rule is the United Kingdom, which is ‘GB’ in the ISO 3166, but ‘UK’ for most email addresses.

German	Germany	ISO 8859-1	/usr/man/de_DE.88591
German	Switzerland	ISO 646-CH	/usr/man/de_CH.646ch
Japanese	Japan	JIS	/usr/man/ja_JP.jis
Japanese	Japan	SJIS	/usr/man/ja_JP.sjis
Japanese	Japan	UJIS (or EUC-J)	/usr/man/ja_JP.ujis

Manual pages for commands and data under `/usr/local` are stored in `/usr/local/man`. Manual pages for the X Window System are stored in `/usr/X11R6/man`. It follows that all manual page hierarchies in the system should have the same structure as `/usr/man`. Empty directories may be omitted from a manual page hierarchy. For example, if `/usr/local/man` has no manual pages in section 4 (Devices), then `/usr/local/man/man4` may be omitted.

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

The MH mail handling system manual pages should have `mh` appended to all manual page filenames. All X Window System manual pages should have an `x` appended to the filename.

The practice of placing various language manual pages in appropriate subdirectories of `/usr/man` also applies to the other manual page hierarchies, such as `/usr/local/man` and `/usr/X11R6/man`. (This portion of the standard also applies later in the section on the optional `/var/catman` structure.)

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 (requests for the Linux kernel to perform operations).
- `man3`: Library functions and subroutines
  Section 3 describes program library routines that are not direct calls to kernel services. This and chapter 2 are 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, this includes the device files found in `/dev` and the kernel interface to networking protocol support.
- `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.

4.10 /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. This includes internet daemons called by inetd (named in.*) such as in.telnetd and in.fingerd and rpc-based daemons handled by portmap (named rpc.*) such as rpc.nfsd andrpc.mountd.

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).

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

4.11 /usr/share : Architecture-independent data

Any specifications for /usr/share will be included in a supplementary draft 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 for more detailed discussion of /usr/share.

4.12 /usr/src : Source code

/usr/src — Source code

```
└─ linux            Source code for Linux 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-<arch>
/usr/src/linux/include/linux
```

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

`/usr/src/linux` may also be a symbolic link to a kernel source code tree.

5. The /var Hierarchy

/var — Variable data

├─ adm	System administrative data (obsolete), symbolic link to /var/log
├─ catman	Locally-formatted manual pages
├─ lib	Application state information
├─ local	Variable data of software from /usr/local
├─ lock	Lock files
├─ log	Log files and directories
├─ named	DNS files, networking only
├─ nis	Network Information Service (NIS) database files
├─ preserve	Saved files after crash or hang-up from ex or vi
├─ run	Files relevant to running processes
├─ spool	Directories for queued work to be performed later
└─ tmp	Temporary files, used to keep /tmp small

/var contains variable data files. This includes spool directories and files, administrative and logging data, and transient and temporary files.

Some portions of /var are not shareable between different systems. For instance, /var/log, /var/lock, and /var/run. Other portions are shareable, notably /var/spool/mail and /var/spool/news.

/var is specified here in order to make it possible to mount /usr read-only. Everything that once went into /usr that is written to during system operation (as opposed to installation and software maintenance) should be in /var.

If /var cannot be made a separate partition, it is often preferable to move /var out of the root partition and into the /usr partition. (This is sometimes done to reduce the size of the root partition or when space runs low in the root partition.) However, /var should not be linked to /usr because this makes separation of /usr and /var more difficult and is likely to create a naming conflict. Instead, link /var to /usr/var.

5.1 /var/adm : System logging and accounting files (obsolete)

This directory has been superseded by /var/log and other directories. It should be a symbolic link to /var/log until all programs no longer refer to any files in /var/adm.

utmp has been moved to /var/run. All log files have been moved to /var/log, including the wtmp file.

Distribution packaging support should be stored in /var/lib/<name>.

Note: the /var/adm symbolic link should not be necessary on most linux-i386 ELF systems since the change was introduced before ELF was released to the public.

5.2 /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 writable (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. We recommend that most sites 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 an unformatted manual page that normally appears in /usr/<path1>/man/man[1-9], the cached formatted version should go in /var/catman/<path2>/cat[1-9], where <path2> is <path1>. The <path1> and <path2> components are absent in the case of /usr/man and /var/catman.

For example, /usr/man/man1/ls.1 is formatted into /var/catman/cat1/ls.1, and /usr/X11R6/man/<locale>/man3/XtClass.3x into /var/catman/X11R6/<locale>/cat3/XtClass.3x.

Man pages written to /var/catman/cat[1-9] may eventually be transferred to /usr/<path>/cat[1-9] or expired; likewise formatted man pages in /usr/<path>/cat[1-9] may be expired if they are not accessed for a period of time.

If preformatted manual pages come with a Linux system on read-only media (a CD-ROM, for instance), they shall be installed into /usr/<path>/cat[1-9]. /var/catman is reserved as a writeable cache for formatted manual pages.

5.3 /var/lib : Application state information

/var/lib — Application state information

├── emacs	State directory for Emacs
├── games	Variable game data (score files)
├── news	Variable files for Cnews/INN
├── texmf	Variable data associated with TEX
└── xdm	X display manager authentication files and error logs

/var/lib/<name> is the appropriate location for all distribution packaging support. Different Linux distributions may utilize different names, of course.

5.3.1 /var/lib/emacs

The GNU Emacs state directory, the location of architecture-independent data files that Emacs modifies while running, should be /var/lib. Presently, Emacs only locates its lock file directory under the state directory (in <statedir>/emacs/lock), but it may make more extensive use of the state directory in the future. Notably, it only requires the addition of a single option to the Emacs configure program to make this change (before compilation).

5.3.2 /var/lib/games

As well as the subdirectories listed above, any variable data relating to the games found in `/usr/games` should be placed here. `/var/lib/games` should hold the variable data previously found in `/usr/lib/games`; static data, such as help text, level descriptions, and so on, should remain in `/usr/lib/games`.

5.3.3 /var/lib/news

`/var/lib/news` should be used to store all the variable data associated with news servers such as Cnews and INN, including the history file, active file, and so forth.

5.3.4 /var/lib/texmf

`/var/lib/texmf` should be used to store the variable data associated with TEX. In particular, `/var/lib/texmf/fonts` will store all of the fonts which are automatically generated by `MakeTeXPK`.

There should be a link from `/usr/lib/texmf/fonts/tmp` to `/var/lib/texmf/fonts`. This link allows users to use single path `/usr/lib/texmf/fonts/tfm` when making changes to their TEXFONTS environment variable. (This is the default path for Karl Berry's TEX tools, distributed from `ftp.cs.umb.edu:/pub/tex`.[2] If another TEX distribution is used, a link from the appropriate font directory to `/var/lib/texmf/fonts` should be made.)

The `MakeTeXPK` that is distributed with `dvipsk` will place `.pk` files in `fonts/pk/<device>/<fontname>` (e.g., `fonts/pk/CanonCX/cmr10.300pk`). The `.pk` files can be periodically purged from the `/var/lib/texmf` tree, or can be moved into the `/usr/lib/texmf` tree. If automatic `.mf` or `.tfm` generators are used, they should place their data in the `mf` or `tfm` subdirectories of `/var/lib/texmf/fonts`.

5.3.5 /var/lib/xdm

`/var/lib/xdm` contains the variable data from `xdm`, which consists of the `xdm-errors` files and any `xdm` authority files. `xdm` binaries such as the `chooser` should still be placed in the historical location in `/usr/X11R6/lib/X11/xdm`. The `xdm-pid` file should be placed in `/var/lib/xdm` despite the existence of `/var/run`. The remaining files should be placed in `/etc/X11/xdm`.

5.4 /var/local : Variable data of software from /usr/local

This directory contains all variable data which is related to software found in `/usr/local`. Naturally, the implementation of this directory is left up to the site administrator. However, information which can be categorized into another `/var` directory should not be placed in `/var/local`. For example, all lock files still go into `/var/lock`.

2. The reason that Karl Berry's tools are mentioned is that they are the de-facto standard for UNIX installations of TEX. These tools are widely used in the Linux community.

5.5 /var/lock : Lock files

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

To preserve the ability to mount `/usr` read-only, no lock files should be placed on the `/usr` partition.

Device lock files, such as the serial device lock files which were originally found in either `/usr/spool/locks` or `/usr/spool/uucp`, should now be stored in `/var/lock`. The naming convention which should be used is `LCK..` followed by the base name of the device. For example, to lock `/dev/cua0` the file `LCK..cua0` would be created.

The format used for Linux device lock files should be the HDB UUCP lock file format. The HDB format is to store the process identifier (PID) as a ten byte ASCII decimal number, with a trailing newline. For example, if process 1230 holds a lock file, it would contain the eleven characters: space, space, space, space, space, space, one, two, three, zero, and newline.

Then, anything wishing to use `/dev/cua0` can read the lock file and act accordingly (all locks in `/var/lock` should be world-readable).

5.6 /var/log : Log files and directories

The directory contains miscellaneous log files. Most logs should be written to this directory or an appropriate subdirectory.

`lastlog`	record of last login of each user
`messages`	system messages from `syslogd`
`wtmp`	record of all logins and logouts

A symbolic link from `/var/log/utmp` to `/var/run/utmp` may be required until programs no longer refer to `/var/adm/utmp` (`/var/adm` is itself a transitional symbolic link to `/var/log`).

5.7 /var/named : DNS files

This directory contains all the working files of the Internet name server, `named`.

We recommend that `/etc/named.boot` be a symbolic link to `/var/named/named.boot` since `/etc/named.boot` is the default boot file if no arguments are given to `named`.

5.8 /var/nis : 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, belonging to British Telecommunications plc, and may not be used without permission.

5.9 /var/preserve : Saved files after crash or hang-up from ex or vi

This directory contains saved files generated by any unexpected termination of `ex`, `vi`, or their clones.

5.10 /var/run : Run-time variable files

This directory contains system information files describing the system since it was booted. Generally, files in this directory should be cleared (removed or truncated as appropriate) the beginning of the boot process.

Process identifier (PID) files, which were originally placed in `/etc`, are placed in `/var/run`. The naming convention for PID files is `<program-name>.pid`. For example, the `crond` PID file is named `/var/run/crond.pid`.

The internal format of PID files remains unchanged. The file should consist of the process identifier in ASCII-encoded decimal, followed by a newline character. For example, if `crond` was process number 25, `/var/run/crond.pid` would contain three characters: two, five, and newline.

Programs that read PID files should be somewhat flexible in what they accept; i.e., they should ignore extra whitespace, leading zeroes, absence of the trailing newline, or additional lines in the PID file. Programs that create PID files should use the simple specification located in the above paragraph.

The `utmp` file, which stores information about who is currently using the system, is located in this directory.

Programs that maintain transient UNIX-domain sockets should place them in this directory.

5.11 /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 lineprinter daemon, out-bound mail is spooled for delivery to remote systems, and UUCP files are spooled for transmission 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       User mailbox files
├── mqueue     Outgoing mail queue
├── news       News spool directory
├── rwho       Rwhod files
├── smail      Spool directories for smail
└── uucp       Spool directory for UUCP
```

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

5.11.1 /var/spool/lpd

`/var/spool/lpd` — Printer spool directory

```
└── <printer>  Spools 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 and named `lock`.

5.12 /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).

There should be a symbolic link from `/usr/tmp` to `/var/tmp`, for compatibility reasons.

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`, `route` }
- `/usr/bin`: any binaries a user will want to use, and which are not vital
  { `finger`, `rcp`, `rlogin`, `telnet`, etc. }
- `/usr/sbin`: any administrator only binaries that are not vital
  { `in.ftpd`, `inetd`, `lpd`, `portmap`, etc. }

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 the root filesystem small.

Configuration files for networking belong in `/etc`.

6.3 Architecture-independent Structures

The directory `/usr/share` typically contains architecture-independent files such as man-pages, timezone, terminfo information, etc. 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 being worked on.

6.4 Symbolic Links

There are a wide range of uses for symbolic links 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`. This change can be justified by recalling the main reason for having `/var`: to separate directories of files that vary with time and between different systems and machines from those that may be shared and read-only.

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`, and others). 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 ones that don't have floppy drives), may want to link `ifconfig`, `route`, `hostname`, and other networking utilities statically 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 body "`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.

As noted in the introduction, please do not send mail to the mailing list without first contacting the FSSTND coordinator or a listed contributor.

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@colorado.edu>
Ian Jackson	<ijackson@cus.cam.ac.uk>
Ian McCloghrie	<ian@ucsd.edu>
Daniel Quinlan	<Daniel.Quinlan@linux.org>
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	<sweh@spuddy.mew.co.uk>
Fred N. van Kempen	<waltje@infomagic.com>
John A. Martin	<jmartin@csc.com>
Chris Metcalf	<metcalf@lcs.mit.edu>
Ian Murdock	<imurdock@debian.org>
David C. Niemi	<niemidc@clark.net>

CONTENTS